SIAM J. Discrete Math.
Vol. 13, No. 1, pp. 2–24

# BOUNDS FOR DISPERSERS, EXTRACTORS, AND DEPTH-TWO SUPERCONCENTRATORS[*]

JAIKUMAR RADHAKRISHNAN[†] AND AMNON TA-SHMA[‡]

**Abstract.** We show that the size of the smallest depth-two $N$-superconcentrator is

$$\Theta(N \log^2 N/\log\log N).$$

Before this work, optimal bounds were known for all depths except two. For the upper bound, we build superconcentrators by putting together a small number of disperser graphs; these disperser graphs are obtained using a probabilistic argument. For obtaining lower bounds, we present two different methods. First, we show that superconcentrators contain several disjoint disperser graphs. When combined with the lower bound for disperser graphs of Kővari, Sós, and Turán, this gives an almost optimal lower bound of $\Omega(N(\log N/\log\log N)^2)$ on the size of $N$-superconcentrators. The second method, based on the work of Hansel, gives the optimal lower bound.

The method of Kővari, Sós, and Turán can be extended to give tight lower bounds for extractors, in terms of both the number of truly random bits needed to extract one additional bit and the unavoidable entropy loss in the system. If the input is an $n$-bit source with min-entropy $k$ and the output is required to be within a distance of $\epsilon$ from uniform distribution, then to extract even one additional bit, one must invest at least $\log(n-k) + 2\log(1/\epsilon) - O(1)$ truly random bits; to obtain $m$ output bits one must invest at least $m - k + 2\log(1/\epsilon) - O(1)$. Thus, there is a loss of $2\log(1/\epsilon)$ bits during the extraction. Interestingly, in the case of dispersers this loss in entropy is only about $\log\log(1/\epsilon)$.

**Key words.** dispersers, extractors, superconcentrators, entropy loss

**AMS subject classifications.** 94C15, 05C35

**PII.** S0895480197329508

## 1. Introduction.

*Superconcentrators.* An *N-superconcentrator* is a directed graph with $N$ distinguished vertices called *inputs*, and $N$ other distinguished vertices called *outputs*, such that for any $1 \le k \le N$, any set $X$ of $k$ inputs and any set $Y$ of $k$ outputs, there exist $k$ vertex-disjoint paths from $X$ to $Y$. The *size* of a superconcentrator $G$ is the number of edges in it, and the *depth* of $G$ is the number of edges in the longest path from an input to an output.

Superconcentrators were studied originally to show lower bounds in circuit complexity. Valiant [23] showed that there exist $N$-superconcentrators of size $O(N)$; Pippenger [16] showed that there exist $N$-superconcentrators of size $O(N)$ and depth $O(\log N)$. On the other hand, Pippenger [17] showed that every depth-two $N$-superconcentrators has size $\Omega(N \log^2 N)$. This raised the question of the exact tradeoff between depth and size, which attracted much research during the last two decades [17, 7, 19, 1]. Table 1.1 gives a summary of the results. Here $\lambda(d, N)$ is the inverse of

[*]Received by the editors November 3, 1997; accepted for publication (in revised form) July 6, 1999; published electronically January 13, 2000. A preliminary version of this paper appeared as *Tight bounds for depth-two superconcentrators*, in 38th Annual IEEE Symposium on Foundations of Computer Science, 1997, pp. 585–594.

http://www.siam.org/journals/sidma/13-1/32950.html

[†]Tata Institute of Fundamental Research, Homi Bhabha Road, Mumbai 400 005, India (jaikumar@tcs.tifr.res.in). This work of this author was done while visiting the Institute of Computer Science, Hebrew University, Jerusalem, Israel.

[‡]International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704 (amnon@icsi.berkeley.edu). The work of this author was done while at the Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel. The research of this author was supported by a Phil Zacharia postdoctoral fellowship.

TABLE 1.1

| Depth | Size |
|---|---|
| 2 | $O(N \log^2 N)$ [17], $\Omega(N \log^{3/2} N)$ [1] |
| 3 | $\Theta(N \log\log N)$ [1] |
| $4, 5$ | $\Theta(N \log^* N)$ [7, 19] |
| $2d$, $2d+1$ | $\Theta(N\lambda(d, N))$ [7, 19] |
| $\Theta(\beta(N))$ | $\Theta(N)$ [7] |

functions in the Ackerman hierarchy: $\lambda(1, N)$ behaves like $\log N$, $\lambda(2, N)$ behaves like $\log^* N$. In general, $\lambda(d, N)$ decays very rapidly as $d$ grows; $\beta$ grows more slowly than the inverse of any primitive recursive function. We refer the reader to [7] for the definition of $\lambda$ and $\beta$. Thus, the dependence of the size on the depth was well understood for all depths except two. In this paper, we close this gap.

Let size$(N)$ denote the size of the smallest depth-two $N$-superconcentrator.

THEOREM 1.1 (main result). Size$(N) = \Theta\left(N \cdot \frac{\log^2 N}{\log\log N}\right)$.

For the upper bound, we use the method of Wigderson and Zuckerman [24], who showed how superconcentrators can be constructed using a type of expander graphs called *disperser graphs.*

DEFINITION 1.2 (disperser graphs [20, 6]). *A bipartite graph $G = (V_1 = [N], V_2 = [M], E)$ is a $(K, \epsilon)$-disperser graph, if for every $X \subseteq V_1$ of cardinality $K$, $|\Gamma(X)| > (1 - \epsilon)M$ (i.e., every large enough set in $V_1$* misses *less than an $\epsilon$ fraction of the vertices of $V_2$). The size of $G$ is $|E(G)|$.*

Nisan and Wigderson suggested (see [14]) that it might be possible to choose better parameters in the construction given in [24]. We implement their suggestion to obtain superconcentrators by putting together a smaller number of disperser graphs. These disperser graphs are obtained by probabilistic arguments.

*Remark.* The best explicit construction known gives $N$-superconcentrators of size $O(N(\log N)^{\mathrm{poly}(\log\log n)})$ (see [22, 13]).

We also observe a connection in the opposite direction: every depth-two superconcentrator contains many disjoint disperser graphs. Thus, lower bounds for disperser graphs imply lower bounds for depth-two superconcentrators. Using this method, we derive a simple $\Omega(N \cdot (\log N / \log\log N)^2)$ lower bound for depth-two $N$-superconcentrators; this is only a factor of $\log\log N$ away from the upper bound. To obtain the optimal lower bound, we use a method based on the work of Hansel [9] (see also Katona and Szemerédi [11]).

*Dispersers and extractors.* Disperser graphs arise from disperser functions. For a random variable $X$ taking values in $\{0,1\}^n$, the *min-entropy* of $X$ is given by

$$H_\infty(X) = \min_{x \in \{0,1\}^n} \log(1/\Pr[X = x]).$$

DEFINITION 1.3 (dispersers). $F : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a $(k, \epsilon)$-disperser if for all random variables $X$ taking values in $\{0,1\}^n$ with $H_\infty(X) \geq k$ and all $W \subseteq \{0,1\}^m$ of size at least $\epsilon 2^m$, we have*

$$\Pr[F(X, Z) \in W] \; > \; 0,$$

*where $Z$ is uniformly distributed over $\{0,1\}^d$.*

With $F : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, we associate the bipartite graph $G_F = (V_1, V_2, E)$, where $V_1 = \{0,1\}^n$, $V_2 = \{0,1\}^m$, and there is one edge of the form $(x, w)$

for each $z$ such that $f(x,z) = w$. In this graph, the degree of every vertex in $V_1$ is exactly $2^d$.

It is then easy to verify the following.

PROPOSITION 1.4. *$F$ is a $(k,\epsilon)$-disperser iff $G_F$ is a $(2^k,\epsilon)$-disperser graph.*

One special case of disperser graphs is the class of *highly-expanding graphs* [18], sometimes called *a-expanding graphs* [24]. These are bipartite graphs $G = (A = [N], B = [N], E)$, where for any two subsets $X \subseteq A$, $Y \subseteq B$ of size $a$, there is an edge between $X$ and $Y$. This is clearly equivalent to saying that $G$ is a $(K = a, \epsilon = \frac{a}{N})$-disperser graph. If $G$ is an $a$-expanding graph, then $\bar{G}$ (the bipartite complement of $G$) has no subgraph isomorphic to $K_{a,a}$. Such graphs have been studied extensively, and the problem of determining the maximum possible number of edges in these graphs is known as *the Zarankiewicz problem* (see [2, pp. 309–326]). An elegant averaging argument, due to Kővari, Sós, and Turán, gives good upper bounds on the number of edges in these graphs. When applied to disperser graphs, this method gives the following lower bounds.

THEOREM 1.5 (lower bounds for disperser graphs). *Let $G = (V_1 = [N], V_2 = [M], E)$ be a $(K,\epsilon)$-disperser. Denote by $\bar{D}$ the average degree of a vertex in $V_1$.*

(a) *Assume that $K < N$ and $\lceil \bar{D} \rceil \leq \frac{(1-\epsilon)M}{2}$ (i.e., $G$ is not trivial). If $\frac{1}{M} \leq \epsilon \leq \frac{1}{2}$, then $\bar{D} = \Omega(\frac{1}{\epsilon} \cdot \log \frac{N}{K})$, and if $\epsilon > \frac{1}{2}$, then $\bar{D} = \Omega(\frac{1}{\log(1/(1-\epsilon))} \cdot \log \frac{N}{K})$.*

(b) *Assume that $K \leq \frac{N}{2}$ and $\bar{D} \leq M/4$. Then, $\frac{\bar{D}K}{M} = \Omega(\log \frac{1}{\epsilon})$.*

Dispersers play an important role in reducing the error probability of algorithms that make *one-sided error*. In such applications, we typically have $\epsilon \leq 1/2$. Also, $a$-expanding graphs fall in this category, because there $\epsilon$ tends to 0. Hence, the case $\epsilon \leq 1/2$ is the one usually studied. However, for showing lower bounds for superconcentrators, we need to consider the case $\epsilon > 1/2$.

For reducing the error in algorithms that make *two-sided* error, one requires the function to satisfy stronger properties. Such functions are called extractors. For a survey of constructions and applications of dispersers and extractors, see the paper of Nisan [13].

For distributions $D_1$ and $D_2$ on $\{0,1\}^n$, the *variational distance* between $D_1$ and $D_2$ is given by

$$d(D_1, D_2) \;=\; \max_{S \subseteq \{0,1\}^n} |D_1(S) - D_2(S)|.$$

DEFINITION 1.6 (extractors). *$F : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a $(k,\epsilon)$-extractor, if for any distribution $X$ on $\{0,1\}^n$ with $H_\infty(X) \geq k$, we have that $d(F(X,U_d), U_m) < \epsilon$, where $U_d, U_m$ are random variables uniformly distributed over $\{0,1\}^d$ and $\{0,1\}^m$, respectively.*

In this view, an extractor uses $d$ random bits to extract $m$ quasi-random bits from a source with min-entropy $k$. Graphs arising from extractors have uniformity properties similar to random graphs.

DEFINITION 1.7 (extractor graphs). *A bipartite multigraph $G = (V_1 = [N], V_2 = [M], E)$ is a $(K,\epsilon)$-extractor with (left) degree $D$, if every $x \in V_1$ has degree $D$ and for every $X \subseteq V_1$ of size $K$, and any $W \subseteq V_2$,*

$$\left| \frac{|E(X,W)|}{|E(X,V_2)|} - \frac{|W|}{|V_2|} \right| < \epsilon.$$

*Here, $E(V,W)$ is the set of edges between $V$ and $W$ in $G$.*

We then have the following analogue of Proposition 1.4 (see Chor and Goldreich [4] and Zuckerman [25]).

PROPOSITION 1.8. *$F$ is a $(k, \epsilon)$-extractor iff $G_F$ is a $(2^k, \epsilon)$-extractor graph.*

THEOREM 1.9 (lower bounds for extractors). *There is a constant $C > 0$ such that the following holds. Let $G = (V_1 = [N], V_2 = [M], E)$ be a $(K, \epsilon)$-extractor with $K \leq \frac{N}{C}$. Then,*

(a) *if $\epsilon \leq \frac{1}{2}$ and $D \leq \frac{M}{2}$, then $D = \Omega(\frac{1}{\epsilon^2} \cdot \log(\frac{N}{K}))$;*

(b) *if $D \leq \frac{M}{4}$, then $\frac{DK}{M} = \Omega((\frac{1}{\epsilon})^2)$.*

In the terminology of functions this means that if $F : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a $(k, \epsilon)$-extractor, then $d \geq \log(n-k) + 2\log(\frac{1}{\epsilon}) - O(1)$ and $d+k-m \geq 2\log(\frac{1}{\epsilon}) - O(1)$. These two bounds have the following interpretation.

(a) In order to extract one extra random bit (i.e., having $m \geq d+1$) we need to invest at least $d \geq \log(n-k) + 2\log(\frac{1}{\epsilon}) - O(1)$ truly random bits (and $d \geq \log(n-k) + \log(\frac{1}{\epsilon}) - O(1)$ for dispersers).

(b) There is an unavoidable entropy loss in the system. The input to the extractor has entropy at least $k + d$ ($k$ in $X$ and $d$ in the truly random bits that we invest), while we get back only $m$ quasi-random bits. Thus, there is a loss of $k + d - m \geq 2\log(\frac{1}{\epsilon}) - O(1)$ bits. In the case of dispersers we have $d + k - m \geq \log\log(\frac{1}{\epsilon}) - O(1)$.

Surprisingly, the entropy loss (which can be compared to the heat wasted in a physical process) has different magnitudes in dispersers (about $\log\log\frac{1}{\epsilon}$) and extractors (about $2\log\frac{1}{\epsilon}$). In [8, 21], explicit $(k, \epsilon)$-extractors $F : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$, with $d = n - k + 2\log\frac{1}{\epsilon} + 2$ are constructed. Theorem 1.9 shows that the entropy loss of $2\log\frac{1}{\epsilon}$ in these extractors is unavoidable.

Theorems 1.5 and 1.9 improve the lower bounds shown by Nisan and Zuckerman [15]; they showed that $D \geq \max\{\log(\frac{N}{K}), \frac{1}{2\epsilon}\}$, and $\frac{DK}{M} \geq 1 - \epsilon$. Furthermore, our lower bounds match the upper bounds up to constant factors. Using standard probabilistic arguments [20, 26] one can show that our lower bounds are tight up to constant factors (for completeness we include the proofs in Appendix C).

THEOREM 1.10 (probabilistic constructions). *For every $1 < K \leq N$, $M > 0$ and $\epsilon > 0$ there exists a*

(a) *$(K, \epsilon)$-disperser graph $G = (V_1 = [N], V_2 = [M], E)$ with degree $D = \lceil \frac{1}{\epsilon}(\ln(\frac{N}{K}) + 1) + \frac{M}{K}(\ln(\frac{1}{\epsilon}) + 1) \rceil$,*

(b) *$(K, \epsilon)$-extractor graph $G = (V_1 = [N], V_2 = [M], E)$ with $D = \lceil \max\{\frac{1}{\epsilon^2}(\ln(\frac{N}{K}) + 1),\ \ln 2 \cdot \frac{M}{K} \cdot \frac{1}{\epsilon^2}\} \rceil$.*

**1.1. Organization of the paper.** In section 2, we first describe the lower bounds for dispersers. We describe the argument informally, leaving the formal proof for the appendix. Then we derive the lower bounds for extractors assuming a technical lemma on hypergeometric distributions. In section 3, we present the new upper and lower bounds for depth-two superconcentrators. The appendix has three parts. In the first we give the formal proof of the lower bounds for dispersers; in the second, we give the proof of the technical lemma used in section 2; in the third, we prove Theorem 1.10.

## 2. Bounds for dispersers and extractors.

In this section we present the lower bounds for disperser and extractor graphs. In the rest of this section, we will drop the word "graphs," and refer to them as dispersers and extractors. As stated earlier, the lower bounds for dispersers claimed in Theorem 1.5 follows from the bounds obtained by Kővari, Sós, and Turán for the Zarankiewicz problem. Instead of quoting their result directly, we will present the complete proof based on their method. This will help clarify the proof of Theorem 1.9, where we use the same method to show lower bounds for extractors.

In the rest of this section, we will use the following notation. For a bipartite graph $G = (V_1, V_2, E)$, $D(G)$ will denote the maximum degree of a vertex in $V_1$ and $\bar{D}(G)$ will denote the average degree of a vertex in $V_1$.

**2.1. Dispersers.** We now describe the proof of Theorem 1.5. Suppose $G = (V_1 = [N], V_2 = [M], E)$ is a $(K, \epsilon)$-disperser.

We first observe that part (b) follows from part (a). Let $G' = ([M], [N], E')$ be the graph obtained from $G$ by interchanging the roles of $V_1$ and $V_2$. Then, $G'$ is a $(\lceil \epsilon M \rceil, \frac{K}{N})$-disperser. Hence, by the first half of part (a) we have

$$\bar{D}(G') = \Omega\left(\frac{N}{K} \cdot \log\left(\frac{M}{\epsilon M}\right)\right).$$

Now $\bar{D}(G') = \bar{D}(G)N/M$; thus $\frac{\bar{D}(G)K}{M} = \Omega(\log(\frac{1}{\epsilon}))$, as claimed in part (b).

Now consider part (a). For a vertex $v$ of $G$ and a subset $X$ of vertices of $G$ we say that $X$ *misses* $v$ (and also $v$ misses $X$) if $\Gamma(v) \cap X = \emptyset$. Now, we let $B$ be a random subset of $V_2$ of size $L = \lceil \epsilon M \rceil$. For $v \in V_1$ with degree $d_v$, we have

$$\Pr[B \text{ misses } v] = \binom{M - d_v}{L} \binom{M}{L}^{-1}. \tag{2.1}$$

The expected number of vertices missed by $B$ (that is, $\mathbf{E}[|V_1 \setminus \Gamma(B)|]$) is the sum of these probabilities. Since $B$ can miss at most $K - 1$ vertices, we have

$$\sum_{v \in V_1} \binom{M - d_v}{L} \binom{M}{L}^{-1} \leq K - 1.$$

Note that $f(u) = \binom{u}{t}$ is a convex function of $u$. By applying Jensen's inequality, $f(\mathbf{E}[X]) \leq \mathbf{E}[f(X)]$, to the left-hand side above, we obtain

$$N \binom{M - \bar{D}}{L} \binom{M}{L}^{-1} \leq K - 1. \tag{2.2}$$

Our lower bounds follow from this inequality. We will now informally sketch the main points of the derivation; the formal proof is in the appendix.

For $\epsilon \leq 1/2$, the left-hand side of (2.2) is approximately $N \exp(-\epsilon \bar{D})$. Thus, we obtain the lower bound

$$\bar{D} \ \geq \ \frac{1}{\epsilon} \ln \frac{N}{K - 1}.$$

This strengthens the previous lower bound $D \geq \max\{1/(2\epsilon), \log(N/K)\}$ (due to Nisan and Zuckerman [15]).

For $\epsilon > 1/2$, the left-hand side of (2.2) is approximated better by $N\left(\frac{1-\epsilon}{2}\right)^{\bar{D}}$, i.e.,

$$\bar{D} \ \geq \ \frac{\log(N/(K-1))}{\log(1/(1-\epsilon)) + 1}. \qquad \square$$

**2.2. Extractors.** Since a $(K, \epsilon)$-extractor is also a $(K, \epsilon)$-disperser, the lower bounds for dispersers apply to extractors as well. We will now improve these bounds by exploiting the stronger properties of extractors. As in the proof of Theorem 1.5 we will show that Theorem 1.9(a) implies Theorem 1.9(b). To that end we define "slice" extractors.

**2.2.1. Slice-extractors.**

DEFINITION 2.1. *Let $G = (V_1 = [N], V_2 = [M], E)$ be a bipartite graph. For $v \in V_1$ and $B \subseteq V_2$, let*

$$\operatorname{disc}(v, B) = \Pr_{w \in \Gamma(v)}[w \in B] - \frac{|B|}{M},$$

*where $w$ is generated by picking a random edge leaving $v$. We say that $v$ $\epsilon$-*misses *$B$ (and also $B$ $\epsilon$-*misses *$v$) when $|\operatorname{disc}(v, B)| \geq \epsilon$.*

DEFINITION 2.2 (slice-extractor). *$G$ is a $(K, \epsilon, p)$-slice-extractor, if every $B \subseteq V_2$ of size $\lceil pM \rceil$, $\epsilon$-misses fewer than $K$ vertices of $V_1$.*

A slice-extractor seems to be weaker than an extractor because it is required to handle only subsets of $V_2$ of one fixed size, whereas an extractor must handle sets of all sizes. It is simple to show (see, e.g., [26])

*Claim* 2.3. *If $G = (V_1 = [N], V_2 = [M], E)$ is $(K, \epsilon)$-extractor, then $G$ is also a $(2K, \epsilon, p)$-slice-extractor for all $p$.*

In fact, we have the following lemma.

LEMMA 2.4. *Suppose $\lceil qM \rceil \leq \lceil pM \rceil < M/2$ and $G = (V_1 = [N], V_2 = [M], E)$ is a $(K, \epsilon, p)$-slice-extractor. Then $G$ is also a $(2K, 2\epsilon, q)$-slice-extractor.*

*Proof.* We will show that for any $A \subseteq V_1$ of size $K$ and any $S \subseteq V_2$ of size $\lceil qM \rceil$,

$$\left| \Pr_{w \in \Gamma(A)} [w \in S] - \frac{\lceil qM \rceil}{M} \right| \leq 2\epsilon. \tag{2.3}$$

This implies that $S$ can not $2\epsilon$-miss more than $2K$ vertices, and $G$ is a $(2K, 2\epsilon, q)$-slice-extractor.

We now show (2.3). Let $S_1, S_2 \subseteq V_2$ be subsets of size $\lceil qM \rceil$ and $T \subseteq V_2$ a subset of size $\lceil pM \rceil - qM$ disjoint from $S_1 \cup S_2$. Denote $T_1 = T \cup S_1$ and $T_2 = T \cup S_2$. Since $G$ is a $(K, \epsilon, p)$ slice-extractor, $|\Pr_{w \in \Gamma(A)}[w \in T_i] - p| \leq \epsilon$, for $i = 1, 2$. This implies that

$$\left| \Pr_{w \in \Gamma(A)}[w \in S_1] - \Pr_{w \in \Gamma(A)} [w \in S_2] \right| \leq 2\epsilon, \tag{2.4}$$

for every two subsets $S_1, S_2 \subseteq V_2$ of size $\lceil qM \rceil$. Now, pick $S \subseteq V_2$ of size $\lceil qM \rceil$ randomly and uniformly. Then,

$$\mathop{\mathbf{E}}_{S}[\Pr_{w \in \Gamma(A)}[w \in S]] = \frac{|S|}{M}.$$

Hence, exist sets $S^+$ and $S^-$ such that

$$\Pr_{w \in \Gamma(A)}[w \in S^-] \leq \frac{\lceil qM \rceil}{M} \leq \Pr_{w \in \Gamma(A)}[w \in S^+].$$

This, when combined with (2.4), implies (2.3). ∎

**2.2.2. Proof of Theorem 1.9(a).**

LEMMA 2.5. *There exists a constant $C > 0$, such that if $G = (V_1 = [N], V_2 = [M], E)$ is a $(K, \epsilon, p)$-slice-extractor with $D \leq M/2$, $K \leq N/C$, $p \leq 1/10$, $pM \geq 1$ and $\epsilon \leq p/25$, then*

$$\bar{D} \geq \frac{p}{C\epsilon^2} \ln \frac{N}{CK}.$$

*Proof.* We proceed as in the case of dispersers, by picking a random $\lceil pM \rceil$-sized subset $R \subseteq V_2$. To bound from below the probability that $R$ $\epsilon$-misses a vertex $v \in V_1$, we will use the following lemma, whose proof appears in the appendix.

LEMMA 2.6. *Let $R$ be a random subset of $[M]$ of size $qM$, $\Gamma$ be a nonempty subset of $[M]$ of size $D$, and $w : \Gamma \to [0,1]$ be a weight function such that $w(\Gamma) \overset{\text{def}}{=} \sum_{i \in \Gamma} w(i) = 1$. Suppose $\delta \leq 1/25$, $q \leq 1/4$ and $D \leq M/2$. Then,*

$$\Pr[|w(\Gamma \cap R) - q| \geq \delta q] \geq C^{-1} \exp(-C\delta^2 qD).$$

*Here $C$ is a constant independent of $\delta$, $q$, $D$, and $w$.*

Fix $v \in V_1$. Denote the set of $v$'s neighbors by $\Gamma$. Since in our definition of extractors we allow multiple edges, $|\Gamma|$ can be smaller than $d_v$ (the degree of $v$); so when we pick a random edge leaving $v$, some vertices in $\Gamma$ might be more likely to be visited than others. Let us define a weight function $w : \Gamma \to [0,1]$ by letting $w(b)$ be the number of multiple edges between $v$ and $b$ divided by $d_v$. Then, by definition

$$v \ \epsilon\text{-misses } R \ \text{ iff } \ |w(R \cap \Gamma) - p| \geq \epsilon = (\epsilon/p)p.$$

We take $D = d_v$, $\lceil pM \rceil = qM$ (so $q \leq \frac{1}{4}$) and $\delta = \frac{\epsilon}{p} \leq \frac{1}{25}$. For $C$ a large enough constant we have $\Pr[R \ \epsilon\text{-misses } v] \geq C^{-1} \exp(-C\delta^2 q d_v) \geq C^{-1} \exp(-C\epsilon^2 d_v/p)$. It follows that the expected number of vertices missed by $R$ is at least

$$\sum_{v \in V_1} C^{-1} \exp\left(-C\frac{\epsilon^2}{p} d_v\right).$$

Since $R$ never misses $K$ vertices, we have

$$\sum_{v \in V_1} C^{-1} \exp\left(-C\frac{\epsilon^2}{p} d_v\right) \leq K - 1.$$

Since $\exp(-Cx)$ is a convex function of $x$, Jensen's inequality implies that

$$N \ C^{-1} \exp\left(-C\frac{\epsilon^2}{p}\bar{D}\right) \leq K - 1.$$

By taking logarithms we obtain

$$\bar{D} \geq \frac{p}{C\epsilon^2} \ln \frac{N}{CK}. \qquad ∎$$

We now show Theorem 1.9(a). Note that we may assume that $\epsilon \leq 10^{-3}$ (say); otherwise the claim follows from the lower bound for dispersers proved in Theorem 1.5(a). But, if $\epsilon \leq 10^{-3}$, our claim follows immediately from Lemma 2.5 by taking $p = 1/10$.

**2.2.3. Proof of Theorem 1.9(b).** We next show that, as in the proof of Theorem 1.5, part (b) follows from part (a) by reversing the roles of $V_1$ and $V_2$.

*Claim* 2.7. *Suppose $G = (V_1 = [N], V_2 = [M], E)$ is a $(K, \epsilon)$-extractor. Let $G' = (V_1' = [M], V_2' = [N], E')$ be the graph obtained from $G$ by reversing the roles of $V_1$ and $V_2$. Suppose $p \geq K/N$ and $pN$ is an integer. Then, for all $T > 2$, $G'$ is a $(\frac{4M}{T}, \epsilon' = T\epsilon p, p)$-slice-extractor.*

*Proof.* Suppose $G'$ is not a $(\frac{4M}{T}, \epsilon' = T\epsilon p, p)$-slice-extractor. Then, there is some $B \subseteq V_2'$ of size $pN$ that $\epsilon'$-misses at least $\frac{4M}{T}$ vertices of $V_1'$. Let $A^- = \{v \in V_1' : \mathrm{disc}(v, B) \leq -\epsilon'\}$ and $A^+ = \{v \in V_1' : \mathrm{disc}(v, B) \geq \epsilon'\}$. One of these sets must have size at least $\frac{2M}{T}$, say $A^-$. Then

$$\begin{aligned} |E'(A^-, B)| &= \sum_{v \in A^-} |E'(v, B)| \\ &\leq \sum_{v \in A^-} d_v \left(\frac{|B|}{N} - \epsilon'\right) \\ &\leq |E'(A^-, V_2')| \cdot \left(\frac{|B|}{N} - \epsilon'\right). \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{|E'(A^-, B)|}{|B|D} &\leq \frac{|E'(A^-, V_2')|}{|B|D} \left(\frac{|B|}{N} - \epsilon'\right) \\ &= \frac{|E(V_1, A^-)|}{ND} \left(1 - \frac{N\epsilon'}{|B|}\right). \end{aligned} \tag{2.5}$$

Since $G$ is an extractor, we have

$$\frac{|E(V_1, A^-)|}{ND} \leq \frac{|A^-|}{M} + \epsilon. \tag{2.6}$$

We will now consider the sets $B \subseteq V_1$ and $A^- \subseteq V_2$ in the extractor $G$, and obtain a contradiction by showing that there are fewer edges between them than required. By combining (2.5) and (2.6), we obtain

$$\begin{aligned} \frac{|E(B, A^-)|}{|B|D} &\leq \left(\frac{|A^-|}{M} + \epsilon\right)\left(1 - \frac{N\epsilon'}{|B|}\right) \\ &\leq \frac{|A^-|}{M} + \epsilon - \frac{|A^-|}{M}\frac{N\epsilon'}{|B|} \\ &\leq \frac{|A^-|}{M} + \epsilon - 2\epsilon \\ &= \frac{|A^-|}{M} - \epsilon. \end{aligned}$$

(For the third inequality, we used $|A^-| \geq 2M/T$, $|B| = pN$ and $\epsilon' = T\epsilon p$.) But this contradicts our assumption that $G$ is a $(K, \epsilon)$-extractor. ◻

Finally, to obtain part (b) of Theorem 1.9, we choose $p = K/N$ and $T = 16C$ in the above claim and conclude that $G'$ is a $(M/(4C), 16\epsilon CK/N, K/N)$-slice-extractor. Since $D(G) \leq M/4$, we have $\bar{D}(G') \leq N/4$. By Markov's inequality, at least half the vertices of $V_1(G')$ have degree at most $N/2$. By restricting ourselves to the vertices of lowest degree, we obtain an $(M/(4C), \epsilon' = 16\epsilon CK/N, p = K/N)$-slice-extractor $G''$ with $|V_1(G'')| \geq M/2$, $V_2(G'') = N$, $D(G'') \leq N/2$ and $\bar{D}(G'') \leq \bar{D}(G')$. If $\epsilon < 1/(400C)$, we have $\epsilon' \leq p/25$. Then, by Lemma 2.5, we have $\bar{D}(G'') \geq \frac{p}{C\epsilon'^2} \ln(\frac{M/2}{CM/(4C)})$; i.e.,

$$\bar{D}(G'') = \Omega\left(\frac{p}{\epsilon^2 p^2}\right) = \Omega\left(\frac{N}{K}\frac{1}{\epsilon^2}\right).$$

Since $\bar{D}(G'') \leq N\bar{D}(G)/M$, we get $\frac{K\bar{D}(G)}{M} = \Omega(\frac{1}{\epsilon^2})$.

**3. Superconcentrators of depth two.** In this section, we show bounds on the size of depth-two superconcentrators. First, we show an $O(N\log^2 N/\log\log N)$ upper bound using the upper bounds for dispersers from Theorem 1.10 (a). Next, we show that the lower bounds on dispersers, shown in Theorem 1.5 (a), imply an (almost tight) $\Omega(N(\log N/\log\log N)^2)$ lower bound for superconcentrators. Finally, by using a different method, we improve this lower bound to $\Omega(N\log^2 N/\log\log N)$, matching the upper bound (up to constant factors).

Recall that size$(N)$ is the size of the smallest depth-two $N$-superconcentrator. It is enough to establish the claimed bounds assuming that $N$ is a power of two. For, let $2^n \leq N < 2^{n+1}$, where $n \geq 1$; then size$(2^n) \leq$ size$(N) \leq$ size$(2^{n+1})$.

**3.1. The upper bound.**

THEOREM 3.1. Size$(N) = O(N\log^2 N/\log\log N)$.

*Proof.* Our construction is based on a similar construction due to Wigderson and Zuckerman [24]. We build a depth-two graph $(A = [N], C, B = [N], E)$, where $C$ is the disjoint union of $C_i$, $i = 0, \ldots, \lceil \log_{\log N} N \rceil - 1$, where $|C_i| = 2\log^{i+1} N$. For every $i$ put a $(K = \log^i N, \epsilon = 1/4)$-disperser $D_i = (A, C_i, E_i)$, and another $(K, \epsilon)$-disperser between $B$ and $C_i$.

For $\log^i N \leq K \leq \log^{i+1} N$, for every $K$-set $X \subseteq A$, $\Gamma(X)$ covers at least 3/4 of $C_i$. Similarly, for every $K$-set $Y \subseteq B$, $\Gamma(Y)$ covers at least 3/4 of $C_i$. So, at least half of the vertices of $C_i$ are common neighbors of $X$ and $Y$. We thus have the following claim.

*Claim* 3.2. *Any two sets $X \subseteq A$ , $Y \subseteq B$ of size $K$ have at least $K$ common neighbors in $C$.*

However, as shown by Meshulam [12], Menger's theorem implies that this is sufficient for $G$ to be a superconcentrator (clearly, it is necessary). All that remains is to count the number of edges in $G$. By Theorem 1.10 (a), we may take $|E_i| = O(N\log N)$. Thus, we have $|E(G)| = \sum_i 2|E_i| = O(N\log^2 N/\log\log N)$. □

*Remark.* This construction differs from the one in [24] in only one respect: in their construction each $C_i$ takes care of all $K$-sets for $2^i \leq K < 2^{i+1}$, whereas in ours each $C_i$ takes care of all $K$-sets for $\log^i N \leq K < \log^{i+1} N$. The point is that constructing a disperser that works for just one $K$, in itself, requires average degree $\log N$ (as can be seen from Theorem 1.5). Furthermore, we can build dispersers that recover almost all the random bits we invest (see Theorem 1.10(a)). This enables us to hash sets of size $K$ into sets of size $K\log N$ and use fewer $C_i$'s.

**3.2. Lower bound via dispersers.** Now we show that any depth-two superconcentrator must contain $\Omega(\log N/\log\log N)$ disjoint disperser graphs and derive from this an $\Omega(N(\log N/\log\log N)^2)$ lower bound. The idea is as follows. Consider any depth-two superconcentrator $G = (A = [N], C, B = [N], E)$. By definition, for any $1 \leq K \leq N$, and any two subsets $X \subseteq A$, $Y \subseteq B$ of cardinality $K$, $X$, and $Y$

have at least $K$ common neighbors in $C$. In particular, if we fix a subset $X \subseteq A$ of cardinality $K$ and look at $\Gamma(X)$, we see that every $K$-subset of $B$ must have at least $K$ neighbors in $\Gamma(X)$. In other words, the induced graph on $\Gamma(X)$ and $B$ is a disperser. By doing this for different $K$'s we get several disjoint dispersers. Our lower bound then follows by applying the disperser lower bound to each of them.

THEOREM 3.3. $\text{Size}(N) = \Omega(N \cdot (\log N/\log\log N)^2)$.

*Proof.* Let $G = (A = [N], C, B = [N], E)$ be a depth-two $N$-superconcentrator. We will proceed in stages. In stage $i$, we will consider subsets of $A$ and $B$ of size $K_i = \log^{3i} N$. If $K_i \leq \sqrt{N}$ (i.e., $i \leq (1/6)\log N/\log\log N$), then we will show that there is a subset $C_i$ of the middle layer $C$ such that the number of edges between $B$ (the output vertices) and $C_i$ is at least $N\log N/\log\log N$. The sets $C_i$ will be *disjoint* for different values of $i$. Collecting the edges from the different $C_i$'s, we have

$$\begin{aligned}|E(G)| &\geq \left\lfloor \frac{\log N}{6\log\log N} \right\rfloor \cdot \Omega\left(N\frac{\log N}{\log\log N}\right) \\ &= \Omega\left(N\left(\frac{\log N}{\log\log N}\right)^2\right).\end{aligned}$$

Suppose the average degree in $A$ is $\bar{D}$. If $\bar{D} \geq \log^2 N$, then the number of edges between $A$ and $C$ is $N\bar{D} \geq N\log^2 N$ and we are done. So assume $\bar{D} \leq \log^2 N$.

Let $X_i \subseteq A$ be the set of $K_i = \log^{3i} N$ vertices with smallest degrees (breaking ties using some order on the vertices). Let $Z_i = \Gamma(X_i)$. Clearly $|Z_i| \leq K_i\bar{D}$. Let $C_i = Z_i \setminus (Z_1 \bigcup Z_2 \bigcup \cdots \bigcup Z_{i-1})$. Since $X_i \subseteq X_{i+1}$ for all $i$, we also have $Z_i = \Gamma(X_i) \subseteq \Gamma(X_{i+1}) = Z_{i+1}$; thus, $C_i = Z_i \setminus Z_{i-1}$.

*Claim* 3.4. *$G$ restricted to $B$ and $C_i$ is a $(K_i,\ \epsilon = 1 - \frac{1}{2\bar{D}})$-disperser.*

*Proof of claim.* Any two sets $X \subseteq A,\ Y \subseteq B$ of cardinality $K_i$ must have $K_i$ common neighbors in $C$. In particular, any set $Y \subseteq B$ of size $K_i$ has $K_i$ distinct neighbors in $Z_i = \Gamma(X_i)$, and therefore at least $K_i - |Z_{i-1}|$ distinct neighbors in $C_i$.

Notice that $K_i - |Z_{i-1}| \geq K_i - K_{i-1}\bar{D} > K_i/2$. Thus, in $G$ restricted to $B$ and $C_i$, any subset in $B$ of size $K_i$ has more than $K_i/2$ distinct neighbors. Thus, the claim follows if $K_i/2 \geq (1-\epsilon)|C_i|$. Indeed

$$(1-\epsilon)|C_i| = \frac{|C_i|}{2\bar{D}} \leq \frac{K_i\bar{D}}{2\bar{D}} = \frac{K_i}{2},$$

where the inequality follows from $|C_i| \leq |Z_i| \leq K_i\bar{D}$.

By Theorem 1.5(a), the number of edges between $A$ and $C_i$ is

$$\Omega\left(\frac{N \cdot \log\left(\frac{N}{K_i}\right)}{\log\left(\frac{1}{(1-\epsilon)}\right)}\right).$$

As long as $K_i \leq \sqrt{N}$, this is at least $\Omega(N\log N/\log\log N)$. Since the $C_i$'s are disjoint, we have obtained $\Omega(\log N/\log\log N)$ disjoint dispersers, each having $\Omega(N\log N/\log\log N)$ edges. ▯

**3.3. The improved lower bound.** If we look at the construction of Theorem 3.1, we see that sets of cardinality $K_i$ *communicate* mainly through a specific subset of $C$ denoted $C_i$. Furthermore, the vertices of $C_i$ can be identified using their degree: vertices in $C_i$ have degree about $N/K_i$. In our proof we will find this structure in the superconcentrator.

THEOREM 3.5. $\text{Size}(N) = \Omega(N \log^2 N / \log\log N)$.

*Proof.* Let $G = (A = [N], C, B = [N], E)$ be a depth-two $N$-superconcentrator. We assume that $N$ is large. As in the proof of Theorem 3.3, we proceed in stages. In stage $i$ ($i = 1, 2, \ldots$), we consider sets of size $K = K_i = \log^{4i} N$. Let

$$C_i = \left\{ w \in C : \frac{N}{K} \frac{1}{\log^2 N} \;\le\; \deg(w) \;<\; \frac{N}{K} \log^2 N \right\};$$
$$D_i = \left\{ w \in C : \deg(w) \;<\; \frac{N}{K} \frac{1}{\log^2 N} \right\}.$$

We will show that if $K \le N^{3/4}$ (i.e., $i \le \frac{3}{16}(\log N / \log\log N)$), then there are at least $\frac{1}{10} N \log N$ edges incident on $C_i$. Since the sets $C_i$ are disjoint for different values of $i$, we have

$$\begin{aligned} |E(G)| &\ge \left\lfloor \frac{3}{16}(\log N / \log\log N) \right\rfloor \cdot \frac{1}{10} N \log N \\ &= \Omega(N \log^2 N / \log\log N). \end{aligned}$$

It remains to show that the number of edges incident on $C_i$ is at least $\frac{1}{10} N \log N$. We assume that

$$|E(G)| \le \frac{1}{10} N \log^2 N; \tag{3.1}$$

otherwise the theorem follows immediately.

LEMMA A. *For every pair of $K$-sets $X \subseteq A$ and $Y \subseteq B$, there is a common neighbor in $C_i \cup D_i$.*

*Proof of lemma.* All vertices outside $C_i \cup D_i$ have degree at least $(N/K) \log^2 N$. Then, assumption (3.1) implies that the number of vertices outside $C_i \cup D_i$ is at most

$$\frac{|E(G)|}{(N/K) \log^2 N} \;\le\; \frac{K}{10}.$$

Since $X$ and $Y$ have at least $K$ common neighbors in the original graph, they must in fact have at least $\frac{9}{10} K$ common neighbors in $C_i \cup D_i$.

We wish to show that $G$ restricted to $A \cup B$ and $C_i$ cannot be sparse. We know from Lemma A that every pair of $K$-sets in $A \cup B$ has a common neighbor in $C_i \cup D_i$. Suppose that $G$ restricted to $A \cup B$ and $C_i$ is sparse. We will first obtain sets $S \subseteq A$ and $T \subseteq B$ such that $S$ and $T$ have no common neighbors in $C_i$. Then, all pairs of $K$-sets in $S$ and $T$ have to communicate via $D_i$; in other words, the bipartite graph induced on $S$ and $T$ by the connections via $D_i$ is a $K$-expanding graph. Since the number of edges incident on $D_i$ is small (because of (3.1)), $D_i$ cannot provide enough connections for such a $K$-expanding graph, leading to a contradiction. We thus have three tasks ahead of us.

- First, we need to show how to obtain sets $S$ and $T$. For this we use a method based on the work of Hansel [9]. We go through all vertices in $C_i$, and for each, either delete all its neighbors in $A$ or all its neighbors in $B$. Clearly, after this the surviving vertices in $A$ and $B$ do not have any common neighbor in $C_i$. It is remarkable that even after this severe destruction, we expect large subsets of vertices $S \subseteq A$ and $T \subseteq B$ to survive.

- Second, we need to show that the number of connections required between $S$ and $T$ is large. This follows from the fact that the bipartite graph induced on $S$ and $T$ by the connections via $D_i$ is a $K$-expanding graph.
- Finally, we need to show that the low degree vertices in $D_i$ cannot provide the required number of connections between $S$ and $T$. This will follow from the definition of $D_i$ and the fact that $S$ and $T$ are sufficiently random subsets of $A \cup B$.

Lemma B. *If $K = K_i \leq N^{3/4}$, then there are more than $\frac{1}{10} N \log N$ edges incident on $C_i$.*

*Proof of lemma.* For $u \in A \cup B$, let $d_u$ be the number of neighbors of $u$ in $C_i$. Let $A'$ be the set of $\frac{N}{2}$ vertices $u \in A$ with smallest $d_u$, and $B'$ be the set of $\frac{N}{2}$ vertices $v \in B$ with smallest $d_v$. We will prove that there is some $u \in A' \cup B'$ with $d_u > \frac{1}{5} \log N$. This implies the lemma for, say, $u \in A'$. Then, for all $u \in A \setminus A'$, $d_u > \frac{1}{5} \log N$, and we have more than $\frac{1}{10} N \log N$ edges incident on $C_i$, as required.

Now we have to prove that there is some $u \in A' \cup B'$ with large $d_u$. Otherwise, for all $u \in A' \cup B'$, $d_u \leq \frac{1}{5} \log N$. We will show that this contradicts (3.1).

For each $w \in C_i$, perform the following action (independently for each $w$):

– with probability $\frac{1}{2}$, delete all neighbors of $w$ from $A'$;

– with probability $\frac{1}{2}$, delete all neighbors of $w$ from $B'$.

Set $d = \frac{1}{5} \log N$. For each vertex $u$ in $A' \cup B'$ that survives, delete it independently with probability $1 - 2^{-(d-d_u)}$.

It is clear that after the above process, the probability that a vertex in $A' \cup B'$ survives is exactly $2^{-d}$. Let $S$ be the subset of vertices of $A'$ that survive and $T$ the subset of vertices of $B'$ that survive. Our construction ensures that $S$ and $T$ do not have a common neighbor in $C_i$. Lemma A then implies that every pair of $K$-sets in $S$ and $T$ has a common neighbor in $D_i$.

Consider the bipartite graph $H = (S, T, E)$, where $E$ consists of pairs $(u, v) \in S \times T$ such that $u$ and $v$ have a common neighbor in $D_i$. Then $H$ is a $K$-expanding graph (i.e., there is an edge joining every pair of $K$-sets in $S$ and $T$). It follows (see Lemma 3.8 below) that

$$|E(H)| \geq \frac{|S| \cdot |T|}{K} - |S| - |T|.$$

Thus, if $S$ and $T$ are large, the required number of edges in $H$ is also large. It is not hard to see that the expected size of $S$ and $T$ is large ($\frac{N 2^{-d}}{2} \sim N^{4/5} \gg K$). But we need $S$ and $T$ to be large *simultaneously.* Instead of ensuring this, it will be easier to directly estimate the average number of edges needed by $H$.

*Claim* 3.6.

$$\mathbf{E}[|E(H)|] \;\geq\; \mathbf{E}\left[\frac{|S| \cdot |T|}{K} - |S| - |T|\right] \;>\; \frac{N^2 2^{-2d}}{10K}.$$

This gives a lower bound on the average number of connections required between $S$ and $T$. Conversely, our next claim shows that if the number of edges in $G$ is small, then the average number of edges in $H$ (which is the number of connections between $S$ and $T$ passing via $D_i$) is small.

*Claim* 3.7.

$$\mathbf{E}[|E(H)|] \;\leq\; |E(G)| \cdot \frac{N}{K \log^2 N} \cdot 2^{-2d}.$$

Before proceeding to the proofs of these claims, let us complete the proof of Lemma B. Putting the two claims together we obtain

$$|E(G)| \cdot \frac{N}{K \log^2 N} \cdot 2^{-2d} > \frac{N^2 2^{-2d}}{10K},$$
$$\text{i.e.,} \quad |E(G)| > \frac{N}{10} \log^2 N.$$

But then $G$ has too many edges, contradicting (3.1).

*Proof of Claim* 3.6. We have $|S| = \sum_{u \in A'} X_u$ and $|T| = \sum_{v \in B'} Y_v$, where $X_u$ and $Y_v$ are the 0-1 indicator variables for the events "$u \in S$" and "$v \in T$," respectively.

For $u \in A'$ and $v \in B'$, $X_u$ and $Y_v$ are independent whenever $u$ and $v$ don't have a common neighbor in $C_i$, and $\mathbf{E}[X_u Y_v] = 2^{-2d}$. Since $d_u \leq \frac{1}{5} \log N$ and vertices in $C_i$ have degree at most $\frac{N}{K} \log^2 N$, each $X_u$ is independent of all but $\frac{N}{K} \log^2 N \cdot \frac{\log N}{5} < \frac{N (\log N)^3}{5K} < \frac{N}{4}$ of the $Y_v$'s. Therefore,

$$\begin{aligned} \mathbf{E}[|S| \cdot |T|] &= \sum_{u \in A', v \in B'} \mathbf{E}[X_u Y_v] \\ &\geq |A'| \cdot \frac{N}{4} \cdot 2^{-2d} \\ &= \frac{1}{8} N^2 2^{-2d}. \end{aligned}$$

Clearly, $\mathbf{E}[|S|] = \sum_{u \in A'} E[X_u] = \frac{N}{2} 2^{-d}$ and similarly $\mathbf{E}[|T|] = \frac{N}{2} 2^{-d}$. Thus we have

$$\mathbf{E}\left[\frac{|S| \cdot |T|}{K} - |S| - |T|\right] \geq \frac{N^2 2^{-2d}}{8K} - N 2^{-d} = \frac{N^2 2^{-2d}}{K}\left(\frac{1}{8} - \frac{2^d K}{N}\right) > \frac{N^2 2^{-2d}}{10K},$$

where the last inequality holds because $2^d K \leq N^{1/5} N^{3/4} = o(N)$ and $N$ is large.

*Proof of Claim* 3.7. Consider all pairs $(u, v) \in A' \times B'$, such that $u$ and $v$ have a common neighbor in $D_i$. Since the degree of a vertex in $D_i$ is at most $(N/K) \log^{-2} N$, the number of such pairs is at most

$$\begin{aligned} \sum_{w \in D_i} \deg(w)^2 &\leq \frac{N}{K \log^2 N} \sum_{w \in D_i} \deg(w) \\ &\leq \frac{N}{K \log^2 N} |E(G)|. \end{aligned} \tag{3.2}$$

As argued in the proof of Claim 3.6, for every pair $(u, v) \in A' \times B'$, if $u$ and $v$ don't have a common neighbor in $C_i$, then $\Pr[(u, v) \in S \times T] = 2^{-2d}$; conversely, if $u$ and $v$ have a common neighbor in $C_i$, then one of them will be deleted, and $\Pr[(u, v) \in S \times T] = 0$. Thus, in both cases $\Pr[(u, v) \in S \times T] \leq 2^{-2d}$. Our claim follows from this and (3.2) by linearity of expectation. ☐

Finally, we prove the density bound for $K$-expanding graphs.

*Claim* 3.8. *If* $(V_1 = [N_1], V_2 = [N_2], E)$ *is a* $K$*-expanding graph, then* $|E| \geq \frac{N_1 N_2}{K} - N_1 - N_2$

*Proof.* If either $N_1$ or $N_2$ is less than $K$, then the claim is trivial. Otherwise, obtain $\lfloor \frac{N_1}{K} \rfloor$ disjoint sets of size $K$ from $V_1$. No set of size $K$ can miss more than $K$ vertices in $V_2$; in particular, each such set has $N_2 - K$ edges incident on it. Collecting the contributions from the $\lfloor \frac{N_1}{K} \rfloor$ sets, we get at least $(\frac{N_1}{K}-1)(N_2-K) > \frac{N_1N_2}{K}-N_1-N_2$ edges. ∎

**Appendix A. Lower bounds for dispersers.**

We now complete the proof of Theorem 1.5. We will use inequality (2.2) derived in section 2.1.

First, consider the case $\epsilon \leq \frac{1}{2}$. To simplify the left-hand side of (2.2), we will use the inequality $\binom{a-c}{b}\binom{a}{b}^{-1} \geq (\frac{a-b-c+1}{a-b+1})^b$, valid whenever $a-b-c+1 \geq 0$. In our application, we have $\bar{D} \leq (1-\epsilon)M/2 \leq (1-\epsilon)M$ (an assumption in Theorem 1.5) and $L = \lceil \epsilon M \rceil \leq \epsilon M + 1$; thus, $M - \bar{D} - L + 1 \geq M - (1-\epsilon)M - \epsilon M - 1 + 1 = 0$. Then, (2.2) gives

$$N\left(\frac{M-L-\bar{D}+1}{M-L+1}\right)^L \leq K-1;$$

i.e.,

$$\begin{aligned}\frac{N}{K-1} &\leq \left(\frac{M-L+1}{M-L-\bar{D}+1}\right)^L \\ &= \left(1+\frac{\bar{D}}{M-L-\bar{D}+1}\right)^L \\ &\leq \exp(\bar{D}L/(M-\bar{D}-L+1)).\end{aligned}$$

On taking lns and solving for $\bar{D}$, we obtain

$$\bar{D} \geq \frac{(M-L+1)\ln(N/(K-1))}{L+\ln(N/(K-1))}.$$

If $\ln(N/(K-1)) > L$, then

$$\bar{D} > \frac{M-L+1}{2} \geq \frac{(1-\epsilon)M}{2},$$

contradicting our assumption. Thus, we may assume that $\ln(N/(K-1)) \leq L$. Then,

$$\begin{aligned}\bar{D} &\geq \frac{M-L+1}{2L}\ln\frac{N}{K-1} \\ &\geq \frac{(1-\epsilon)M}{2\epsilon M+2}\ln\frac{N}{K-1} \\ &\geq \frac{1}{8\epsilon}\ln\frac{N}{K-1}.\end{aligned}$$

For the case $\epsilon > \frac{1}{2}$, we must approximate the left-hand side of (2.2) differently. Since $\binom{a}{b}$ is a nondecreasing function of $a$, we have from (2.2) that

$$N\binom{M-\lceil\bar{D}\rceil}{L}\binom{M}{L}^{-1} \leq K-1.$$

Since $\binom{a-b}{c}\binom{a}{c}^{-1} = \binom{a-c}{b}\binom{a}{b}^{-1}$, we have

$$N\binom{M-L}{\lceil\bar{D}\rceil}\binom{M}{\lceil\bar{D}\rceil}^{-1} \leq K-1.$$

Now we use the inequality $\binom{a-b}{c}\binom{a}{c}^{-1} \geq (\frac{a-b-c+1}{a})^c$ and obtain

$$\begin{aligned} K-1 &\geq N\left(\frac{M-\lceil\bar{D}\rceil-L+1}{M}\right)^{\lceil\bar{D}\rceil} \\ &\geq N\left(\frac{M-L+1}{2M}\right)^{\lceil\bar{D}\rceil} \\ &\geq N\left(\frac{1-\epsilon}{2}\right)^{\lceil\bar{D}\rceil}. \end{aligned}$$

Thus,

$$\lceil\bar{D}\rceil \geq \frac{\log(N/(K-1))}{\log(2/(1-\epsilon))}. \qquad \square$$

**Appendix B. Lower bounds on deviation.**

This section is devoted to the proof of Lemma 2.6. We reproduce the lemma below for easy reference. Note that in the version below we use $\epsilon$ instead of $\delta$ and $p$ instead of $q$.

LEMMA 2.6. *Let $R$ be a random subset of $[M]$ of size $pM$, $\Gamma$ be a nonempty subset of $[M]$ of size $D$, and $w : \Gamma \to [0,1]$ be a weight function such that $w(\Gamma) \overset{\text{def}}{=} \sum_{i\in\Gamma} w(i) = 1$. Suppose $\epsilon \leq 1/25$, $p \leq 1/4$, and $D \leq M/2$. Then,*

$$\Pr[|w(\Gamma \cap R) - p| \geq \epsilon p] \geq C^{-1}\exp(-C\epsilon^2 pD).$$

*Here $C$ is a constant independent of $\epsilon$, $p$, $D$, and $w$.*

**B.1. Overview of the proof.** We have two cases based on the value of $p$.

*Case* 1 (*small* $p$). We first assume that $pD \leq 12$. In this case, we show that with constant probability $\Gamma \cap R = \emptyset$.

LEMMA B.1. $\Pr[\Gamma \cap R = \emptyset] \geq \exp(-50)$.

*Case* 2 (*large* $p$). We now assume that $pD > 12$. In this case, the proof has two main parts.

*Part* 1. The expected value of $|\Gamma \cap R|$ is easily seen to be $pD$. We first show lower bounds on the probability that $|\Gamma \cap R|$ deviates from this expected value by at least $\epsilon pD$.

LEMMA B.2. *If $pD \geq 12$, then for some constant $C_0$ (independent of $p$, $D$, $M$, and $\epsilon$)*

(a) $\Pr[|\Gamma \cap R| \leq (1-\epsilon)pD] \;\geq\; C_0^{-1}\exp(-C_0\epsilon^2 pD)$,

(b) $\Pr[|\Gamma \cap R| \geq (1+\epsilon)pD] \;\geq\; C_0^{-1}\exp(-C_0\epsilon^2 pD)$.

*Part* 2. Note that Part 1 suffices when the weights are all equal. Next, we consider a general distribution of weights. We show that when $|\Gamma \cap R|$ differs significantly from its expected value, then $w(\Gamma \cap R)$ is also likely to differ from its expected value. Note that the expected value of $w(\Gamma \cap R)$ is $p$.

LEMMA B.3. *Let $R^+$ be a random subset of $\Gamma$ of size $\lceil pD\rceil$ and $R^-$ be a random subset of size $\lfloor(1-4\epsilon)pD\rfloor$. Then, at least one of the following two statement holds:*

(a) $\Pr[w(R^-) \leq p(1-\epsilon)] \geq 1/25$,

(b) $\Pr[w(R^+) \geq p(1+\epsilon)] \geq 1/4$.

We first complete the proof of Lemma 2.6 assuming that Lemmas B.1, B.2, and B.3 hold. We shall justify Lemmas B.1, B.2, and B.3 after that.

*Proof of Lemma* 2.6. If $pD \leq 12$, then with constant probability, $\Gamma \cap R$ is empty. Clearly, whenever $\Gamma \cap R$ is empty, $|w(\Gamma \cap R) - p| \geq \epsilon p$. The claim follows from this.

Next, assume that $pD \geq 12$. We now use Lemma B.3 and conclude that at least one of the two statements, (a) and (b), of the lemma holds. Suppose (a) holds. Then, for all sizes $k \leq (1-4\epsilon)pD$, we have

$$\Pr[w(R_k) \leq p(1-\epsilon)] \ \geq \ \frac{1}{25}, \tag{B.1}$$

where $R_k$ is a random $k$-sized subset of $\Gamma$. Let $\mathcal{E}$ denote the event $|\Gamma \cap R| \leq (1-4\epsilon)pD$.

$$\begin{aligned}\Pr[|w(\Gamma \cap R) - p| \geq \epsilon p] &\geq \Pr[\mathcal{E}] \cdot \Pr[|w(\Gamma \cap R) - p| \geq \epsilon p \mid \mathcal{E}] \\ &\geq \Pr[\mathcal{E}] \cdot \Pr[w(\Gamma \cap R) \leq p(1-\epsilon) \mid \mathcal{E}] \\ &= \Pr[\mathcal{E}] \cdot \sum_{k=0}^{(1-4\epsilon)pD} \Pr[|\Gamma \cap R| = k \mid \mathcal{E}] \\ &\quad \times \Pr[w(\Gamma \cap R) \leq p(1-\epsilon) \mid |\Gamma \cap R| = k].\end{aligned}$$

By Lemma B.2 (a), we have $\Pr[\mathcal{E}] \geq C_0^{-1} \exp(-C_0 \epsilon^2 pD)$, for some constant $C_0$. Also, under the condition $|\Gamma \cap R| = k$, the random set $\Gamma \cap R$ has the same distribution as $R_k$. Then, using (B.1), we have

$$\begin{aligned}\Pr[|w(\Gamma \cap R) - p| \geq \epsilon p] &\geq \Pr[\mathcal{E}] \cdot \sum_{k=0}^{(1-4\epsilon)pD} \Pr[|\Gamma \cap R| = k \mid \mathcal{E}] \cdot \Pr[w(R_k) \leq p(1-\epsilon)] \\ &\geq C_0^{-1} \exp(-C_0 \epsilon^2 pD) \cdot \frac{1}{25} \\ &\geq C^{-1} \exp(-C \epsilon^2 pD)\end{aligned}$$

for $C = 25\, C_0$.

In the remaining case, statement (b) of Lemma B.3 holds, and the claim follows by a similar argument, this time using Lemma B.2 (b). ∎

**B.2. Proofs.** *Proof of Lemma* B.1. We have

$$\Pr[\Gamma \cap R| = \emptyset] \ = \ \binom{M-D}{pM}\binom{M}{pM}^{-1} \ \geq \ \left(\frac{M-D-pM}{M-pM}\right)^{pM} \ = \ \left(1 - \frac{D}{M-pM}\right)^{pM}.$$

Now, we use the inequality $1 - x \geq \exp(-x/(1-x))$, valid whenever $x < 1$. Thus,

$$\Pr[\Gamma \cap R| = \emptyset] \ \geq \ \exp\left(-\frac{DpM}{M-D-pM}\right) \ \geq \ \exp(-4pD) \ \geq \ \exp(-48).$$

For the second inequality, we use $D \leq M/2$ and $p \leq 1/4$, and for the last inequality, we use $pD \leq 12$. ∎

**Part 1.**

*Proof of Lemma* B.2. We will present the detailed argument only for part (a); the argument for part (b) is similar.

$$\Pr[|\Gamma \cap R| \geq (1-\epsilon)pD] = \sum_{k \leq (1-\epsilon)pD} \Pr[|\Gamma \cap R| = k].$$

We will be interested only in the last approximately $\sqrt{pqD}$ terms. Let

$$A = \lfloor (1-\epsilon)pD \rfloor \quad \text{and} \quad B = \left\lceil A - \sqrt{pqD} \right\rceil + 1.$$

Then,

$$\Pr[|\Gamma \cap R| \leq (1-\epsilon)pD] \geq \sum_{k=A}^{B} \Pr[|\Gamma \cap R| = k].$$

It can be verified that $\Pr[|\Gamma \cap R| = k]$ is an increasing function of $k$, for $A \leq k \leq B$. Thus,

$$\Pr[|\Gamma \cap R| \leq (1-\epsilon)pD] \geq (A - B + 1) \Pr[|\Gamma \cap R| = B]. \tag{B.2}$$

We will now estimate the two factors on the right-hand side.

- First, $A - B + 1 \geq \sqrt{pqD} - 2$, and since $pD \geq 12$ and $q \geq \frac{3}{4}$, we have $A - B + 1 \geq \frac{1}{3}\sqrt{pqD}$.
- To estimate the second term, we write $B = pd - h$; then we have

  $$\epsilon pD + \sqrt{pqD} - 2 \ \leq \ h \ \leq \ \epsilon pD + \sqrt{pqD}.$$

  Since $pD \geq 12$ and $q \geq \frac{3}{4}$, we have $1 \leq h < pD < qD$. Claim B.4 below shows that

  $$\Pr[|\Gamma \cap R| = B] \geq \frac{1}{\sqrt{4\pi pqD}} \exp\left(-\frac{4h^2}{pqD}\right).$$

Substituting these two estimates in (B.2), we obtain

$$\Pr[|\Gamma \cap R| \ \leq \ (1-\epsilon)pD] \geq \frac{1}{6\sqrt{\pi}} \exp\left(-\frac{4h^2}{pqD}\right).$$

To finish the proof of the lemma we consider two cases.

- If $\epsilon pD \leq \sqrt{pqD}$, then $h \leq 2\sqrt{pqD}$, and the bound above gives

  $$\Pr[|\Gamma \cap R| \leq (1-\epsilon)pD] \ \geq \ \frac{1}{6\sqrt{\pi}} \exp\left(-\frac{4 \cdot 4pqD}{pqD}\right) \ = \ \frac{1}{6\sqrt{\pi}} \exp(-16).$$

- Conversely, if $\epsilon pD \geq \sqrt{pqD}$, then $h \leq 2\epsilon pD$, and

  $$\Pr[|\Gamma \cap R| \leq (1-\epsilon)pD] \ \geq \ \frac{1}{6\sqrt{\pi}} \exp\left(-\frac{4 \cdot 4(\epsilon pD)^2}{pqD}\right) \ = \ \frac{1}{6\sqrt{\pi}} \exp(-32\epsilon^2 pD).$$

Now we return to the claim.

*Claim* B.4. *If* $1 \leq h < pD$, *then* $\Pr[|\Gamma \cap R| = pD - h] \geq \frac{1}{\sqrt{4\pi pqD}} \exp(-\frac{4h^2}{pqD})$.

*Proof of claim.* We have

$$\begin{aligned}
\Pr[|\Gamma \cap R| = pD - h] &= \frac{\binom{D}{pD-h}\binom{n-D}{p(n-D)+h}}{\binom{n}{pn}} \\
&\geq \sqrt{2\pi pqn}\ 2^{-nH(p)} \\
&\times \frac{1}{\sqrt{2\pi pqD}}\ 2^{DH(p)}\ \exp\left(-\frac{2h^2}{pqD}\right) \exp\left(-\frac{1}{6}\right) \\
&\times \frac{1}{\sqrt{2\pi pq(n-D)}}\ 2^{(n-D)H(p)}\ \exp\left(-\frac{2h^2}{pq(n-D)}\right) \exp\left(-\frac{1}{6}\right) \\
&\geq \frac{1}{\sqrt{4\pi pqD}} \exp\left(-\frac{4h^2}{pqD}\right).
\end{aligned}$$

For the last inequality we used the assumption $n - D \geq D$. For the first inequality we used the following bounds on binomial coefficients:

- For the denominator, we used the bound (see [3, p. 4])
$$\binom{n}{pn} \leq \frac{2^{nH(p)}}{\sqrt{2\pi pqn}}.$$
- For the numerator, we used
$$\binom{n}{pn+h}, \binom{n}{pn-h} \ > \ \frac{2^{nH(p)}}{\sqrt{2\pi pqn}} \exp\left(-\frac{2h^2}{pqn}\right) \exp\left(-\frac{1}{6}\right),$$
valid for $1 \leq h < \min\{pn, qn\}$. To justify this, we start from (see [3, p. 12])
$$\binom{n}{pn+h} > \frac{2^{nH(p)}}{\sqrt{2\pi pqn}} \exp\left(-\frac{h^2}{pqn}(\frac{1}{2} + \frac{hp}{2qn} + \frac{h^2 q}{3p^2n^2} + \frac{q}{n}) - \frac{1}{6}\right).$$
Since $h < qn$, we have $\frac{hp}{qn} < p \leq 1$; since $h \leq pn$, we have $h^2 q \leq h^2 \leq p^2n^2$; since $\min\{pn, qn\} \geq 1$, we have $n \geq 2$ and $\frac{q}{n} \leq \frac{1}{n} \leq \frac{1}{2}$. Thus, we get the required bound
$$\binom{n}{pn+h} > \frac{2^{nH(p)}}{\sqrt{2\pi pqn}} \exp\left(-\frac{2h^2}{pqn}\right) \exp\left(-\frac{1}{6}\right).$$
The bound on $\binom{n}{pn-h}$ follows from the above inequality because
$$\begin{aligned}\binom{n}{pn-h} = \binom{n}{n-pn+h} &= \binom{n}{qn+h} \\ &> \frac{2^{nH(p)}}{\sqrt{2\pi pqn}} \exp\left(-\frac{2h^2}{pqn}\right) \exp\left(-\frac{1}{6}\right). \qquad \square\end{aligned}$$

**Part 2.** We now present the proof of Lemma B.3. We shall first consider the case $p = 1/2$. The general case will follow from this.

LEMMA B.5. *Let $S$ be a random subset of $\Gamma$ of size $\ell = (\frac{1}{2} - 2\delta)D$. Assume $\delta \leq 1/12$. Then,*
$$\Pr\left[w(S) \leq \frac{1}{2} - \delta\right] \ \geq \ \frac{1}{12}.$$

Before we proceed to the proof of this lemma, let us deduce Lemma B.3 from it.

*Proof of Lemma* B.3. Let $X$ be a random subset of $\Gamma$ of size $2\lceil pD \rceil$. Let
$$\rho \ = \ \Pr[w(X) < 2p(1+\epsilon)].$$
We consider two cases based on the value of $\rho$:

1. $\rho \leq 1/2$. Let $X$ be as above. Let $Y$ be a random subset of $X$ of size $\lfloor (\frac{1}{2} - 4\epsilon)|X| \rfloor$. Thus, by Lemma B.5, for each value of $X$
$$\Pr\left[w(Y) \leq \left(\frac{1}{2} - 2\epsilon\right) w(X)\right] \geq \frac{1}{12}.$$
Since $\rho \geq 1/2$, with probability 1/24, we have $w(Y) < (\frac{1}{2} - 2\epsilon) \cdot 2p(1+\epsilon) < p(1-\epsilon)$. This implies that
$$\Pr[w(R^-) < p(1-\epsilon)] \ \geq \ \frac{1}{24}.$$

2. $\rho < 1/2$. In this case, let $Y$ be a random subset of $X$ of size $\lceil pD \rceil$. Whenever $w(X) \geq 2p(1+\epsilon)$, at least one of $Y$ and $X \setminus Y$ has weight of at least $p(1+\epsilon)$. Thus

$$\Pr[w(Y) > p(1+\epsilon) \mid w(X) \geq 2p(1+\epsilon)] \;\geq\; \frac{1}{2}.$$

Since $\rho < 1/2$, we have that $\Pr[w(X) \geq 2p(1+\epsilon)] > 1/2$. Thus

$$\Pr[w(Y) \geq p(1+\epsilon)] \;\geq\; \frac{1}{4}.$$

Now $Y$ is a random subset of $\Gamma$ of size $\lceil pD \rceil$; therefore,

$$\Pr[w(R^+) \geq p(1+\epsilon)] \;\geq\; \Pr[w(Y) \geq p(1+\epsilon)]. \qquad \square$$

*Proof of Lemma* B.5. Let $k = D - 2\ell$, Since $\delta \leq 1/12$, we have that

$$\ell = \left(\frac{1}{2} - 2\delta\right) D \;\geq\; \frac{1}{3}D \quad \text{and} \quad k = 4\delta D \;\leq\; \frac{1}{3}D.$$

Thus, $\ell \geq k$, and we may write $\ell = mk + k'$, where $m$ and $k$ are integers such that $1 \leq m \leq \ell/k$ and $0 \leq k' < k$. We now describe a procedure for generating sets of size $\ell$.

*Step* 1. Let $\pi = \{E_1, E_2, B_1, B_2, \ldots, B_{2m+1}\}$ be a partition of $\Gamma$ such that $|E_1|, |E_2| = k'$, and for $i = 1, 2, \ldots, 2m+1$, $|B_i| = k$. $E_1$ and $E_2$ will be referred to as the *exceptional* blocks.

*Step* 2. Pick a random permutation $\sigma$ of $[2m+1]$ and arrange the blocks in the order

$$\langle E_1, B_{\sigma(1)}, B_{\sigma(2)}, \ldots, B_{\sigma(2m+1)}, E_2 \rangle.$$

*Step* 3. Let

$$\begin{aligned}
\mathsf{Prefix}(\pi,\sigma) &= E_1 \cup B_{\sigma(1)} \cup B_{\sigma(2)} \cup \cdots \cup B_{\sigma(m)};\\
\mathsf{Middle}(\pi,\sigma) &= B_{\sigma(m+1)}; \text{ and}\\
\mathsf{Suffix}(\pi,\sigma) &= B_{\sigma(m+2)} \cup B_{\sigma(m+3)} \cup \cdots \cup B_{\sigma(2m+1)} \cup E_2.
\end{aligned}$$

If $\pi$ and $\sigma$ are chosen randomly, then $\mathsf{Prefix}(\pi,\sigma)$ and $\mathsf{Suffix}(\pi,\sigma)$ are random sets of size $\ell$ (i.e., they have the same distribution as the random set $S$ in the statement of the lemma).

We will prove the lemma by contradiction. Suppose the claim of the lemma is false.

$$\Pr_{\pi,\sigma}\left[w(\mathsf{Prefix}(\pi,\sigma)) > \frac{1}{2} - \delta\right] > \frac{11}{12} \quad \textit{and} \quad \Pr_{\pi,\sigma}\left[w(\mathsf{Suffix}(\pi,\sigma)) > \frac{1}{2} - \delta\right] > \frac{11}{12}.$$

It follows that with probability more than $5/6$, both $\mathsf{Prefix}$ and $\mathsf{Suffix}$ are *heavy* and consequently $w(\mathsf{Middle}(\pi,\sigma)) < 2\delta$. Let $\mathcal{E}(\pi,\sigma)$ denote this event; then,

$$\Pr_{\pi,\sigma}[\mathcal{E}(\pi,\sigma)] \;>\; \frac{5}{6}. \tag{B.3}$$

$\mathsf{Middle}(\pi,\sigma)$ is a random subset of $\Gamma$ of size $k$, and $E_1(\pi)$ and $E_2(\pi)$ are random subsets of $\Gamma$ of size $k'$. Since $k' < k$, we may conclude that

$$\Pr_{\pi}[w(E_1(\pi)) < 2\delta] > \frac{5}{6} \qquad \text{and} \qquad \Pr_{\pi}[w(E_2(\pi)) < 2\delta] > \frac{5}{6}.$$

It follows that with probability 2/3, both $E_1$ and $E_2$ are light. Let $\mathcal{F}(\pi)$ denote this event; then,

$$\Pr_{\pi}[\mathcal{F}(\pi)] \;>\; \frac{2}{3}. \tag{B.4}$$

Let $B^-$ be the block in $\pi$ with the smallest weight; let its weight be $w^-$. Let $B^+$ be the block in $\hat{\pi}$ with the largest weight; let its weight be $w^+$. For a partition $\pi$ and an ordering $\sigma$, let $\sigma'$ be the ordering derived from $\sigma$ by interchanging the positions of $B^+$ an $B^-$. Clearly, if $\sigma$ is chosen uniformly from the set of all permutations of $[2m+1]$, then $\sigma'$ is a random ordering with the same distribution as $\sigma$. It then follows from (B.3) that

$$\Pr_{\pi,\sigma}[\mathcal{E}(\pi,\sigma')] \;>\; \frac{5}{6}. \tag{B.5}$$

Now, from (B.3), (B.4), and (B.5) we have

$$\Pr_{\pi,\sigma}[\mathcal{E}(\pi,\sigma) \wedge \mathcal{F}(\pi) \wedge \mathcal{E}(\pi,\sigma')] > \frac{1}{3}. \tag{B.6}$$

The probability that $B^- \neq \mathsf{Middle}(\pi,\sigma)$ and $B^+$ does not appear on the same side of $\mathsf{Middle}$ as $B^-$ is

$$\frac{2m}{2m+1}\cdot\frac{m+1}{2m} = \frac{m}{2m+1} \geq \frac{2}{3},$$

where the inequality holds since $m \geq 1$. By combining this with (B.6), we conclude that with nonzero probability the following events take place simultaneously.

(a) $\mathsf{Prefix}(\pi,\sigma)$, $\mathsf{Suffix}(\pi,\sigma)$, $\mathsf{Prefix}(\pi,\sigma')$, and $\mathsf{Suffix}(\pi,\sigma')$ are all heavy;

(b) $E_1(\pi)$ and $E_2(\pi)$ are both light (i.e., have weight less than $2\delta$); and

(c) $B^- \neq \mathsf{Middle}(\pi,\sigma)$, and $B^-$ and $B^+$ do not appear on the same side of $\mathsf{Middle}$.

We will show that this is impossible. Suppose (say) $B^- \in \mathsf{Prefix}(\pi,\sigma)$. Then, from the definition of $\sigma'$ and (a), we have

$$w(\mathsf{Prefix}(\pi,\sigma')) \;=\; w(\mathsf{Prefix}(\pi,\sigma)) + w^+ - w^- \;>\; \frac{1}{2} - \delta + w^+ - w^-.$$

Since $\mathsf{Prefix}(\pi,\sigma')$ and $\mathsf{Suffix}(\pi,\sigma')$ are heavy, we have

$$\begin{aligned} 1 &= w(\mathsf{Prefix}(\pi,\sigma')) + w(\mathsf{Middle}(\pi,\sigma')) + w(\mathsf{Suffix}(\pi,\sigma')) \\ &> \left(\frac{1}{2} - \delta + w^+ - w^-\right) + w^- + \left(\frac{1}{2} - \delta\right) \\ &= 1 + w^+ - 2\delta. \end{aligned}$$

This is impossible, since, as we now show, $w^+ \geq 2\delta$ whenever (a) and (b) hold. For, by (a) $w(\mathsf{Prefix}(\pi,\sigma)) > 1/2 - \delta$, and by (b) $w(E_1(\pi)) < 2\delta$. Hence, one of the blocks $B_1, B_2 \ldots, B_m$, has weight more than

$$\frac{1}{m}\left(\frac{1}{2} - 3\delta\right) \;\geq\; \frac{k}{\ell}\left(\frac{1}{2} - 3\delta\right) \;\geq\; \frac{4\delta}{1/2 - \delta}\left(\frac{1}{2} - 3\delta\right) \;=\; 4\delta\,\frac{1-6\delta}{1-2\delta}.$$

Since $\delta \leq 1/12$, this is at least $2\delta$. $\square$

**Appendix C. Existence of dispersers and extractors.** In this section we prove Theorem 1.10.

*Dispersers.* First, consider the part (a) of Theorem 1.10. Sipser [20] showed the existence of disperser graphs with parameters $(N = m^{\log m}, M = m, K = m, D = 2\log^2 m, \epsilon = 1/2)$; we use his argument and obtain disperser graphs with parameters close to the lower bounds shown in section 2.

We construct a random graph $G = (V_1 = [N], V_2 = [M], E)$ by choosing $D$ random neighbors for each $v \in V_1$. Fix $\epsilon > 0$ and let $L = \lceil \epsilon M \rceil$. If $G$ is not a $(K, \epsilon)$-disperser, there is some subset of $V_2$ of size $L$ that misses some $K$ vertices of $V_1$. Thus, $\Pr[G$ is not a $(K,\epsilon)$-disperser$]$ is at most

$$\begin{aligned} &\binom{N}{K}\binom{M}{L}(1 - L/M)^{KD} \\ &< (eN/K)^K (eM/L)^L (1 - L/M)^{KD} \\ &\leq (eN/K)^K (eM/L)^L \exp(-LKD/M)). \qquad \text{(C.1)} \end{aligned}$$

(To justify the last inequality we use $1 - x \leq e^{-x}$.)

Plugging $D$ we have $(eN/K)^K \cdot (eM/L)^K \leq \exp(LKD/M)$. Therefore, by (C.1), we have

$$\Pr[G \text{ is not a disperser}] < 1.$$

So there is at least one instance of $G$ that meets our requirements. $\square$

*Extractors.* We now consider part (b) of Theorem 1.10. Essentially the same bounds were derived by Zuckerman [26]. We derive it again for completeness. We will use Definition 1.7. We will obtain a bipartite graph $G = (V_1 = [N], V_2 = [M], E)$ where all vertices in $V_1$ have the same degree $D$ such that for every $S \subseteq V_1$ of cardinality $K$, and any $R \subseteq V_2$,

$$\left| \frac{|E(S,R)|}{KD} - \frac{|R|}{M} \right| < \epsilon. \qquad \text{(C.2)}$$

We first observe that (C.2) can be replaced by a seemingly weaker condition.

*Claim* C.1. *If for every $S \subseteq V_1$ of size $K$, and any $R \subseteq V_2$, $|E(S,R)| < KD(\frac{|R|}{M} + \epsilon)$, then $G$ is a $(K, \epsilon)$-extractor.*

For, if there exist some $S$ and $R$ with $|E(S,R)| \leq KD(\frac{|R|}{M - \epsilon})$, then consider the set $\bar{R} = V_2 \setminus R$. We have $|E(S,\bar{R})| > KD(\frac{|\bar{R}|}{M} + \epsilon)$, contradicting the hypothesis of the claim.

Now we prove the existence of extractors. Consider the random graph $G = (V_1 = [N], V_2 = [M], E)$ obtained by choosing $D$ random neighbors with replacement for each $v \in V_1$.

Fix $S \subseteq V_1$ of size $K$ and $R \subseteq V_2$. Let $p = \frac{|R|}{M}$. We wish to estimate the probability (over the choices of the edges) that $|E(S,R)| \geq KD(p + \epsilon)$. The number of edges between $S$ and $R$ is the sum of $KD$ identically distributed independents random variables $X_1, X_2, \ldots, X_{KD}$, each taking the value 1 with probability $p = \frac{|R|}{M}$ and the value 0 with probability $1-p$. Thus, we can bound the probability of deviation using standard estimates for the binomial distribution:

$$\Pr[|E(S,R)| \geq (p + \epsilon)KD] \leq \exp(-2\epsilon^2 KD).$$

(This version of Chernoff's bounds appears in Chvátal [5] and Hoeffding [10].) Thus, $\Pr[G$ is not a $(K, \epsilon)$-extractor$]$ is at most

$$\begin{aligned} &\binom{N}{K} 2^M \exp(-2\epsilon^2 KD) \\ &< \left(\frac{eN}{K}\right)^K 2^M \exp(-2\epsilon^2 KD) \\ &= \left(e^{K(1+\ln(N/K))} \cdot e^{-\epsilon^2 KD}\right) \cdot \left(e^{M \ln 2} \cdot e^{-\epsilon^2 KD}\right). \end{aligned}$$

Since $D \geq \frac{1}{\epsilon^2}(1 + \ln(N/K))$ the first factor is at most 1; similarly, since $D \geq \frac{M \ln 2}{\epsilon^2 K}$ the second factor is at most 1. It follows that

$$\Pr[G \text{ is not an extractor}] < 1.$$

Hence, there is an instance of $G$ that satisfies our requirements. $\square$

**Acknowledgments.** We thank Roy Armoni, Oded Goldreich (who first asked us about entropy loss), Nati Linial, Avner Magen, Noam Nisan, Avi Wigderson, and Shiyu Zhou for many helpful discussions. We thank Aravind Srinivasan for helping us with the tail estimates. We are also grateful to the anonymous referee for many helpful comments. We thank David Zuckerman for Lemma 2.4 and its proof.

## REFERENCES

[1] N. Alon and P. Pudlák, *Superconcentrators of depth* 2 *and* 3*; odd levels help (rarely)*, J. Comput. System Sci., 48 (1994), pp. 194–202.
[2] B. Bollobas, *Extremal Graph Theory*, Academic Press, London, 1978.
[3] B. Bollobas, *Random Graphs*, Academic Press, New York, 1985.
[4] B. Chor and O. Goldreich, *Unbiased bits from sources of weak randomness and probabilistic communication complexity*, SIAM J. Comput., 17 (1988), pp. 230–261.
[5] V. Chvátal, *The tail of the hypergeometric distribution*, Discrete Math., 25 (1979), pp. 285–287.
[6] A. Cohen and A. Wigderson, *Dispersers, deterministic amplification and weak random sources*, in Proceedings IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamos, CA, 1989, pp. 14–19.
[7] D. Dolev, C. Dwork, N. Pippenger, and A. Wigderson, *Superconcentrators, generalizers and generalized connectors with limited depth*, in Proceedings ACM Symposium on Theory of Computing (STOC), ACM, New York, 1983, pp. 42–51.
[8] O. Goldreich and A. Wigderson, *Tiny families of functions with random properties: A quality-size trade-off for hashing*, Random Structures Algorithms, 11 (1997), pp. 315–343.
[9] G. Hansel, *Nombre minimal de contacts de fermature nécessaires pour réaliser une fonction booléenne symétrique de n variables*, C. R. Acad. Sci. Paris, 258 (1964), pp. 6037–6040.
[10] W. Hoeffding, *Probability inequalities for sums of bounded random variables,* J. Amer. Statist. Assoc., 58 (1963), pp. 13–30.
[11] G. Katona and E. Szemerédi, *On a problem of graph theory*, Studia Sci. Math. Hungar., 2 (1967), pp. 23–28.
[12] R. Meshulam, *A geometric construction of a superconcentrator of depth* 2, Theoret. Comput. Sci., 32 (1984), pp. 215–219.
[13] N. Nisan, *Refining randomness: How and why*, in Proceedings IEEE Symposium on Computational Complexity, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp 44–58.
[14] N. Nisan and A. Wigderson, *Research pearls in theory of computation: Homework* 2*, problem* 2*;* also available online from http://www.cs.huji.ac.il/course/pearls/.
[15] N. Nisan and D. Zuckerman, *Randomness is linear in space*, J. Comput. System Sci., 52 (1996), pp 43–52.
[16] N. Pippenger, *Superconcentrators*, SIAM J. Comput., 6 (1977), pp. 298–304.
[17] N. Pippenger, *Superconcentrators of depth* 2, J. Comput. System Sci., 24 (1982), pp 82–90.
[18] N. Pippenger, *Sorting and selecting in rounds*, SIAM J. Comput., 16 (1987), pp. 1032–1038.

[19] P. Pudlák, *Communication in bounded depth circuits*, Combinatorica, 14 (1994), pp. 203–216.
[20] M. Sipser, *Expanders, randomness, or time versus space*, J. Comput. System Sci., 36 (1988), pp. 379–383.
[21] A. Srinivasan and D. Zuckerman, *Computing with very weak random sources*, in Proceedings IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 264–275.
[22] A. Ta-Shma, *Refining Randomness*, Ph.D. thesis, The Hebrew University, Jerusalem, 1996.
[23] L. Valiant, *Graph-theoretic properties in computational complexity*, J. Comput. System Sci., 13 (1976), pp. 278–285.
[24] A. Wigderson and D. Zuckerman, *Expanders that beat the eigenvalue bound: Explicit construction and applications*, in Proceedings ACM Symposium on the Theory of Computing, ACM, New York, 1993, pp. 245–251.
[25] D. Zuckerman, *Simulating BPP using a general weak random source*, Algorithmica, 16 (1996), pp. 367–391.
[26] D. Zuckerman, *Randomness-optimal oblivious sampling*, Random Structures Algorithms, 11 (1997), pp. 345–367.

SIAM J. Discrete Math.
Vol. 13, No. 1, pp. 25–32

# NONHAMILTONIAN 3-CONNECTED CUBIC PLANAR GRAPHS[*]

R. E. L. ALDRED[†], S. BAU[‡], D. A. HOLTON[†], AND BRENDAN D. MCKAY[§]

**Abstract.** We establish that every cyclically 4-connected cubic planar graph of order at most 40 is hamiltonian. Furthermore, this bound is determined to be sharp, and we present all nonhamiltonian examples of order 42. In addition we list all nonhamiltonian cyclically 5-connected cubic planar graphs of order at most 52 and all nonhamiltonian 3-connected cubic planar graphs of girth 5 on at most 46 vertices. The fact that all 3-connected cubic planar graphs on at most 176 vertices and with face size at most 6 are hamiltonian is also verified.

**Key words.** nonhamiltonian, cubic, planar

**AMS subject classification.** 05C38

**PII.** S0895480198348665

## 1. Introduction.

In this paper we describe an investigation (making much use of computation) of cyclically $k$-connected cubic planar graphs ($CkCP$s) for $k = 4, 5$ and report the results. We shall also have occasion to consider cubic 3-connected planar graphs with no restriction on cyclic connectivity; these we refer to as $C3CP$s. The investigation extends the work of Holton and McKay in [10], in which the smallest order of a nonhamiltonian C3CP was shown to be 38. We provide answers to two questions raised in that paper, namely, the following:

(a) What is the smallest order of a nonhamiltonian C4CP?

(b) Is there more than one nonhamiltonian C5CP on 44 vertices?

The answer to the former question is determined to be 42, and all nonhamiltonian C4CPs of that order are presented. The latter question is answered in the negative, and a complete list of all nonhamiltonian C5CPs on at most 52 vertices is presented.

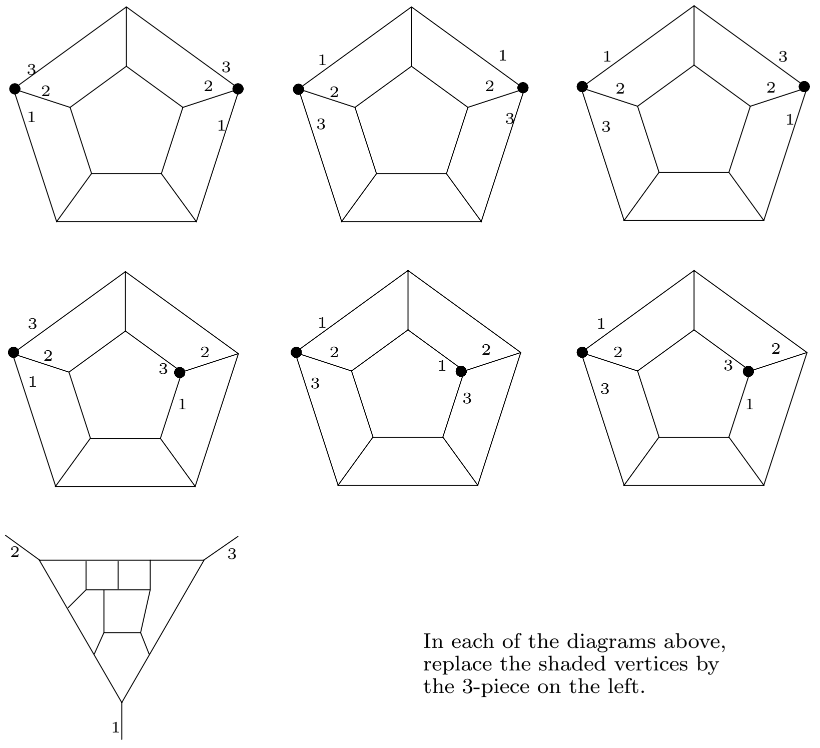
Before proceeding, we include some definitions and results from the existing literature as we shall make use of them in the rest of the paper. By a *$k$-gon* we mean a face of a plane graph bounded by $k$ edges. Note that a $k$-cycle is not necessarily a $k$-gon. By a *$k$-cut* we mean a set of $k$ edges whose removal leaves the graph disconnected and of which no proper subset has that property. The two components (and clearly there are only two) formed by the removal of a $k$-cut are called *$k$-pieces*. A $k$-cut is *nontrivial* if each of its $k$-pieces contains a cycle and is *essential* if it is nontrivial and each of its $k$-pieces contains more than $k$ vertices. A cubic graph is *cyclically $k$-connected* if it has no nontrivial $t$-cuts for $0 \le t \le k-1$, and it has *cyclic connectivity $k$* if, in addition, it has at least one nontrivial $k$-cut. We denote by $\lambda'(G)$ the value of $k$ such that the C3CP $G$ has cyclic connectivity $k$.

If $G$ is a hamiltonian C3CP, then an *a-edge* is an edge which is present in every hamiltonian cycle in $G$, while a *b-edge* is absent from every hamiltonian cycle in $G$.

To focus our search for nonhamiltonian C4CPs of smallest order we shall make use of the following theorem which formed the main result in [10].

[*]Received by the editors December 2, 1998; accepted for publication May 7, 1999; published electronically January 13, 2000.
http://www.siam.org/journals/sidma/13-1/34866.html

[†]Department of Mathematics and Statistics, University of Otago, P.O. Box 56, Dunedin, New Zealand (raldred@maths.otago.ac.nz, dholton@maths.otago.ac.nz).
[‡]Inner Mongolia Finance and Economics College, Hohhot, China.
[§]Department of Computer Science, Australian National University, Canberra, ACT 0200, Australia (bdm@cs.anu.edu.au).



In each of the diagrams above, replace the shaded vertices by the 3-piece on the left.

FIG. 1. *The* 38*-vertex nonhamiltonian C*3*CPs.*

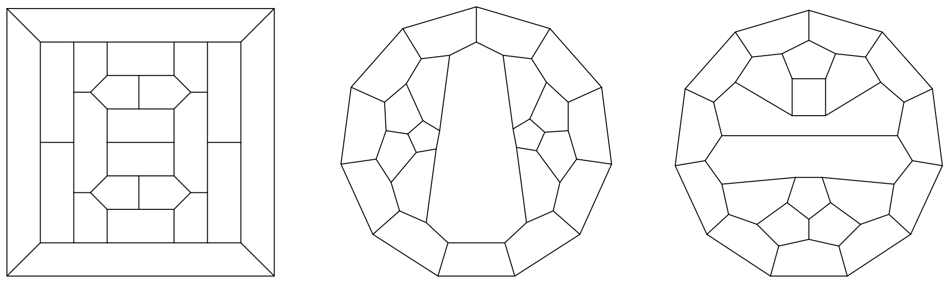


FIG. 2. *The* 42*-vertex nonhamiltonian C*4*CPs.*

THEOREM 1. *Every $C3CP$ of order at most* 36 *is hamiltonian. Furthermore, if $G$ is a nonhamiltonian $C3CP$ with* $38 \leq |V(G)| \leq 42$*, then one of the following is true:*

(a) *$G$ is one of the six $C3CP$s of order* 38 *and cyclic connectivity* 3 *shown in Figure* 1;

(b) *$G$ has* 40 *or* 42 *vertices and has a nontrivial* 3*-cut for which one of the components has at most five vertices;*

(c) $|V(G)| = 42$, $\lambda'(G) = 4$, *and $G$ has an essential* 4*-cut, one* 4*-piece of which consists of a pair of adjacent* 4*-faces while the other is obtained from a nonhamiltonian $C4CP$ on* 38 *vertices by deleting two adjacent vertices;*

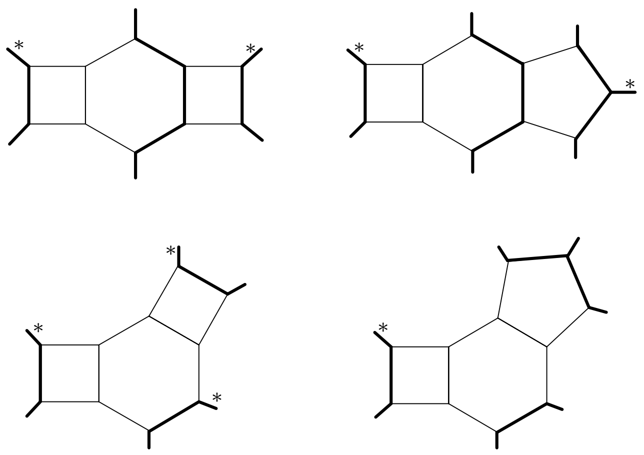


FIG. 3. *Some forbidden subgraphs.*

(d) *$G$ is one of the two graphs* NH42.b *and* NH42.c *in Figure* 2;

(e) $\lambda'(G) = 4$ *and $G$ has no essential* 4*-cut.*

Hence, if $G$ is a nonhamiltonian C4CP of smallest order and not either of the graphs NH42.b, NH42.c, then $\lambda'(G) = 4$, $G$ contains no essential 4-cut (i.e., the only nontrivial 4-cuts in $G$ are the edges adjacent to of 4-cycles, and there must be at least one such 4-cut) and $|V(G)| \in \{38, 40, 42\}$. (The graph NH42.a in Figure 2 is from Grünbaum [8] and is a nonhamiltonian C4CP on 42 vertices with no essential 4-cuts.)

**2. The cyclically 4-connected case.** Throughout this section we shall assume that $G$ is a nonhamiltonian C4CP of smallest order, other than NH42.b and NH42.c. In the light of the theorem above we know that $\lambda'(G) = 4$, $G$ contains no essential 4-cut, and $|V(G)| \in \{38, 40, 42\}$.

Subgraphs that $G$ cannot contain are called *forbidden* subgraphs. Since $G$ is cyclically 4-connected and contains no essential 4-cuts, triangles and adjacent 4-cycles are forbidden subgraphs. Similarly, cuts of sizes 1 and 2, nontrivial 3-cuts, and essential 4-cuts are *forbidden cuts.*

Following the method of Okamura [14], we wish to show that none of the four subgraphs in Figure 3 can occur in $G$. For each subgraph $H$ in Figure 3, the *reduction* $H(G)$ of $G$ *using* $H$ is obtained by edge reduction. Delete the edges drawn thinly in the figure, and then suppress the resulting vertices of degree 2. The resulting thick edges are then edges in $H(G)$. Notice that if an edge marked with an asterisk corresponds to an edge in a hamiltonian cycle in $H(G)$, then $G$ is hamiltonian.

THEOREM 2. *If $G$ is a nonhamiltonian C4CP of smallest order and contains a subgraph $H$, such that $H$ is one of the subgraphs in Figure* 3*, then $H(G)$, the reduction of $G$ using $H$, is a hamiltonian C3CP with a b-edge.*

*Proof.* We know that $G$ must have at most 42 vertices and that the reduction of $G$ by any of the subgraphs in Figure 3 must have 10 fewer vertices than $G$. Thus $H(G)$ must be hamiltonian. By the observation above, that a hamiltonian cycle through an edge in $H(G)$ corresponding to an edge with an asterisk lifts to a hamiltonian cycle of $G$, we see that $H(G)$ must contain a $b$-edge. The proof that $H(G)$ is 3-connected is as in [14]. □

Hence, if $G$ contains one of the subgraphs in Figure 3, then $G$ can be constructed

Table 1
*Numbers of C3CPs with b-edges.*

| Order of graphs | Without triangles | With triangles | Total |
|---|---|---|---|
| 24 | 1 | 0 | 1 |
| 26 | 7 | 24 | 31 |
| 28 | 26 | 483 | 509 |
| 30 | 146 | 6724 | 6870 |
| 32 | 776 | 79416 | 80192 |

from a smaller C3CP with $b$-edges using the inverse of some reduction by such a subgraph. Since the subgraphs in Figure 3 are derived from those of Okamura [14], we call such an inverse an *Okamura expansion*.

Our computation proceeded as follows.

*Phase* 1. Generate all 3-connected cubic planar graphs on up to 32 vertices which contain a $b$-edge but no triangle. The method used was that of Mohar [13], in conjunction with the graph isomorphism system described by McKay in [11].

*Phase* 2. Note that a C3CP $H$ with a triangle has a $b$-edge if and only if the C3CP obtained from $H$ by contracting the triangle has a $b$-edge. Thus the C3CPs on up to 32 vertices, with $b$-edges and triangles can be constructed from the graphs generated in Phase 1 on up to 30 vertices by successively expanding vertices to triangles. This was done, yielding graphs in the numbers indicated in Table 1. (Table 1 extends part of Table 1 of [10], where, it should be noted, a typographical error was made indicating that the number of C4CPs with essential 4-cuts was 5863 when it is actually 5865.)

*Phase* 3. Perform all possible Okamura expansions on the graphs obtained in Phases 1 and 2, discarding the new graphs which have so formed those with forbidden cuts.

*Phase* 4. Test the graphs yielded by Phase 3 for hamiltonicity, and discard those with hamilton cycles. (All hamiltonicity checking in our computations was done using the method of McKay [12].)

No graphs remained at the completion of Phase 4, proving the following theorem.

Theorem 3. *Let $G$ be a nonhamiltonian C4CP of order* 38, 40, *or* 42. *Then $G$ contains none of the subgraphs in Figure* 3.

*Phase* 5. Generate all of the C4CPs on at most 42 vertices with no pair of adjacent 4-gons, discarding those which contain any of the forbidden subgraphs in Figure 3. The small set of forbidden subgraphs provides an effective filter so that from approximately $1.5 \times 10^{10}$ graphs generated, only about five million graphs are yielded in this phase.

*Phase* 6. Test the graphs obtained from Phase 5 for hamiltonian cycles, discarding the hamiltonian graphs.

After Phase 6 there were three graphs left, which were precisely those of Figure 2.

Thus our computations establish the following result.

Theorem 4. *All C4CPs on at most* 40 *vertices are hamiltonian. Furthermore, the only nonhamiltonian C4CPs on* 42 *vertices are the three shown in Figure* 2.

**3. The cyclically 5-connected case.** In [7], Faulkner and Younger established that the smallest nonhamiltonian C5CP has 44 vertices. The nature of the search employed there was such that the graph known to exhibit this bound, NH44 of Figure 4, could not be guaranteed to be unique. We have now generated all C5CPs on up

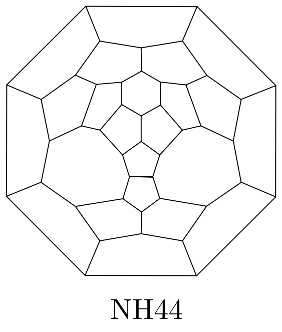


FIG. 4. *The* 44*-vertex nonhamiltonian C5CP.*

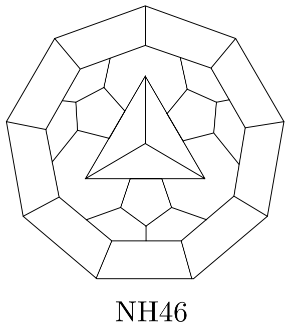


FIG. 5. *The* 46*-vertex nonhamiltonian C5CP.*

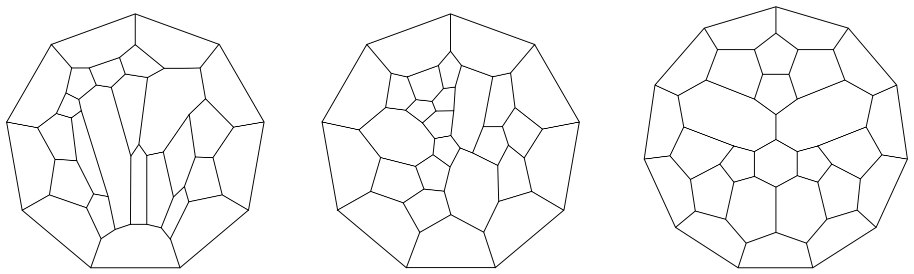


FIG. 6. *The* 50*-vertex nonhamiltonian C5CPs.*

to 52 vertices using the generation procedure of Barnette [1] and Butler [6]. The graphs were then checked for hamiltonian cycles. This search confirms that NH44 is the unique nonhamiltonian C5CP of smallest order. It also yielded a complete list of all nonhamiltonian C5CPs on at most 52 vertices. These are shown in Figures 4, 5, 6, and 7.

**4. Some questions arising.** The graphs yielded in the above searches suggested further questions, some of which we have partially pursued. As was noted, no smallest nonhamiltonian cubic planar cyclically $k$-connected graph can contain a triangle. Thus all such graphs have girth 4 or 5, regardless of the value of $k$. Apart from the C5CPs, all of the minimal graphs determined through the above procedures have girth 4. This raises a question: What is the smallest order of a nonhamiltonian C3CP of girth 5?

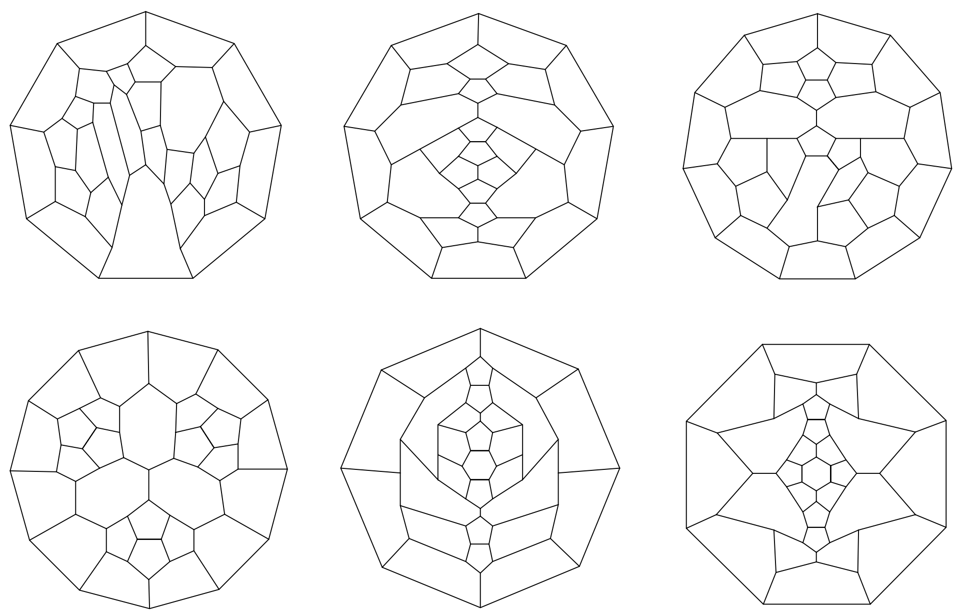


Fig. 7. *The* 52*-vertex nonhamiltonian C*5*CPs.*

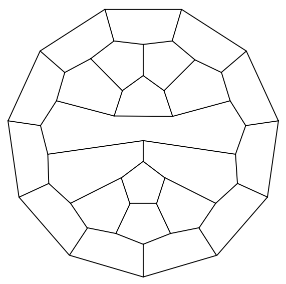


Fig. 8. *The other* 44*-vertex girth* 5 *nonhamiltonian C*4*CP.*

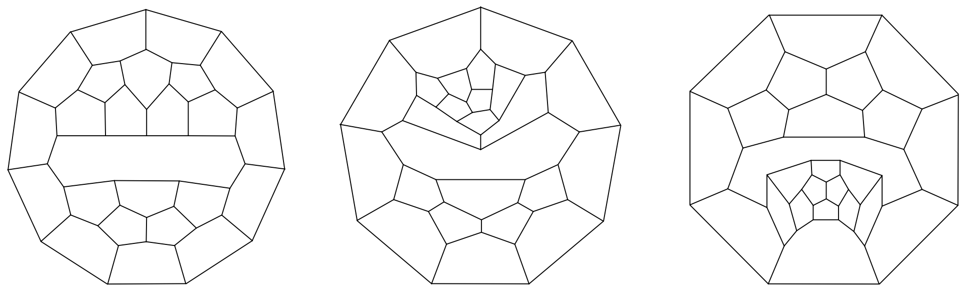


Fig. 9. *The other* 46*-vertex girth* 5 *nonhamiltonian C*4*CPs.*

TABLE 2
*Some numbers of graphs computed.*

| $n$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | $n_6$ | $n_7$ | $n_8$ |
|---|---|---|---|---|---|---|---|---|
| 38 | $4.53\times10^{11}$ | 6 | 0 | 0 | 192 | 0 | 187 | 0 |
| 40 | $3.58\times10^{12}$ | 37 | 191 | 0 | 651 | 0 | 627 | 0 |
| 42 | $2.86\times10^{13}$ | 274 | 5062 | 3 | 2070 | 0 | 1970 | 0 |
| 44 | $2.30\times10^{14}$ | – | – | – | 7290 | 1 | 6833 | 1 |
| 46 | $1.86\times10^{15}$ | – | – | – | 25381 | 3 | 23384 | 1 |
| 48 | $1.52\times10^{16}$ | – | – | – | – | – | 82625 | 0 |
| 50 | $1.25\times10^{17}$ | – | – | – | – | – | 292164 | 3 |
| 52 | $1.03\times10^{18}$ | – | – | – | – | – | – | 6 |

In the table above, the column headers are defined as follows:
$n$ = order of $G$,
$n_1 \simeq$ number of C3CPs,
$n_2$ = number of nonhamiltonian C3CPs with $\lambda'(G) = 3$ and girth $\geq 4$,
$n_3$ =number of nonhamiltonian C3CPs with girth 3,
$n_4$ = number of nonhamiltonian C4CPs,
$n_5$ = number of C3CPs with girth 5,
$n_6$ = number of nonhamiltonian C4CPs with $\lambda'(G) = 4$ and girth 5,
$n_7$ = number of C5CPs, and
$n_8$ = number of nonhamiltonian C5CPs.

We generated all girth 5 C3CPs up to 46 vertices and checked them for hamiltonicity. The only such graphs which were found to be nonhamiltonian and not cyclically 5-connected are the 44 and 46 vertex graphs in Figures 8 and 9. All of these graphs are cyclically 4-connected. Thus a smallest order of a nonhamiltonian C3CP of girth 5 with a nontrivial 3-cut is at least 48. We can easily construct a nonhamiltonian C3CP of girth 5 with a nontrivial 3-cut on 60 vertices using one of the 42 vertex graphs in Figure 2 and the dodecahedron.

Each of the nonhamiltonian C3CPs found in our searches has at least one face of size 8 or more. Barnette conjectures that all C3CPs with maximum face size at most 6 are hamiltonian. In particular, this conjecture covers the fullerenes, C3CPs with 12 faces of size 5, and all other faces of size 6. Using generation techniques of Brinkmann and McKay [5] we generated all C3CPs with maximum face size at most 6 (note that all such graphs on more than 20 vertices must have faces of size 6) up to 176 vertices and tested them for hamiltonicity. All were found to be hamiltonian. This supports the conjecture and extends the known hamiltonicity of fullerenes beyond 150 vertices (established in Brinkmann and Dress [3]).

**5. Some graph counts.** In Table 2 above we include some numbers of the graphs produced in our computations. The numbers in the column headed $n_1$ are the approximate total numbers of C3CPs of each of the indicated orders. We, of course, did not generate all such graphs but include them as an indication of the sparsity of nonhamiltonian C3CPs and of the effectiveness of the filters employed to make these computations feasible.

**Acknowledgment.** We wish to thank Gunnar Brinkmann for assisting with the computation.

## REFERENCES

[1] D. Barnette, *On generating planar graphs,* Discrete Math., 7 (1974), pp. 199–208.

[2] J. Bosák, *Hamiltonian lines in cubic graphs,* in Proceedings of International Seminar on Graph Theory and Applications, Rome, 1966, Gordon and Breach, New York, pp. 33–46.

[3] G. Brinkmann and A. Dress, *PentHex puzzles: A reliable and efficient top-down approach to fullerene-structure enumeration,* Adv. Appl. Math., 21 (1998), pp. 473–480.

[4] G. Brinkmann and A. Dress, *A constructive enumeration of fullerenes,* J. Algorithms, 23 (1997), pp. 345–358.

[5] G. Brinkmann and B. D. McKay, *Fast Generation of Some Classes of Planar Graphs,* in preparation.

[6] J. W. Butler, *A generation procedure for the simple* 3*-polytopes with cyclically* 5*-connected graphs,* Canad. J. Math., 26 (1974), pp. 686–708.

[7] G. B. Faulkner and D. H. Younger, *Non-hamiltonian simple planar maps,* Discrete Math., 7 (1974), pp. 67–74.

[8] B. Grünbaum, *Convex Polytopes,* John Wiley, New York, 1967.

[9] D. A. Holton and B. D. McKay, *Cycles in* 3*-connected cubic planar graphs.* II, Ars Combin., 21(A) (1986), pp. 107–114.

[10] D. A. Holton and B. D. McKay, *The smallest non-hamiltonian* 3*-connected cubic planar graphs have* 38 *vertices,* J. Combin. Theory Ser. B, 45 (1988), pp. 305–319; *Erratum,* 47 (1989), p. 248.

[11] B. D. McKay, *Nauty User's Guide* (*Version* 1.5)*,* Tech. report Tr-CS-90-02, Computer Science Dept., Australian National University, Canberra, Australia, 1990.

[12] B. D. McKay, *A Fast Search Algorithm for Hamiltonian Cycles in Cubic Graphs,* in preparation.

[13] B. Mohar, *Search for minimal non-hamiltonian simple* 3*-polytopes,* in Proceedings Fourth Yugoslav Seminar on Graph Theory, Novi Sad, University of Novi Sad, Institute of Mathematics, Novi Sad, 1983, pp. 191–208.

[14] H. Okamura, *Every simple* 3*-polytope of order* 32 *or less is hamiltonian,* J. Graph Theory, 6 (1982), pp. 185–196.

SIAM J. Discrete Math. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 1, pp. 33–47

# COMPLETELY DISCONNECTING THE COMPLETE GRAPH[*]

JOHN GINSBURG[†] AND BILL SANDS[‡]

**Abstract.** In this paper we consider procedures for completely disconnecting a complete graph on $n$ vertices $K_n$. By this we simply mean that we wish to remove all of the edges of $K_n$ in a sequence of steps by which the graph $K_n$ is successively transformed into the empty graph on $n$ vertices. The rules are that, on each step, we may remove at most one edge from each connected component of the present graph, and in addition we may impose a limit $w$ on the maximum number of edges we can remove at a time. What is the minimum number of steps $f_w(n)$ it will take to completely disconnect $K_n$? The two cases considered are $w = \infty$ and $w = 2$. We show that the ratio of $f_w(n)$ to the number of edges of $K_n$ is asymptotic to $2/3$ in the first case and to $1/2 + \lambda(1 - \lambda) = .74194412\ldots$ in the second, where $\lambda$ is the real root of the equation $\lambda = 2(1 - \lambda)^3$.

**Key words.** complete graph, edge deletion

**AMS subject classification.** 05C85

**PII.** S0895480197326553

## 1. Introduction.

In a recent work [2] the authors investigated the "celery cutting" problem: to describe an optimal procedure with which to cut any given collection of sticks of various integer lengths completely into pieces of unit length, using a knife that can cut as many as $w$ sticks at a time, where $w$ is a given integer. As shown in [2], an optimal procedure results when one repeatedly cuts as many sticks as possible, always cutting the longest available sticks first and always cutting the sticks in half, or as nearly in half as possible.

There are various natural generalizations of this type of cutting problem. We can think of a "celery stick" as being a *path*, whose vertices correspond to the units to be cut apart, and whose edges correspond to the connections to be cut between the units. Making a cut to a celery stick then corresponds to deleting one edge from a path, changing it into a disjoint union of two smaller paths. In this way, a collection of sticks corresponds to a graph $G$ which is a disjoint union of paths, and cutting up all the sticks into units is equivalent to deleting all of the edges of the graph $G$ in a sequence of steps. On each step we are allowed to remove or "cut" as many as $w$ edges from the present graph, but we can delete no more than one edge from any of the connected components of the present graph. This last condition must be included since on any step we can make only one cut to any particular stick which is then present. If we phrase the celery cutting problem in these graph-theoretic terms, it has a natural and interesting generalization to all graphs $G$, and one can seek, for a given graph or class of graphs, an optimal procedure by which to cut or delete all of the edges of the graph in this way. We will speak of this as *completely disconnecting* the graph. In this paper we will consider how this problem extends to the class of complete graphs. We will discuss two instances of the problem, $w = 2$ (we can cut or

[*]Received by the editors August 27, 1997; accepted for publication (in revised form) August 9, 1999; published electronically January 13, 2000. The majority of the research of this paper was done while the second author was visiting the University of Winnipeg.

http://www.siam.org/journals/sidma/13-1/32655.html

[†]Department of Mathematics and Statistics, University of Winnipeg, Winnipeg, MB, Canada R3B 2E9 (ginsburg-j@l-h.uwinnipeg.ca).

[‡]Department of Mathematics and Statistics, University of Calgary, Calgary, AB, Canada T2N 1N4 (sands@math.ucalgary.ca).

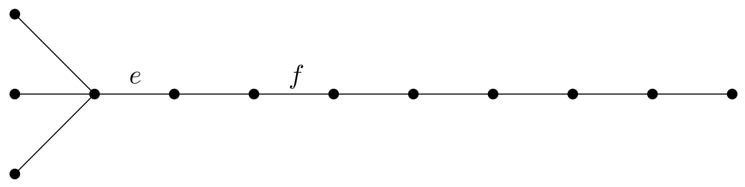


FIG. 1.

delete at most two edges at a time) and $w = \infty$ (we can delete any number of edges at a time).

Probably the most natural direction in which to extend the celery cutting problem is to consider *trees* rather than paths. While we will not investigate this topic here, it is worth noting that the obvious analog of the optimal algorithm described above for paths will *not*, in general, be optimal for trees. The analog to cutting a path in half would be to delete an edge from a tree so that the resulting two connected components have sizes which are as nearly equal as possible. If we are allowed to delete as many as $w$ edges at a time, we would then repeatedly apply this to the $w$ largest trees present. However, this will *not* always delete all of the edges of the tree in the minimum number of steps. An example to show this is the “rake” in Figure 1. Suppose we take $w$ to be 5. On the first step we can only delete one edge. We want to cut this tree into two equal components, and so we delete the edge labeled $f$ in Figure 1. This cuts the tree up into two smaller trees. For each of these trees we find an edge which will split the tree up into two equal or nearly equal pieces, and we delete these two edges together. We then do the same thing for all of the four resulting trees, continuing in this manner until all edges have been removed. We see that five steps will be required by this procedure. However, this procedure is not optimal. In fact, if one starts by deleting the edge labeled $e$ (which does *not* cut the tree into equal-sized parts), then the rest of the edges can be removed using just three more steps as we invite the reader to show.

One issue in efficiently cutting up a graph is whether to strive to “bisect” the graph, that is, to delete edges so that the graph is divided up into two components of equal size (or as nearly equal as possible). However, as we have just seen, even for trees, successively bisecting may not lead to an optimal procedure. Another issue concerns the size and nature of the set of edges to be deleted in splitting a connected graph into components. Should we attempt to break the graph into two components as quickly as possible (by successively deleting as few edges as we can in order to disconnect the graph)? Or should we take more steps to split the graph up (by successively deleting the edges of a larger sized cutset) and obtain more equal-sized components? In any tree, just as in paths, every single edge is a cutset, and deleting *any* one edge divides the graph into two components, so the size of the cutset is not an issue here. However for other types of graphs, this must obviously be an important consideration.

For complete graphs, which are the focus of this paper, the size of these cutsets becomes the main concern. The complete graph on $n$ vertices as usual is denoted by $K_n$. For any two nonempty disjoint sets of vertices $A$ and $B$ whose union is the whole set of vertices, the set $(A, B)$ consisting of all edges joining a vertex of $A$ to a vertex of $B$ is a cutset of $K_n$, and so for every integer $k$ with $1 \leq k \leq n-1$, $K_n$ has a cutset of size $k(n-k)$. The smallest of these cutsets occurs for $k = 1$ and the largest for $k = \lfloor n/2 \rfloor$.

In our study of complete graphs we will see that, unlike the case for paths or celery sticks, the width of the knife $w$ (that is, the limit on how many edges we are allowed to remove at a time) also plays an important role.

We denote the set of vertices and set of edges of a graph $G$, respectively, by $V(G)$ and $E(G)$. If $G_1$ and $G_2$ are disjoint graphs, we denote their disjoint union by $G_1 \oplus G_2$—this denotes the graph whose vertex set is $V(G_1) \cup V(G_2)$ and whose edge set is $E(G_1) \cup E(G_2)$. We refer the reader to [1] for other basic concepts and terminology regarding graphs.

To end this section we give a more precise description of what is meant by a procedure which successively removes all of the edges of a graph. Let $w$ be a given positive integer. A *procedure for completely disconnecting a graph* $G$ is a sequence $G_0, G_1, \ldots, G_t$ of spanning subgraphs of $G$, where $t$ is a nonnegative integer and $G_0 = G$, $G_t$ is the empty graph on the vertex set $V(G)$, and such that, for all $i = 0, 1, \ldots, t-1$, $G_{i+1}$ is a graph obtained from $G_i$ by deleting at most $w$ edges from $G_i$ with no more than one edge being deleted from any connected component of $G_i$. For such a procedure, we will say that the integer $t$ is the *number of steps* used by the procedure. A procedure for completely disconnecting $G$ which uses the smallest number of steps is called an *optimal procedure* for $G$.

**2. The optimal procedure for completely disconnecting $K_n$ when $w = \infty$.** In our first theorem we actually assume no upper bound on the number of edges that can be removed on each step, beyond the condition that the edges removed must be in separate components. We abbreviate this situation by writing $w = \infty$. It is clear that this is equivalent to having $w \geq n/2$, where $n$ is the number of vertices of the graph.

We will let $f_\infty(G)$ denote the least number of steps required to completely disconnect a graph $G$ when $w = \infty$. When $G = K_n$, we will write $f_\infty(n)$ instead of $f_\infty(K_n)$.

Imagine an optimal procedure for disconnecting $K_n$. In disconnecting $K_n$ we can remove only one edge at a time until $K_n$ has been separated into two components $C_1$ and $C_2$. These components will be subgraphs of $K_a$ and $K_b$, where $a$ and $b$ are positive integers such that $a + b = n$, corresponding to disjoint sets of vertices $A$ and $B$ of sizes $a$ and $b$, respectively. In getting to this point we must, of course, have deleted the cutset consisting of all of the $ab$ edges connecting $A$ and $B$ (with one step required for each edge). In fact, we would not have yet needed to delete any other edges, that is, the edges joining two vertices of $A$ or joining two vertices of $B$, in getting to $C_1$ and $C_2$, because so far all edges can be deleted only one at a time and therefore can be done in any order. Therefore, in considering an optimal procedure for completely disconnecting $K_n$, we can assume that we first reduce $K_n$ to two disjoint complete subgraphs $K_a$ and $K_{n-a}$, where (say) $n/2 \leq a \leq n-1$, and that no other edges of $K_n$ are removed until this much has been accomplished. This takes $a(n-a)$ moves. Since it is also clear that $f_\infty(x) \geq f_\infty(y)$ whenever $x \geq y$, we will be able to finish the remaining graph $K_a \oplus K_{n-a}$ in $f_\infty(a)$ moves by reducing both components simultaneously. Our first theorem says that the best way to do all this is to separate $K_n$ into *equal-sized* components, even though this takes the *largest* number of initial moves.

THEOREM 2.1.

$$f_\infty(n) = \left\lceil \frac{n}{2} \right\rceil \left\lfloor \frac{n}{2} \right\rfloor + f_\infty\left(\left\lceil \frac{n}{2} \right\rceil\right).$$

*Proof.* By the above discussion, we need only show that for all integers $a$ satisfying $n/2 \le a < n$,

$$\lceil n/2 \rceil \lfloor n/2 \rfloor + f_\infty(\lceil n/2 \rceil) \le a(n-a) + f_\infty(a).$$

This is true for $n \le 3$ by inspection. Proceeding by induction, consider some integer $n > 3$ and an integer $a$ such that $n/2 \le a < n$, and suppose that the result holds for all integers smaller than $n$. In particular, then,

$$f_\infty(a) = \lceil a/2 \rceil \lfloor a/2 \rfloor + f_\infty(\lceil a/2 \rceil),$$

and by the definition of $f_\infty$,

$$f_\infty(\lceil n/2 \rceil) \le \lceil a/2 \rceil (\lceil n/2 \rceil - \lceil a/2 \rceil) + f_\infty(\lceil a/2 \rceil).$$

So it is enough to prove that

$$\lceil n/2 \rceil \lfloor n/2 \rfloor + \lceil a/2 \rceil (\lceil n/2 \rceil - \lceil a/2 \rceil) \le a(n-a) + \lceil a/2 \rceil \lfloor a/2 \rfloor. \tag{1}$$

But since

$$\begin{aligned} a^2 - \lceil a/2 \rceil^2 - \lceil a/2 \rceil \lfloor a/2 \rfloor &= (a - \lceil a/2 \rceil)(a + \lceil a/2 \rceil) - \lceil a/2 \rceil \lfloor a/2 \rfloor \\ &= \lfloor a/2 \rfloor (a + \lceil a/2 \rceil) - \lceil a/2 \rceil \lfloor a/2 \rfloor = a \lfloor a/2 \rfloor, \end{aligned}$$

(1) can be rewritten successively as

$$a\lfloor a/2 \rfloor - an + \lceil a/2 \rceil \lceil n/2 \rceil + \lceil n/2 \rceil \lfloor n/2 \rfloor \le 0,$$

$$a\lfloor a/2 \rfloor - a(n - \lceil n/2 \rceil) - \lceil n/2 \rceil (a - \lceil a/2 \rceil) + \lceil n/2 \rceil \lfloor n/2 \rfloor \le 0,$$

$$a\lfloor a/2 \rfloor - a\lfloor n/2 \rfloor - \lceil n/2 \rceil \lfloor a/2 \rfloor + \lceil n/2 \rceil \lfloor n/2 \rfloor \le 0,$$

and finally

$$(a - \lceil n/2 \rceil)(\lfloor a/2 \rfloor - \lfloor n/2 \rfloor) \le 0,$$

which is true, since $\lfloor a/2 \rfloor \le \lfloor n/2 \rfloor \le \lceil n/2 \rceil \le a$. □

COROLLARY 2.2. *For all integers $n \ge 2$,*

$$\frac{n^2 - 1}{3} \le f_\infty(n) \le \frac{n^2 + n - 3}{3}$$

*where both bounds are the best ones possible.*

*Proof.* First note that $f_\infty(2) = 1$ and $f_\infty(3) = 3$, so that the inequalities hold when $n = 2$ and 3. Now consider some $n > 3$, and assume the result holds for all positive integers less than $n$. Then by the theorem,

$$\begin{aligned}
f_\infty(n) &= \lceil n/2\rceil\lfloor n/2\rfloor + f_\infty(\lceil n/2\rceil)\\
&\geq \lceil n/2\rceil\lfloor n/2\rfloor + \frac{\lceil n/2\rceil^2 - 1}{3}\\
&= \frac{3\lceil n/2\rceil\lfloor n/2\rfloor + n^2 - (n-\lceil n/2\rceil)(n+\lceil n/2\rceil) - 1}{3}\\
&= \frac{n^2 - 1 + 3\lceil n/2\rceil\lfloor n/2\rfloor - \lfloor n/2\rfloor(n+\lceil n/2\rceil)}{3}\\
&= \frac{n^2 - 1 + \lfloor n/2\rfloor(2\lceil n/2\rceil - n)}{3}\\
&\geq \frac{n^2-1}{3}\ .
\end{aligned}$$

Equality holds if and only if $n$ is even *and* equality holds for the integer $\lceil n/2\rceil$, from which it is easy to see that $f_\infty(n) = (n^2-1)/3$ if and only if $n$ is a power of 2.

Similarly we can prove the right-hand inequality:

$$\begin{aligned}
f_\infty(n) &= \lceil n/2\rceil\lfloor n/2\rfloor + f_\infty(\lceil n/2\rceil)\\
&\leq \lceil n/2\rceil\lfloor n/2\rfloor + \frac{\lceil n/2\rceil^2 + \lceil n/2\rceil - 3}{3}\\
&= \frac{3\lceil n/2\rceil\lfloor n/2\rfloor + n^2 - (n-\lceil n/2\rceil)(n+\lceil n/2\rceil) + (n-\lfloor n/2\rfloor) - 3}{3}\\
&= \frac{n^2 + n - 3 + 3\lceil n/2\rceil\lfloor n/2\rfloor - \lfloor n/2\rfloor(n+\lceil n/2\rceil) - \lfloor n/2\rfloor}{3}\\
&= \frac{n^2 + n - 3 - \lfloor n/2\rfloor(n + 1 - 2\lceil n/2\rceil)}{3}\\
&\leq \frac{n^2+n-3}{3}\ .
\end{aligned}$$

Equality holds if and only if $n$ is odd *and* equality holds for $\lceil n/2\rceil$. One can check that this happens exactly when $n = 2^k + 1$ for some integer $k$. □

We can observe other nice special cases, for example, $f_\infty(n) = n^2/3$ if and only if $n = 3\cdot 2^k$ for some integer $k$.

It is interesting to compare the number of steps necessary to disconnect all of the edges of $K_n$ to the number of edges of $K_n$. As the preceding inequalities show, the ratio of these two numbers is asymptotic to 2/3.

Corollary 2.3. *$f_\infty(n) \sim \frac{2}{3}\binom{n}{2}$ as $n \to \infty$.*

**3. Removing at most two edges at a time: The case $w = 2$.** In this section we restrict our attention to the case when $w = 2$. In this case, a procedure for completely disconnecting a graph $G$ is a sequence $G_0, G_1, \ldots, G_t$ of graphs, where $t$ is a nonnegative integer and $G_0 = G$, $G_t$ is the empty graph, and such that, for all $i = 0, 1, \ldots, t-1$, $G_{i+1}$ is a graph obtained from $G_i$ by deleting at most two edges from $G_i$, with no more than one edge being deleted from any connected component of $G_i$.

For $i = 0, 1, \ldots, t-1$, the graph $G_{i+1}$ is obtained by deleting either one or two edges from $G_i$. The former will be referred to as a *unit deletion* and as a *unit step* of the procedure. The latter steps are referred to as *doublet deletions.* In the case when

$w = 2$, we will let $f_2(G)$ denote the number of steps used in an optimal procedure for completely disconnecting $G$.

Suppose that a procedure for completely disconnecting a graph $G$ takes $t$ steps, of which $t_1$ are unit steps and $t_2$ are doublet steps. Since every edge of $G$ is deleted at some step, we have $t_1 + 2t_2 = |E(G)|$. Therefore $t = t_1 + t_2 = (|E(G)| + t_1)/2$. In particular, every optimal procedure for $G$ uses the same number of unit steps. We will denote the number of unit steps in any optimal procedure for $G$ by $u_2(G)$. We thus have the following.

LEMMA 3.1. *Let $G$ be a graph. Then $f_2(G) = (|E(G)| + u_2(G))/2$.*

We need two more simple lemmas before discussing $K_n$ in earnest.

LEMMA 3.2. *Let $G$ and $H$ be disjoint graphs, and suppose that $|E(H)| \le |E(G)|$. Suppose there is a procedure for completely disconnecting $G$ which uses at most one unit deletion. Then there is a procedure for completely disconnecting $G \oplus H$ which uses at most one unit deletion.*

*Proof.* We use induction on $k = |E(H)|$. If $k = 0$ there is nothing to prove. Suppose $k = 1$, and let $e$ be the single edge of $H$. There is a procedure for completely disconnecting $G$ which uses at most one unit deletion. If no unit deletions are used, then delete the single edge $e$ at the end of that procedure to give a procedure for $G \oplus H$ which uses at most one unit deletion. If there is one unit deletion involved at some step of the procedure for $G$, then we can delete the edge $e$ together with the single edge from $G$ involved in this step. Carrying out the procedure for $G$ with the modification of this one step then gives an appropriate procedure for $G \oplus H$.

Thus we suppose that $k > 1$ and that the statement is true for all graphs $G \oplus H$, where $|E(H)| < k$. Now consider a graph $G \oplus H$ when $|E(H)| = k$. There is a procedure for $G$ in which at most one unit deletion is involved. In the first step of this procedure a set of at most two edges $\{e_1, e_2\}$ is deleted from $G$ where we allow the possibility that $e_1 = e_2$. Clearly the graph $G_1 = (V(G), E(G) - \{e_1, e_2\})$ also can be completely reduced by a procedure which uses at most one unit deletion. Now, choose edges $f_1$ and $f_2$ of $H$ so that $f_1$ and $f_2$ are distinct if $e_1$ and $e_2$ are distinct, and are identical if $e_1$ and $e_2$ are. In case the two edges are distinct, we can begin a procedure for disconnecting $G \oplus H$ by deleting $e_1$ and $f_1$ together on the first step, and then $e_2$ and $f_2$ together on the second step. If the two edges are identical we begin a procedure for $G \oplus H$ by deleting $e_1$ and $f_1$ together on the first step. Either way, we will have reduced (in one or two doublet steps) $G \oplus H$ to a graph $G_1 \oplus H_1$ where $G_1$ is a graph which can be completely reduced by a procedure which uses at most one unit deletion, and where $H_1$ is a graph with fewer than $k$ edges. We can then apply the induction hypothesis to $G_1 \oplus H_1$ and obtain an appropriate procedure for $G \oplus H$. ☐

LEMMA 3.3. *Let $r$ and $s$ be integers with $6 \le r \le s \le r + 2$. Then $u_2(K_r \oplus K_s) \le 1$.*

*Proof.* This is trivial if $r = s$—we just pair off the edges of the two equal-sized components $K_r$ in any order whatsoever and then remove them two by two in this order; no unit deletions are required.

Consider the second possibility, that $s = r + 1$. We note that there is a positive integer $p$ such that $2p \le r + 1$ and $\binom{r+1}{2} - \binom{r}{2} = r \le 2\binom{p}{2}$. In fact, it is easy to check that, for $r \ge 6$, $p = \lfloor \frac{r+1}{2} \rfloor$ satisfies these two conditions. Now consider the copy of $K_{r+1}$ in the graph $K_r \oplus K_{r+1}$. Let $A_1$ and $A_2$ be disjoint sets of vertices in $K_{r+1}$, with $|A_1| = |A_2| = p$. Let $E_1$ and $E_2$, respectively, denote the set of edges of the subgraphs induced by $A_1$ and $A_2$ (copies of $K_p$), and let $E$ denote the set of all edges of $K_{r+1}$. We have that

$$|E-(E_1 \cup E_2)| = \binom{r+1}{2} - 2\binom{p}{2} \le \binom{r}{2}.$$

Let $m = |E-(E_1 \cup E_2)|$. Now, pair the $m$ edges of $E-(E_1 \cup E_2)$ with $m$ of the edges of the copy of $K_r$ in $K_r \oplus K_{r+1}$. Delete these pairs of edges one after the other. This involves no unit deletions and reduces $K_r \oplus K_{r+1}$ to the graph $G = (K_p \oplus K_p) \oplus H$ where $H$ is a subgraph of $K_r$ such that $|E(H)| \le |E(K_p \oplus K_p)|$. Since the graph $K_p \oplus K_p$ can obviously be completely reduced with no unit deletions, it follows from Lemma 3.2 that we can continue and completely reduce $G$ using at most one unit deletion.

The final case is when $s = r+2$. This we handle similarly to the second case. One can handle the case when $r=6$ or $r=7$ directly and without difficulty. For $r \ge 8$ we proceed as follows. There is a positive integer $p$ such that $2p \le r+2$ and $\binom{r+2}{2} - \binom{r}{2} \le 2\binom{p}{2}$. One checks that $p = \lfloor \frac{r+2}{2} \rfloor$ satisfies these two conditions. As in the above argument, we then pair off all the edges of $K_{r+2}$ which are not edges of a copy of $K_p \oplus K_p$ with edges of $K_r$. We then delete these pairs, and the remainder of the argument is identical. $\square$

For the complete graph $G = K_n$, we will write $f_2(n)$ and $u_2(n)$ instead of $f_2(K_n)$ and $u_2(K_n)$. As discussed in the previous section, in considering an optimal procedure for completely disconnecting $K_n$, we can assume that we first reduce $K_n$ to two disjoint complete subgraphs $K_k$ and $K_{n-k}$, and that no other edges of $K_n$ are removed until this much has been accomplished. This requires $k(n-k)$ unit moves, after which the procedure would continue to optimally disconnect the graph $K_k \oplus K_{n-k}$. Our main concern here is what ratio of $k$ to $n$ is involved in the beginning part of an optimal procedure. Unlike the case $w = \infty$, we will find that the "starting ratio" should *not* be 1/2. Our results instead point to the unique (irrational) real root $\lambda$ of the polynomial equation $\lambda = 2(1-\lambda)^3$. We note here that $\lambda = .41024548\ldots$. We will also make use of the quantity

$$\epsilon = \frac{4\lambda^2 - 5\lambda + 2}{\lambda(2\lambda^2 - 6\lambda + 5)},$$

for which we note that $\epsilon = .5273194\ldots$.

But first let's see where this number $\lambda$ comes from. Our idea of how to disconnect $K_n$ is to separate it into two pieces whose size ratio is such that the smaller piece's edges can be used exactly to break the larger piece into this same size ratio, thereby making a recursion possible. Thus, letting $k = \lambda n$ be the number of vertices in the smaller piece, this piece will contain $\lambda n(\lambda n - 1)/2$ edges, and we would like these edges to pair off exactly with the edges that must be removed to break $K_{n-\lambda n}$ into pieces of sizes $\lambda(n-\lambda n)$ and $(1-\lambda)(n-\lambda n)$. This results in the equation

$$\frac{\lambda n(\lambda n - 1)}{2} = \lambda(1-\lambda)n \cdot (1-\lambda)^2 n.$$

Equating coefficients of $n^2$ (and ignoring the smaller-order term) gives us the above equation for $\lambda$.

So much for the intuitive reason for our consideration of $\lambda$; now for the details of the proof.

THEOREM 3.4. *Let $n \ge 6$ and $k$ be positive integers such that $\lambda n + \epsilon \le k \le n/2$, where $\lambda$ and $\epsilon$ are defined as above. Then there is a procedure for completely* *disconnecting $K_n$, which begins by using $k(n-k)$ unit deletions to reduce $K_n$ to $K_k \oplus K_{n-k}$, and then which uses at most one more unit deletion in reducing $K_k \oplus K_{n-k}$ to the empty graph.*

*Proof.* Our proof is by induction on $n$. We can verify the conclusion directly for $6 \leq n \leq 11$ without difficulty. In fact, for $n = 7$ and $n = 9$ the result actually holds vacuously, since no integer $k$ exists in the stated interval. The other four cases can all be done in a similar way. We will show the arguments for $n = 8$ and $n = 11$ as being typical. For $n = 8$, the inequality for $k$ implies that $3.75 \leq k \leq 4$ and so we must have $k = 4$. In this case, after using $(4)(4) = 16$ unit deletions to reduce $K_8$ to $K_4 \oplus K_4$, we can clearly complete the job by reducing the two equal-sized components two edges at a time, one from each. No more unit deletions are involved. For the case $n = 11$, the inequality for $k$ implies that $4.98 \leq k \leq 5.5$ and so we must have $k = 5$. In this case, we use $(6)(5) = 30$ unit deletions to reduce $K_{11}$ to $K_6 \oplus K_5$. We then carry out nine doublet deletions, in which we remove one edge from each of the two graphs $K_6$ and $K_5$. We do this in such a way that the nine edges deleted from $K_6$ disconnect $K_6$ into $K_3 \oplus K_3$, while the nine edges these are paired with from $K_5$ can be chosen in any fashion from the $\binom{5}{2} = 10$ edges of $K_5$. After this we will still have one edge remaining from the $K_5$, and we will also have the graph $K_3 \oplus K_3$. We next disconnect the latter graph in three doublet steps, by pairing the edges from the two copies of $K_3$. This leaves the one edge from $K_5$, which we do in a final unit step. Thus only one unit step is required after the initial reduction from $K_{11}$ to $K_6 \oplus K_5$.

Assume that $n \geq 12$ and that the result holds for all positive integers less than $n$. Let $k$ be any integer satisfying the hypothesis, and put $\alpha = k/n$. Our conditions on $k$ imply that $\lambda < \alpha \leq 1/2$. Now, consider the integers $n' = n - k$ and $k' = k - \lfloor \lambda k \rfloor$. Note that $n' \geq n/2 \geq 6$. We claim that $\lambda n' + \epsilon \leq k'$. This holds since

$$k' = k - \lfloor \lambda k \rfloor \geq k - \lambda k \geq (\lambda n + \epsilon) - \lambda k = \lambda(n-k) + \epsilon = \lambda n' + \epsilon.$$

In order to apply the induction hypothesis to the integer $n'$ and the integer $k'$ we also need to know that $k' \leq n'/2$, that is,

$$k - \lfloor \lambda k \rfloor \leq \frac{1}{2}(n-k).$$

We will assume this for the moment, and we will return later to modify the argument in case this last condition fails. Thus, we apply the induction hypothesis to $n'$ and $k'$: there is a procedure for completely disconnecting $K_{n-k}$ which uses $(k - \lfloor \lambda k \rfloor)[(n-k) - (k - \lfloor \lambda k \rfloor)]$ unit deletions to disconnect $K_{n-k}$ into $H = K_{(n-k)-(k-\lfloor \lambda k \rfloor)} \oplus K_{k-\lfloor \lambda k \rfloor}$ and then continues to completely disconnect $H$ with at most one more unit deletion being employed.

We next show that $\binom{k}{2} \geq k'(n'-k')$. This is equivalent to saying that

$$k(k-1) \geq 2(n - 2k + \lfloor \lambda k \rfloor)(k - \lfloor \lambda k \rfloor).$$

Put $\lfloor \lambda k \rfloor = \lambda k - x$ where $0 \leq x < 1$, replace $k$ by $\alpha n$, and expand; then the desired inequality is equivalent to

$$(2)\quad (5\alpha^2 - 2\alpha - 6\lambda\alpha^2 + 2\lambda\alpha + 2\lambda^2\alpha^2)n^2 - (\alpha - 6x\alpha + 4x\lambda\alpha + 2x)n + 2x^2 \geq 0.$$

Using the fact that $\lambda = 2(1-\lambda)^3$ we see that the coefficient of $n^2$ in this last inequality is equal to

$$\begin{aligned}\alpha(5\alpha-2-6\lambda\alpha+2\lambda+2\lambda^2\alpha) &= \alpha[(5-6\lambda+2\lambda^2)(\alpha-\lambda)+(2\lambda^3-6\lambda^2+7\lambda-2)] \\ &= \alpha[(5-6\lambda+2\lambda^2)(\alpha-\lambda)-2(1-\lambda)^3+\lambda] \\ &= \alpha(\alpha-\lambda)(2\lambda^2-6\lambda+5).\end{aligned}$$

Since we have that

$$(\alpha-\lambda)n = k-\lambda n \ \geq\ \epsilon = \frac{4\lambda^2-5\lambda+2}{\lambda(2\lambda^2-6\lambda+5)}\ ,$$

and since $2\lambda^2-6\lambda+5 \geq 0$, the inequality (2) will follow from

$$\alpha(2\lambda^2-6\lambda+5)\epsilon n \geq (\alpha-6x\alpha+4x\lambda\alpha+2x)n.$$

This is equivalent to

$$\alpha(4\lambda^2-5\lambda+2) \geq \lambda(\alpha+2x(-3\alpha+2\lambda\alpha+1)),$$

which simplifies to

$$\alpha(2\lambda^2-3\lambda+1) \geq \lambda x(2\lambda\alpha-3\alpha+1). \tag{3}$$

Now if $2\lambda\alpha-3\alpha+1 \leq 0$, (3) will follow from the true inequality $2\lambda^2-3\lambda+1>0$. Conversely, if $2\lambda\alpha-3\alpha+1>0$, then because $0 \leq x < 1$, (3) will follow from

$$\alpha(2\lambda^2-3\lambda+1) \geq \lambda(2\lambda\alpha-3\alpha+1).$$

This last inequality is equivalent to $\alpha \geq \lambda$, and is therefore true.

We have therefore shown that $\binom{k}{2} \geq k'(n'-k')$. This means that the number of edges of the graph $K_k$ is at least as large as the number $m = k'(n'-k')$—the size of the cutset $C$ which must be deleted from $K_{n'}$ to reduce it to the graph $H = K_{n'-k'} \oplus K_{k'}$. Now, consider the following procedure for completely disconnecting $K_n$. First use $k(n-k)$ unit steps to reduce $K_n$ to $K_{n-k} \oplus K_k$. Then carry out $m$ doublet deletions, in each of which we delete one of the edges of $C$ together with one of the edges of the graph $K_k$. At the end of these $m$ steps, all of the edges of $C$ have been deleted, and $K_{n'}$ has been reduced to $H = K_{n'-k'} \oplus K_{k'}$, while $m$ edges of the graph $K_k$ have been deleted from it, leaving some spanning subgraph $L$ of $K_k$. The induction hypothesis implies that a procedure exists to completely reduce $H$ with at most one more unit deletion. The fact that $n-k \geq k$ implies that the number of edges in the graph $L$ (that is, the number of edges of $K_k$ minus $m$) is no larger than the number of edges of the graph $H$ (this being the number of edges of $K_{n-k}$ minus $m$). Therefore by Lemma 3.2, we can continue and completely reduce $H \oplus L$ using at most one more unit deletion. Therefore the desired statement is shown to be true for $n$.

As mentioned in the preceding argument, in order to apply the induction hypothesis, we needed to know that $k' \leq n'/2$, that is, $k-\lfloor\lambda k\rfloor \leq \frac{1}{2}(n-k)$. This condition may fail. In this case we would certainly have that $n'-k' \leq n'/2$. We then can reverse the roles of $k'$ and $n'-k'$. Let $k_1 = n'-k'$. Thus $k_1 = (n-k)-(k-\lfloor\lambda k\rfloor)$. We have that $k_1 \leq n'/2$, and we also need that $k_1 \geq \lambda n'+\epsilon$. This latter statement is equivalent to $(n-k)-(k-\lfloor\lambda k\rfloor) \geq \lambda(n-k)+\epsilon$. This would follow from the inequality $(n-k)-(k-(\lambda k-1)) \geq \lambda(n-k)+\epsilon$, which is equivalent to $(1-\lambda)n \geq 2(1-\lambda)k+1+\epsilon$; that is,

$$\frac{n}{2} \geq k+\frac{1+\epsilon}{2(1-\lambda)} = k+1.2948773\ldots, \tag{4}$$

If this last condition holds, then we can instead apply the inductive hypothesis to the integer $n' = n - k$ in place of $n$ and apply $k_1$ in place of $k$. As the argument above shows, the number of edges of the graph $K_k$ is at least as big as the number of edges in a cutset of size $k_1(n' - k_1) = k'(n' - k')$ for the graph $H = K_{n'-k_1} \oplus K_{k_1}$, and so the identical argument applies. Thus the only circumstance we have left to consider is when condition (4) fails. Since by assumption we do have that $n/2 \geq k$, this can happen only if either $n = 2k$, $n = 2k + 1$, or $n = 2k + 2$. In all three of these cases, the statement we want to prove follows from Lemma 3.3, since $k \geq 6$. □

We have the following immediate corollary of Theorem 3.4.

COROLLARY 3.5. *Let $n \geq 6$. Then*

$$u_2(n) \leq \lceil \lambda n + \epsilon \rceil (n - \lceil \lambda n + \epsilon \rceil) + 1.$$

*Proof.* When $k = 7$ or $9$, this says that $u_2(7) \leq 4 \cdot 3 + 1 = 13$ and $u_2(9) \leq 5 \cdot 4 + 1$ $=21$, respectively, both of which can be verified directly. For any other integer $n \geq 6$, $\lceil \lambda n + \epsilon \rceil$ satisfies the conditions for $k$ in Theorem 3.4, and the statement follows. □

Note that Corollary 3.5 implies that, if $n$ is sufficiently large and $w = 2$, then no optimal procedure for completely disconnecting $K_n$ can result by first proceeding to divide $K_n$ "in half," that is, into two components of size $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$. This entails at least $\lceil n/2 \rceil \lfloor n/2 \rfloor$ unit deletions, a number which is roughly $.25n^2$. The right-hand side of Corollary 3.5 is of order $\lambda(1 - \lambda)n^2 = (.24194412\ldots)n^2$.

Theorem 3.4 provides us with an upper bound on the size of an initial cutset with which to completely disconnect a complete graph efficiently. After another simple preliminary lemma, we will consider a lower bound.

LEMMA 3.6. *Let $G$ and $H$ be disjoint graphs. Then*

$$u_2(G \oplus H) \ \geq \ u_2(G) - |E(H)|.$$

*Proof.* Consider an optimal procedure for completely disconnecting the graph $G \oplus H$, requiring $m = u_2(G \oplus H)$ unit deletions. Each step of the procedure is one of three types: one or two edges of $G$ are deleted, or one or two edges of $H$ are deleted, or one edge $e_G$ from $G$ is deleted together with one edge $e_H$ from $H$. Note that there can be no more than $|E(H)|$ steps of the third kind. Now, suppose we parallel the procedure for $G \oplus H$ on $G$: whenever the first type of deletion step occurs we apply this same deletion step to $G$; whenever the second type of deletion (edges of $H$ only) occurs we do nothing on $G$; whenever the third type of deletion step occurs, instead of deleting the two edges $\{e_G, e_H\}$ from $G \oplus H$, we delete the single edge $e_G$ from $G$. This parallel procedure then completely reduces $G$ and there are no more than $|E(H)|$ additional unit deletion steps in this procedure than there were unit steps in the original procedure. Thus we have a procedure for $G$ with at most $m + |E(H)|$ unit steps. We therefore have $u_2(G) \leq m + |E(H)|$. □

THEOREM 3.7. *Let $n \geq 2$. Then $u_2(n) \geq \lambda n(n - \lambda n) = \lambda(1 - \lambda)n^2$.*

*Proof.* For $n \leq 4$, the inequality follows from the easily calculated values $u_2(2) =$ $1$, $u_2(3) = 3$, $u_2(4) = 4$. So we assume that $n > 4$ and that the inequality holds for all integers which are $\geq 2$ and smaller than $n$. Now, consider any procedure for completely disconnecting $K_n$. We must show that the number of unit deletions used is at least $\lambda(1 - \lambda)n^2$. The procedure must begin by using $k(n - k)$ unit steps in reducing $K_n$ to $K_k \oplus K_{n-k}$, where $k$ is some positive integer with $1 \leq k \leq n/2$. If $k \geq \lambda n$, then $k(n - k) \geq \lambda n(n - \lambda n)$, and so the desired inequality holds. So we may as well assume that $k < \lambda n$. By Lemma 3.6, the rest of the procedure, which completely disconnects $K_k \oplus K_{n-k}$, must use at least $u_2(n-k) - \binom{k}{2}$ unit deletions. Now, since $2 \le n-k < n$, the induction hypothesis applies to the integer $n-k$, and we have $u_2(n-k) \ge \lambda(1-\lambda)(n-k)^2$. Therefore, no matter how the procedure continues, the total number of unit deletions needed will be at least

$$k(n-k) + u_2(n-k) - \binom{k}{2} \ \ge\ k(n-k) + \lambda(1-\lambda)(n-k)^2 - \binom{k}{2}.$$

We see that what we wish to prove will follow from the inequality

$$k(n-k) + \lambda(1-\lambda)(n-k)^2 - \binom{k}{2} \ \ge\ \lambda(1-\lambda)n^2.$$

This is equivalent to

$$2k(n-k) - k(k-1) \ge 2\lambda(1-\lambda)(n^2 - (n-k)^2) = 2\lambda(1-\lambda)(2nk - k^2).$$

Dividing through by $k$, we see that what we want is

$$[2 - 4\lambda(1-\lambda)]\, n \ge [3 - 2\lambda(1-\lambda)]\, k - 1.$$

Since $k < \lambda n$ and $3 - 2\lambda(1-\lambda) > 0$, the desired inequality will follow from

$$(2 - 4\lambda + 4\lambda^2)n \ge (3 - 2\lambda + 2\lambda^2)\lambda n - 1. \tag{5}$$

Expanding, we see that (5) is equivalent to $(2 - 7\lambda + 6\lambda^2 - 2\lambda^3)n \ge -1$. Since

$$2 - 7\lambda + 6\lambda^2 - 2\lambda^3 = 2(1-\lambda)^3 - \lambda = 0,$$

the result follows. □

Corollary 3.8. *Let $n \ge 6$. Then*

(i) $\frac{1}{2}\left(\binom{n}{2} + \lambda n(n - \lambda n)\right) \le f_2(n) \le \frac{1}{2}\left(\binom{n}{2} + \lceil \lambda n + \epsilon \rceil (n - \lceil \lambda n + \epsilon \rceil) + 1\right)$,

(ii) $f_2(n) \sim \left(\frac{1}{2} + \lambda(1-\lambda)\right)\binom{n}{2}$ *as* $n \to \infty$.

*Proof.* Statement (i) just combines the two inequalities established above (in Corollary 3.5 and Theorem 3.7) with Lemma 3.1. Dividing through by $\binom{n}{2}$ and taking the limit as $n \to \infty$ gives (ii). □

It is interesting to compare Corollary 3.8(ii) with the result we found in section 2 of this paper. If we allow an arbitrary number of edges to be deleted at each step of a disconnecting procedure for $K_n$ ($w = \infty$), we saw in Corollary 2.3 that the ratio of the number of steps required in an optimal procedure to the number of edges of $K_n$ is asymptotic to 2/3. Here we see that if we allow no more than 2 edges to be removed at a time ($w = 2$), then this ratio is asymptotic to $1/2 + \lambda(1-\lambda) = .74194412\ldots$.

Another look at the preceding inequalities can provide us with additional information. For instance, if we want to actually implement an optimal procedure for completely disconnecting $K_n$, we must know what the starting value of $k$ should be, whereby we split $K_n$ into $K_k \oplus K_{n-k}$. Our last result shows that, given any $n$ (suitably large), there are no more than two possible values for the starting value $k$. In practice we would try each of them to determine the right one.

Theorem 3.9. *Let $n \ge 32$. Suppose that an optimal procedure for completely disconnecting $K_n$ begins by reducing $K_n$ to $K_k \oplus K_{n-k}$, where $k$ is a positive integer such that $k \le n/2$. Then $\lambda n < k < \lambda n + 2$. In particular, $k$ must be one of the integers $\lceil \lambda n \rceil$ or $\lceil \lambda n \rceil + 1$. Moreover, each of these possibilities occurs for infinitely many values of $n$.*

*Proof.* If a procedure for completely disconnecting $K_n$ begins by reducing $K_n$ to $K_k \oplus K_{n-k}$, then there are at least $k(n-k)$ unit deletions required in the procedure. If $n/2 \geq k \geq \lambda n + 2$, then we would have $k(n-k) \geq (\lambda n+2)(n-(\lambda n+2))$. By Corollary 3.5, and noting that $\lambda n + \epsilon + 1 \leq n/2$ for $n \geq 18$,

$$u_2(n) \leq \lceil \lambda n + \epsilon \rceil (n - \lceil \lambda n + \epsilon \rceil) + 1 \leq (\lambda n + \epsilon + 1)(n - \lambda n - \epsilon - 1) + 1.$$

Thus the procedure could not be optimal if

$$(\lambda n + \epsilon + 1)(n - \lambda n - \epsilon - 1) + 1 < (\lambda n + 2)(n - \lambda n - 2).$$

This simplifies to

$$(\epsilon + 1)(1 - 2\lambda)n - (\epsilon + 1)^2 + 1 < 2(1 - 2\lambda)n - 4,$$

and then to

$$n > \frac{5 - (1+\epsilon)^2}{(1-\epsilon)(1-2\lambda)} \approx 31.436,$$

so if $n \geq 32$ we conclude $k < \lambda n + 2$. This establishes the right-hand part of our inequality.

To establish the left-hand inequality, we proceed as follows. Suppose we have a procedure for completely disconnecting $K_n$ which begins by reducing $K_n$ to $K_k \oplus K_{n-k}$. Suppose that $k \leq \lambda n - s$, where we will eventually set $s = 0$. By Lemma 3.6, once $K_n$ has been split into $K_k$ and $K_{n-k}$, there will still be at least $u_2(n-k) - \binom{k}{2}$ unit deletions required to finish the procedure. By Theorem 3.7, this number is at least as big as $\lambda(1-\lambda)(n-k)^2 - \binom{k}{2}$. Therefore the total number of unit deletions employed is at least

$$k(n-k) + \lambda(1-\lambda)(n-k)^2 - \binom{k}{2}.$$

Suppose that $0 < t \leq 2$ and that $\lceil \lambda n + \epsilon \rceil \leq \lambda n + t \leq n/2$ (actually we will only be using the values $t = 2$ and $t = 1$ when appropriate). Then by Corollary 3.5, an optimal procedure for completely disconnecting $K_n$ will use no more than $(\lambda n + t)(n - \lambda n - t) + 1$ unit deletions. Thus if we can show that

$$k(n-k) + \lambda(1-\lambda)(n-k)^2 - \binom{k}{2} > (\lambda n + t)(n - \lambda n - t) + 1, \tag{6}$$

then the procedure cannot be optimal. Now think of $n$ as fixed, and allow $k$ to be nonintegral for the moment. The left-hand side of (6) is a quadratic function in the variable $k$. The coefficient of $k^2$ in this quadratic is $\lambda(1-\lambda) - 3/2$, which is negative. Therefore the minimum value of the left side of (6) for $k$ in the interval $1 \leq k \leq \lambda n - s$ occurs at one of the two end points of the interval. So, to establish (6), it is enough to show that it holds for $k = 1$ and for $k = \lambda n - s$.

For $k = 1$, (6) is

$$n - 1 + \lambda(1-\lambda)(n-1)^2 > \lambda(1-\lambda)n^2 + t(1-2\lambda)n + 1 - t^2,$$

which simplifies to

$$[1 - 2\lambda(1-\lambda) - t(1-2\lambda)]n > 2 - t^2 - \lambda(1-\lambda). \tag{7}$$

But for $t \leq 2$ the coefficient of $n$ is at least $2\lambda^2 + 2\lambda - 1 \approx .157 > 0$, so (7) will be true for sufficiently large $n$. In fact when $t = 2$, the right side of (7) is negative and thus (7) will hold for all $n$. When $t = 1$, one checks that (7) holds for all $n > 2$.

For $k = \lambda n - s$, (6) becomes

$$(\lambda n - s)(n - \lambda n + s) + \lambda(1-\lambda)(n - \lambda n + s)^2 - \frac{1}{2}(\lambda n - s)(\lambda n - s - 1)$$

$$> (\lambda n + t)(n - \lambda n - t) + 1,$$

which simplifies to

$$\left[\lambda(1-\lambda)^3 - \frac{\lambda^2}{2}\right] n^2 + \left[3s\lambda - s + 2s\lambda(1-\lambda)^2 + \frac{\lambda}{2} - t(1-2\lambda)\right] n$$

$$+ \lambda(1-\lambda)s^2 - s^2 - \frac{s(s+1)}{2} + t^2 - 1 \ > \ 0. \tag{8}$$

Since $\lambda = 2(1-\lambda)^3$, the coefficient of $n^2$ is just 0. Furthermore, when $s = .4$ and $t = 2$, the coefficient of $n$ is

$$5.7\lambda + .8\lambda(1-\lambda)^2 - 2.4 \approx .05 > 0$$

and the constant term is also positive, so (3) holds. Since $\lceil \lambda n + \epsilon \rceil \leq \lambda n + 2 \leq n/2$ for all $n \geq 23$, this says that $k \leq \lambda n - .4$ cannot give rise to an optimal procedure. In fact, it also says that $k = \lfloor \lambda n \rfloor$ cannot result in an optimal procedure (and so $k > \lambda n$, since $k$ is actually an integer, remember) unless the fractional part of $\lambda n$ is less than .4. But in this case (recalling that $\epsilon = .52\ldots$), $\lceil \lambda n + \epsilon \rceil = \lceil \lambda n \rceil < \lambda n + 1 \leq n/2$, which means we are allowed to set $t = 1$. Now, putting $s = 0$ and $t = 1$ in (3), the coefficient of $n$ is $5\lambda/2 - 1 \approx .0256 > 0$ and the constant term is 0, so (3) holds and the procedure is not optimal for $k \leq \lambda n$. Thus $k > \lambda n$ as claimed.

In the last part of this proof we will need the following familiar fact: the fractional parts of the integer multiples of any fixed irrational number are dense in the unit interval (for example, see Theorem 439, p. 376 of [3]).

To show that $k = \lceil \lambda n \rceil + 1$ gives a nonoptimal procedure for infinitely many $n$, we let $n$ be a sufficiently large positive integer so that the inequality

$$(\lambda n + 1)(n - \lambda n - 1) + 1 < \left(\lambda n + \frac{3}{2}\right)\left(n - \lambda n - \frac{3}{2}\right) \tag{9}$$

is true and also so that $\lceil \lambda n \rceil = \lceil \lambda n + \epsilon \rceil$, which means that the fractional part of $\lambda n$ is less than $1 - \epsilon$. There are infinitely many such $n$ simply because $\lambda < 1 - \epsilon$, so that consecutive integer multiples of $\lambda$ cannot "jump the gap"; alternatively, we could use the fact that $\lambda$ is irrational and appeal to the theorem mentioned in the previous paragraph. For each such $n$ we have

$$\lceil \lambda n + \epsilon \rceil = \lceil \lambda n \rceil \leq \lambda n + 1 < \lambda n + \frac{3}{2} \leq \lceil \lambda n + \epsilon \rceil + 1 \leq \lceil \lambda n \rceil + 1,$$

and from (9) and Corollary 3.5 we conclude that $k = \lceil \lambda n \rceil + 1$ cannot result in an optimal procedure.

A similar method shows that $k = \lceil \lambda n \rceil$ gives a nonoptimal procedure for infinitely many $n$, but this time we have rather less room to maneuver. Since (as we have seen) the coefficient of $n$ in inequality (3) is positive when $s = 0$ and $t = 1$, it follows that this must also be true if we let $s$ be a sufficiently small (in absolute value) *negative* number and $t$ be greater than $1 - s$ but sufficiently close to $1 - s$. For such a choice of $s$ and $t$, (3) will then hold for sufficiently large $n$. Now find $n$ so that the fractional part of $\lambda n$ is greater than $1 + s$; again there exist infinitely many such $n$ because $\lambda$ is irrational. Then $\lceil \lambda n \rceil < \lambda n - s$, so

$$\lceil \lambda n + \epsilon \rceil \leq \lceil \lambda n \rceil + 1 < \lambda n + 1 - s < \lambda n + t,$$

which shows that with this value of $t$ we can use (3) to test whether the procedure with $k \leq \lambda n - s$ is optimal. Since (3) holds, any $k \leq \lambda n - s$, and, in particular, $k = \lceil \lambda n \rceil$ does not give an optimal procedure. □

We conclude this section by illustrating some of the above results for a few values of $n$. We will take $n = 24, 33$, and 34 as interesting case studies.

Let us consider the case $n = 24$. Here we calculate that $\lambda n = 9.845\ldots$ . We will consider the possibility of completely disconnecting $K_{24}$ by using each of the integers $k = 9, 10, 11, 12$ as the starting value when we split $K_{24}$ into $K_k \oplus K_{24-k}$. When we do this with $k = 9$, the initial disconnection requires $(15)(9) = 135$ unit deletions. Now one can show that any procedure for completely disconnecting $K_{15}$ will need at least $(8)(7) = 56$ unit deletions. Since $\binom{9}{2} = 36$, we see from Lemma 3.6 that proceeding to reduce $K_9 \oplus K_{15}$ will require at least $56 - 36 = 20$ more unit steps. Thus, at least $135 + 20 = 155$ unit deletions will be necessary if we use $k = 9$. If we begin instead by splitting $K_{24}$ into $K_{10} \oplus K_{14}$ this will require $(10)(14) = 140$ unit deletions. From here we can completely disconnect $K_{10} \oplus K_{14}$ with 4 more unit steps as follows. We note that $\binom{10}{2} = 45$ and that $(7)(7) = 49$. We now delete 90 edges from $K_{10} \oplus K_{14}$ in 45 pairs, one from each component, by matching 45 of the 49 edges needed to split $K_{14}$ into $K_7 \oplus K_7$ with the 45 edges of $K_{10}$. After doing this what remains are four of the 49 edges plus all the edges of $K_7 \oplus K_7$. We then use four unit steps to delete the four edges, and then we can finish off $K_7 \oplus K_7$ with no further unit steps involved. Note that Lemma 3.6 implies we cannot completely disconnect $K_{14} \oplus K_{10}$ with fewer than four unit deletions. So, with $k = 10$ we have a procedure which needs $140 + 4 = 144$ unit deletions, and this is the best we can do for this value of $k$. When we take $k = 11$ instead, our initial split is into $K_{11} \oplus K_{13}$, requiring $(11)(13) = 143$ unit deletions. Lemma 3.3 shows that $K_{11} \oplus K_{13}$ can then be finished with no more than one unit step needed. Since the number of edges of $K_{11} \oplus K_{13}$ is odd, one unit deletion will in fact be necessary here. Thus, with $k = 11$ we can completely disconnect $K_{24}$ by a procedure which uses $143 + 1 = 144$ unit deletions (and this is the best we can do for this $k$). This is also the case for $k = 12$: splitting $K_{24}$ into two equal components $K_{12} \oplus K_{12}$ requires $(12)(12) = 144$ unit deletions for the initial disconnection, after which all the rest of the edges can be deleted in doublets one from each of the two copies of $K_{12}$. Values of $k$ which are smaller than nine cannot even come close to 144. So when $n = 24$, there are three different values of $k$ with which one can implement an optimal procedure : $k = \lceil \lambda n \rceil = 10$, $k = \lceil \lambda n \rceil + 1 = 11$, and $k = \lceil \lambda n \rceil + 2 = 12$. By Theorem 3.9, when $n \geq 32$ it is not possible for $k = \lceil \lambda n \rceil + 2$ to give an optimal procedure. In fact, further checking reveals that $n = 24$ is actually the largest value of $n$ for which this can occur.

Next, let us consider the case $n = 33$. We calculate that $\lambda n = 13.538\ldots$ . By Theorem 3.9 we know an optimal procedure for completely disconnecting $K_{33}$ will begin by using one of the integers $k = 14$ or $k = 15$ as the starting value when we split $K_{33}$ into $K_k \oplus K_{33-k}$. When we do this with $k = 14$, the initial disconnection requires $(19)(14) = 266$ unit deletions. The number of edges of $K_{14}$ is 91. Since $(9)(10) = 90$, we can delete 90 pairs of edges, each consisting of one from the copy of $K_{14}$ and one from a cutset of size 90 in the copy of $K_{19}$ which reduces the latter to $K_{10} \oplus K_9$. One edge remains from the copy of $K_{14}$. Lemma 3.3 implies that we can then delete all the edges of $K_{10} \oplus K_9$ using at most one deletion. Since this latter graph has an odd number of edges, in fact one unit deletion will be used. When this unit deletion occurs, we can pair it with the one remaining edge from the copy of $K_{14}$ and delete these in one doublet step. This shows that, after the initial disconnection, no further unit deletions are required, in all just 266 unit deletions. If instead we begin by splitting $K_{33}$ into $K_{18} \oplus K_{15}$, this would already entail $(18)(15) = 270$ unit deletions. So we see that in this case, an optimal procedure must begin with $k = \lceil \lambda n \rceil = 14$.

Finally, consider the case $n = 34$. Here we calculate that $\lambda n = 13.9483\ldots$. By Theorem 3.9 we know an optimal procedure for completely disconnecting $K_{34}$ will begin by using one of the integers $k = 14$ or $k = 15$ as the starting value when we split $K_{34}$ into $K_k \oplus K_{34-k}$. When we do this with $k = 14$, the initial disconnection requires $(20)(14) = 280$ unit deletions. The number of edges of $K_{14}$ is 91. One can show that any procedure for completely disconnecting $K_{20}$ will require at least $(10)(10) = 100$ unit deletions. Therefore, by Lemma 3.6, after our initial disconnection, there will still be at least $100 - 91 = 9$ further unit deletions required, giving a total of at least $280 + 9 = 289$. Conversely, if we begin by splitting $K_{34}$ into $K_{15} \oplus K_{19}$, the initial stage requires $(19)(15) = 285$ unit deletions, and as Theorem 3.4 implies, at most one more unit deletion will be necessary to complete the job. This will beat the 289 unit deletions required for $k = 14$. We see that, for $n = 34$, an optimal procedure must begin with $k = \lceil \lambda n \rceil + 1 = 15$.

## 4. Conclusion.

We end with some questions regarding the case $w > 2$, $w$ finite. It is clear that as $w$ increases, the ratio of the number of steps in an optimal procedure to the number of edges in $K_n$ will decrease from $.74194412\ldots$ (the ratio when $w = 2$) toward $2/3$ (the ratio when $w = \infty$). Does this ratio converge to $2/3$ as $w \to \infty$? What is even less clear is this: What happens to the optimal value of $k$ as a proportion of $n$, the "starting ratio" referred to in section 3? For $w = 2$ it is $\lambda = .41024548\ldots$. Does it just approach $1/2$ as $w$ increases, or are there *finite* values of $w$ for which it has already been shown that the best procedure is to start by disconnecting $K_n$ into equal parts? For example, for $w = 4$ a reasonable-looking procedure is to start by cutting $K_n$ into two equal pieces using unit deletions, then use the above procedure for $w = 2$ simultaneously on both pieces. Is this optimal?

## REFERENCES

[1] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, North-Holland, New York, 1981.
[2] J. Ginsburg and B. Sands, *An optimal algorithm for a parallel cutting problem,* Ars Combin., to appear.
[3] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers,* 5th ed., Oxford University Press, Oxford, UK, 1979.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 1, pp. 48–55

# RECOGNIZING PERFECT 2-SPLIT GRAPHS[*]

CHÍNH T. HOÀNG[†] AND VAN BANG LE[‡]

**Abstract.** A graph is a split graph if its vertices can be partitioned into a clique and a stable set. A graph is a $k$-split graph if its vertices can be partitioned into $k$ sets, each of which induces a split graph. We show that the strong perfect graph conjecture is true for 2-split graphs and we design a polynomial algorithm to recognize a perfect 2-split graph.

**Key words.** graph coloring, perfect graph

**AMS subject classification.** 05C15

**PII.** S0895480197329089

## 1. Introduction.

A graph is a *split* graph if its vertices can be partitioned into a clique and a stable set. Split graphs were studied in [14] and [11]. Split graphs form a class of perfect graphs and can be recognized in polynomial time. A graph $G$ is *perfect* if for each induced subgraph $H$ of $G$ the chromatic number of $H$ equals the number of vertices in a largest clique of $H$. The strong perfect graph conjecture (SPGC), proposed by Berge [1], states that a graph is perfect if and if only it does not contain as an induced subgraph the odd chordless cycle with at least five vertices, called *odd hole*, or the complement of such a cycle, called *odd antihole* (for more information on perfect graphs, see [12] and [2].) Nowadays, graphs containing no odd holes and no odd antiholes are called *Berge* graphs. It is not known whether perfect graphs and Berge graphs can be recognized in polynomial time.

The purpose of this paper is to study a generalization of split graphs and its relation to the SPGC. We shall call a graph a *k-split* graph if its vertices can be partitioned into $k$ sets, each of which induces a split graph. The graph $C_5$ shows that 2-split graphs are not necessarily perfect. We will show that the SPGC holds for 2-split graphs and we will design a polynomial algorithm for recognizing perfect 2-split graphs.

THEOREM 1. *A 2-split graph is perfect if and only if it is Berge.*

In the remainder of this section, we shall prove Theorem 1. In the next section, we shall show that perfect 2-split graphs can be recognized in polynomial time.

A graph is *minimal imperfect* if it is not perfect but each of its proper induced subgraphs is. The proof of Theorem 1 relies on several known results on minimal imperfect graphs. First, Lovász [16] proved that

(1) every minimal imperfect graph $G$ has exactly $\omega(G) \cdot \alpha(G) + 1$ vertices,

where $\omega(G)$, respectively $\alpha(G)$, denotes the number of vertices in a largest clique, respectively, stable set, of $G$. Equation (1) implies that

(2) a graph is perfect if and only if it its complement is.

[*]Received by the editors October 20, 1997; accepted for publication (in revised form) May 7, 1999; published electronically January 13, 2000. This research was supported by NSERC.
http://www.siam.org/journals/sidma/13-1/32908.html

[†]School of Mathematical Sciences, Lakehead University, Thunder Bay, Ontario, Canada P7B-5E1 (chinh.hoang@lakeheadu.ca).

[‡]Fachbereich Informatik, Universität Rostock, Rostock, Germany (le@informatik.uni-rostock.de).

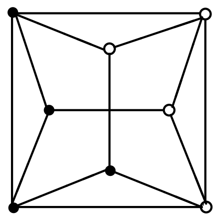


FIG. 1. *A perfect 2-split non-quasi-parity graph.*

Next, a result of Tucker [19] shows that

(3) $\omega(G) \geq 4$ and $\alpha(G) \geq 4$ for every minimal imperfect Berge graph $G$.

Theorem 1 is a consequence of (1) and (3).

*Proof of Theorem* 1. The "only if" part is trivial. We shall prove the "if" part by contradiction. Assume that there is an imperfect Berge graph $G$ satisfying the hypothesis of the theorem. Since $G$ contains an induced subgraph that is minimal imperfect, we may, without loss of generality, assume that $G$ is minimal imperfect. Let $T$ be a set of vertices in $G$ such that each of $T$ and $V(G) - T$ induces a split graph. Let $T$ be partitioned into a clique $C$ and a stable set $S$. Then

$$|T| = |C| + |S| \leq \omega(G) + \alpha(G).$$

Similarly,

$$|G - T| \leq \omega(G) + \alpha(G).$$

Hence by (1),

$$\omega(G)\alpha(G) + 1 = |G| \leq 2(\omega(G) + \alpha(G)).$$

This and (3) give

$$\begin{aligned} 1 &\leq 2\left(\frac{1}{\alpha(G)} + \frac{1}{\omega(G)}\right) - \frac{1}{\omega(G)\alpha(G)} \\ &< 2\left(\frac{1}{\alpha(G)} + \frac{1}{\omega(G)}\right) \\ &\leq 1, \end{aligned}$$

which is a contradiction. $\square$

The class of perfect 2-split graphs contains all split graphs, all bipartite graphs, and their complements. Therefore it is not contained in BIP* [8] and not in the class of strongly perfect graphs [3]. Meyniel [17] established perfectness for a large class of graphs, the class of "quasi-parity" graphs. An *even pair* is a pair of vertices such that all chordless paths joining them have an even number of edges. A graph $G$ is a *quasi-parity graph* if for each induced subgraph $H$ of $G$ either $H$ or its complement contains an even pair. The graph in Figure 1 is a perfect 2-split graph that is not a quasi-parity graph.

**2. Recognition algorithm.** In this section, we shall show that perfect 2-split graphs can be recognized in polynomial time. As usual, $n$ and, respectively, $m$ denote the number of vertices, respectively edges, of a graph $G$. A $(k,l)$ partition of a graph $G$ is a partition of its vertices into $k$ stable sets and $l$ cliques. A graph is a $(k,l)$ graph if its vertices admit a $(k,l)$ partition. Thus $k$-split graphs are $(k,k)$ graphs. $(k,l)$ graphs were first studied by Brandstädt [5], who remarked that deciding whether a graph is a $(k,l)$ graph is NP-complete whenever $k \geq 3$ or $l \geq 3$. However, he showed [4] that finding a $(2,1)$ partition (respectively, $(2,2)$ partition) in a graph can be done in $O(n^2m)$ (respectively, $O(n^{10}m)$) time. Brandstädt pointed out that the algorithms he gave in [5] are incorrect; however, his algorithms in [4] are correct. Brandstädt, Le, and Szymczak [6] later designed an $O((n+m)^2)$ algorithm to recognize (2,1) graphs. Recently, Feder et al. [10] gave a new algorithm to find a $(2,2)$ partition of a graph, if it exists; their algorithm also runs in $O(n^{10}m)$ time.

We shall show that an odd hole in a $(2,2)$ graph $G$, if it exists, can be found in $O(n^4m^3)$ time. Since the complement of a $(2,2)$ graph is again a $(2,2)$ graph, an odd antihole can also be detected in polynomial time. Thus, to determine if a given graph $G$ is a perfect 2-split graph, we first use the algorithm of Brandstädt to construct a (2,2) partition of $G$ (if such a partition exists) and then use our algorithm to test for the existence of an odd hole or antihole in $G$. This shows that perfect 2-split graphs can be recognized in polynomial time.

A chordless path, respectively cycle, on $k$ vertices is denoted by $P_k$, respectively $C_k$. The *length* of a path or a cycle is the number of its edges. We often write $P_k$ $x_1x_2\cdots x_k$ for the path on vertices $x_1, x_2, \ldots, x_k$ and edges $x_ix_{i+1}$ $(1 \leq i < k)$; $x_1$ and $x_k$ are called *end-points* of $P$. For the path $P_4$ $x_1x_2x_3x_4$, the vertices $x_2, x_3$ are called *midpoints*, the edges $x_1x_2, x_3x_4$ are called *wings* of that $P_4$, and the edge $x_2x_3$ is called the *rib* of that $P_4$. By $N_G(x)$ we denote the set of vertices adjacent to vertex $x$ in $G$; when there can be no confusion, we shall write $N(x) = N_G(x)$.

Let $G$ be a graph on $n$ vertices and $m$ edges. To analyze our algorithm, we shall need an upper bound on the number of $P_k$'s in $G$ for a certain number $k$. Since each edge $ab$ can extend into a $P_3$ in at most $n$ ways, the number of $P_3$'s in $G$ is of order $O(nm)$. Since each edge $ab$ can be the rib of at most $n^2$ $P_4$'s, the number of $P_4$ in $G$ is of order $O(n^2m)$. Similarly, the number of $P_6$'s is of order $O(n^4m)$.

The remainder of this section is devoted to showing that an odd hole (if it exists) in a $(2,2)$ graph can be detected in polynomial time. An $(x,y)$-path is an induced path whose end-points are vertices $x$ and $y$. The parity of a path is the parity of its number of edges. We note that the following two problems can obviously be solved in $O(n+m)$ time.

Problem Odd Path

| | |
|---|---|
| Input: | A $(2,0)$ graph $G$, and two nonadjacent vertices $x, y$ in $G$. |
| Question: | Is there an odd induced $(x,y)$-path in $G$? |

Problem Even Path

| | |
|---|---|
| Input: | A $(2,0)$ graph $G$, and two nonadjacent vertices $x, y$ in $G$. |
| Question: | Is there an even induced $(x,y)$-path in $G$? |

**Finding an odd hole in (2,1) graphs.** We shall describe an algorithm for finding an odd hole in a (2,1) graph $G$. Let the vertices of $G$ be partitioned into stable sets $S_1, S_2$, and a clique $C_1$. If $G$ contains an odd hole $H$, then $H$ must contain one or two vertices in $C_1$.

*Step* 1. *Look for an odd hole $H$ containing one vertex in $C_1$.*

For each vertex in $C_1$, list all $P_3$'s containing it as a midpoint. For each $P_3$ $abc$ with $b \in C_1, a, c \in S_1 \cup S_2$, $abc$ extends into an odd hole if and only if there is an odd induced $(a,c)$-path in the (2,0) subgraph of $G$ induced by $S_1 \cup S_2 - (N_G(b) - \{a,c\})$.

Since the number of $P_3$'s in $G$ is of order $O(nm)$, the complexity of this step is $O(nm^2)$.

*Step* 2. *Look for an odd hole $H$ containing two vertices in $C_1$.*

For each pair of vertices $b, c$ in $C_1$, list all $P_4$'s containing them as midpoints. For each $P_4$ $abcd$ with $b, c \in C_1, a, d \in S_1 \cup S_2$, $abcd$ extends into an odd hole if and only if there is an even induced $(a,d)$-path in the (2,0) subgraph of $G$ induced by $S_1 \cup S_2 - ((N_G(b) \cup N_G(c)) - \{a,d\})$.

Since the number of $P_4$'s in $G$ is of order $O(n^2m)$, the complexity of this step is $O(n^2m^2)$. Thus the complexity of finding an odd hole in a (2,1) graph is $O(n^2m^2)$.

**Finding an odd hole in (2,2) graphs.** Let the vertices of $G$ be partitioned into stable sets $S_1, S_2$ and cliques $C_1, C_2$. If there is an odd hole in $S_1 \cup S_2 \cup C_i$, for $i = 1, 2$, then we can find it using the previous algorithm. So we may assume that there is no odd hole in $S_1 \cup S_2 \cup C_i$, for $i = 1, 2$. Now, any odd hole $H$ in $G$ must contain some vertex in $C_1$ and some vertex in $C_2$; we let $H_i$ denote the set of vertices in $H \cap C_i$ for $i = 1, 2$.

First, we can look for a $C_5$ in $G$ in $O(n^5)$ (actually, we can do slightly better). Thus we may assume that $G$ has no $C_5$.

*Step* 1. *Look for an odd hole $H$ such that all the vertices in $H' = H_1 \cup H_2$ appear consecutively in $H$.*

In this case $H'$ induces a $P_k$ $v_1 \dots v_k$ $(k = 2, 3, 4)$ in $G$. Thus we need only to test if this $P_k$ extends into an odd hole. For each induced path $P$ $xv_1 \dots v_k y$ with $x, y \in S_1 \cup S_2$, $P$ extends into an odd hole if and only if there is an even (respectively, odd) induced $(x,y)$-path in the subgraph of $G$ induced by $S_1 \cup S_2 - ((N_G(v_1) \cup N_G(v_2) \dots \cup N_G(v_k)) - \{x,y\})$ if $k$ is even (respectively, odd).

Since the number of $P_6$'s in $G$ is of order $O(n^4m)$, the complexity of this step is $O(n^4m^2)$.

*Step* 2. *Look for an odd hole $H$ such that not all vertices in $H' = H_1 \cup H_2$ appear consecutively in $H$.*

We shall test (i) whether there is a $P_3$ $abc$ with $b \in C_i, a, c \in S_1 \cup S_2$, such that $abc$ extends into an odd hole; and (ii) whether there is a $P_4$ $abcd$ with $b, c \in C_i, a, d \in S_1 \cup S_2$, such that $abcd$ extends into an odd hole.

A $P_3$ $abc$ with $b \in C_i, a, c \in S_1 \cup S_2$ is of type 1 if $a, c \in S_j$ for $j = 1, 2$; otherwise the $P_3$ is of type 2. A $P_4$ $abcd$ with $b, c \in C_i, a, d \in S_1 \cup S_2$ is of type 1 if $a \in S_i, d \in S_j, i \neq j$; otherwise the $P_4$ is of type 2.

*Substep* 2.1. For each $P_3$ $abc$ of type 1 with $b \in C_i, a, c \in S_j$, we can test if it extends into an odd hole in the following way. If $abc$ does not extend into a $P_4$, then it does not extend into an odd hole. Now, consider a $P_4$ of the form $abcd$. We must have $d \in C_{i'} \cup S_{j'}, i \neq i', j \neq j'$. Let $F$ be the subgraph of $G$ induced by $C_{i'} \cup S_1 \cup S_2 - (N_G(b) \cup N_G(c) - \{a,d\})$ and let $F'$ be the graph obtained from $F$ by adding the edge $ad$. Since $G$ contains no $C_5$, the $P_4$ $abcd$ extends into an odd hole in $G$ if and only if there is an odd hole in $F'$ (this odd hole must contain the edge $ad$ by our assumption that $C_{i'} \cup S_1 \cup S_2$ contains no odd hole).

Since $F'$ is a (2,1) graph, this problem can be solved in $O(n^2m^2)$. Thus the complexity of this substep is $O(n^3m^3)$.

*Substep* 2.2. For each $P_4$ $abcd$ of type 1 with $b, c \in C_i, a \in S_1, d \in S_2$, let $F$ be the subgraph of $G$ induced by $C_{i'} \cup S_1 \cup S_2 - (N_G(b) \cup N_G(c) - \{a,d\}), i \neq i'$, and let $F'$ be the graph obtained from $F$ by adding the edge $ad$. Since $G$ contains no $C_5$, the $P_4$ $abcd$ extends into an odd hole in $G$ if and only if there is an odd hole in $F'$.

Since $F'$ is a (2,1) graph, this problem can be solved in $O(n^2m^2)$. Thus the complexity of this substep is $O(n^4m^3)$.

Now, we may assume that no $P_3$ and no $P_4$ of type 1 extend into a odd hole of $G$. We claim that

$$G \text{ contains no odd hole.}$$

Suppose that $G$ contains an odd hole $H$. Since we are in Step 2, the vertices in $H \cap (C_1 \cup C_2)$ do not appear consecutively on $H$. Enumerate the vertices of $H$ as $x_1, \dots, x_t, b_1, \dots, b_u, y_1, \dots, y_r, a_1, \dots, a_s$ in the cyclic order with $x_i \in C_1$ for each $i$, $b_i, a_i \in (S_1 \cup S_2)$ for each $i$, and $y_i \in C_2$ for each $i$. Define $Q_1$ to be the path $a_s x_1 \dots x_t b_1$ and $Q_2$ to be the path $b_u y_1 \dots y_r a_1$. $Q_1$ and $Q_2$ are a $P_3$ or $P_4$ of type 2.

Suppose that $Q_1$ is a $P_3$ of type 2. If $Q_2$ is a $P_3$ of type 2, then the path $b_1 \dots b_u$ has the same parity as the path $a_1 \dots a_s$. However, then $H$ has even length, which is a contradiction. Thus $Q_2$ must be a $P_4$ of type 2. Now, the path $b_1 \dots b_u$ has different parity than the path $a_1 \dots a_s$; then $H$ has even length, which is a contradiction.

Thus we may assume that $Q_1$, and by symmetry $Q_2$, is a $P_4$ of type 2. Now, it is easy to see that the path $b_1 \dots b_u$ has the same parity as the path $a_1 \dots a_s$. However, then $H$ has even length, which is a contradiction. Thus, $G$ cannot contain an odd hole as claimed. ∎

**Optimizing perfect and nonperfect 2-split graphs.** Grötschel, Lovász, and Schrijver [13] designed a polynomial algorithm to find a largest clique and an optimal coloring of a perfect graph. This important algorithm is a variation of the ellipsoid method for linear programming and is unlike a typical combinatorial algorithm. Thus, one may ask for a more combinatorial algorithm. With this in mind we shall comment on the problem of optimizing 2-split graphs.

Let $G$ be a graph with a (2,2) partition $C_1, C_2, S_1, S_2$, where the $C_i$'s are cliques and the $S_i$'s are stable sets. Since we can optimally color $C_1 \cup C_2$, we can color $G$ with at most $\chi(G) + 2$ colors ($\chi(G)$ denotes the chromatic number of $G$). We do not know if there is a combinatorial algorithm to optimally color a perfect 2-split graph.

We shall show that a largest clique of a 2-split graph can be found in polynomial time. It is well known that there is a combinatorial polynomial algorithm for finding a largest stable set and a minimum clique cover of a bipartite graph (for example, see [7, pp. 331–336]). We shall assume that the following function can be computed in polynomial time.

```
function Find-Omega-Bip(G)
  input: a graph $G$ that is the complement of a bipartite graph.
  output: a number Find-Omega-Bip(G)=$\omega(G)$.
```

The clique number of a (2,2) graph can be computed in polynomial time by the following function.

```
function Find-Omega(G)
  input: a (2,2) graph $G$ with partition $C_1, C_2, S_1, S_2$, where the $C_i$'s are cliques and
         the $S_i$'s are stable sets.
  output: a number Find-Omega(G)=$\omega(G)$.
begin
max := Find-Omega-Bip($C_1 \cup C_2$);
for each vertex $b \in S_1 \cup S_2$ do
  begin
```

```
        $k$ := Find-Omega-Bip ($N(b) \cap (C_1 \cup C_2)$);
        **if** $k+1 >$ max **then** max := $k+1$;
    **end**;
**for** each edge $ab$ with $a, b \in S_1 \cup S_2$, **do**
    **begin**
        $k$ := Find-Omega-Bip ($N(a) \cap N(b) \cap (C_1 \cup C_2)$);
        **if** $k+2 >$ max **then** max := $k+2$;
    **end**;
**return** max;
**end**.
```

We are going to show that function Find-Omega correctly computes the clique number of $G$. Let $\omega(C_1 \cup C_2) = k$. Obviously, we have $k \leq \omega(G) \leq k+2$. If $\omega(G) = k$, then the value returned by the function is obviously correct. If $\omega(G) = k+1$, then any largest clique of $G$ must contain a vertex in $B = S_1 \cup S_2$; such a clique can be found by finding, for each vertex $b \in B$, a largest clique in $N(b) \cap (C_1 \cup C_2)$. If $\omega(G) = k+2$, then any largest clique of $G$ must contain two adjacent vertices in $B = S_1 \cup S_2$; such a clique can be found by finding, for each pair of adjacent vertices $b_1, b_2 \in B$, a largest clique in $N(b_1) \cap N(b_2) \cap (C_1 \cup C_2)$. Thus function Find-Omega always returns the correct value.

**3. Perfect 3-split graphs.** We have shown that the SPGC holds for 2-split graphs and we provided a polynomial algorithm for recognizing perfect 2-split graphs. A natural extension of this result would be to show that the SPGC holds for 3-split graphs and, more generally, $k$-split graphs for any fixed $k$. We shall show that a minimal imperfect Berge $k$-split graph cannot have many vertices. When there can be no confusion we shall write $\omega = \omega(G)$ and $\alpha = \alpha(G)$.

LEMMA 2. *For any $k \geq 2, l \geq 2$, any minimal imperfect $(k, l)$ graph has at most $k\alpha + l\omega - r$ vertices where $r = \min(k, l)$.*

We shall need a result of Padberg [18], who showed that

(4) if $G$ is a minimal imperfect graph, then any $\omega$-clique of $G$ is disjoint from precisely one $\alpha$-stable set of $G$, and vice versa.

*Proof of Lemma* 2. Suppose that $G$ is a minimal imperfect $(k, l)$ graph. Let $G$ be partitioned into $k$ stable sets $S_1, \dots, S_k$ and $l$ cliques $K_1, \dots, K_l$. Then $|V(G)| = \sum |S_i| + \sum |K_i|$.

If no stable set $S_i$ is a maximum stable set and no clique $K_i$ is a maximum clique, then the lemma holds trivially.

Suppose that some two cliques, say, $K_1$ and $K_2$, are $\omega$-cliques. Then no stable set $S_i$ can be an $\alpha$-stable set; otherwise $S_i$ is disjoint from $K_1$ and $K_2$, which is a contradiction to (4). Therefore, we have $|V(G)| = \sum |S_i| + \sum |K_i| \leq k(\alpha - 1) + l\omega$. Similarly, if two stable sets $S_i$ and $S_j$ are $\alpha$-stable sets, then we have $|V(G)| \leq k\alpha + l(\omega - 1)$. Thus the lemma holds in this case.

Now, assume that exactly one clique of $\{K_1, \dots, K_l\}$ is a $\omega$-clique and exactly one stable set of $\{S_1, \dots, S_k\}$ is an $\alpha$-stable set. We have, therefore, $|V(G)| \leq \alpha + (k-1)(\alpha - 1) + \omega + (l-1)(\omega - 1) = k\alpha + l\omega - (k+l) + 2$. Since $k \geq 2$ and $l \geq 2$, the lemma also holds in this case. $\square$

THEOREM 3. *3-split Berge graphs are perfect if the SPGC holds for graphs with at most* 33 *vertices. Furthermore,* $(2,3)$ *Berge graphs and* $(3,2)$ *Berge graphs are perfect.*

*Proof of Theorem* 3. Let $G$ be a 3-split Berge graph and suppose that $G$ is not perfect. Then $G$ contains a minimal imperfect graph and so, without loss of generality, we may assume that $G$ is minimal imperfect. By (1) and Lemma 2, we have

$$|V(G)| = \alpha\omega + 1 \leq 3(\alpha + \omega) - 3. \tag{5}$$

By replacing $G$ with its complement if necessary, we may assume that $\alpha \geq \omega$. Now (5) gives $\alpha\omega + 1 \leq 6\alpha - 3$ and so $\omega \leq 5$. This and (3) imply

$$4 \leq \omega \leq 5. \tag{6}$$

From (5) and (6), it is easy to verify that the largest value that $|V(G)|$ can have is 33 and this occurs when $\alpha = 8$ and $\omega = 4$ (for example, if $\omega = 4$ and $\alpha = 9$, then $\alpha\omega + 1 = 37$ but $3(\alpha + \omega) - 3 = 36$, contradicting (5); similarly, when $\omega = 5$ we must have $\alpha = 5$ and so $|V(G)| = \alpha\omega + 1 = 26$).

Similarly, for the case $k = 2$, $l = 3$, we have $|V(G)| \leq 21$. This occurs when $\omega = 5, \alpha = 4$. (2,3) Berge graphs are perfect because the SPGC holds for graphs with at most 24 vertices. (This unpublished proof is due to V. A. Gurvich and V. M. Udalov, who announced it at the Perfect Graph Conference in Princeton, 1993, and is also reported in [9]; [15] gives a proof that the SPGC holds for graphs with at most 20 vertices.) By (2), (3,2) Berge graphs are also perfect. ☐

**The complexity of recognizing perfect (3,2) graphs.** We shall now comment on the difficulty of recognizing Berge (3,2) graphs. Tucker [19] proved that Berge $K_4$-free graphs are perfect; in other words, they are 3-colorable. The problem of recognizing Tucker's graphs is still open. 3-colorable graphs are (3,0) graphs. However, determining if a graph is a $(3, l)$ graph is at least as hard as determining if a graph is a (3,0) graph, for any $l \geq 1$. To see this, consider a graph $G$ and a graph $H$ that is the union of $G$ and $l$ vertex-disjoint cliques on at least four vertices. Clearly, $G$ is a (3,0) graph if and only if $H$ is a $(3, l)$ graph, and $G$ is Berge if and only if $H$ is. Thus, recognizing Berge (3,2) graphs is at least as hard as recognizing Berge $K_4$-free graphs. Thus, our algorithm for recognizing Berge (2,2) graphs seems to be the best possible (in the sense that it is polynomial) given the current state of knowledge on perfect graphs.

## REFERENCES

[1] C. Berge, *Les problèmes de coloration en théorie des graphes*, Publ. Inst. Stat. Univ. Paris, 9 (1960), pp. 123–160.

[2] C. Berge and V. Chvátal, eds., *Topics on Perfect Graphs*, North-Holland, Amsterdam, 1984.

[3] C. Berge and P. Duchet, *Strongly perfect graphs*, in Topics on Perfect Graphs, C. Berge and V. Chvátal, eds., North-Holland, Amsterdam, 1984, pp. 57–61.

[4] A. Brandstädt, *Partitions of graphs into one or two independent sets and cliques*, Report 105, Informatik Berichte, Fern Universität, Hagen, Germany, 1991.

[5] A. Brandstädt, *Partitions of graphs into one or two independent sets and cliques*, Discrete Math., 152 (1996), pp. 47–54. See also Corrigendum, 186 (1998), p. 295.

[6] A. Brandstädt, V. B. Le, and T. Szymczak, *The complexity of some problems related to graph 3-colorability*, Discrete Appl. Math., 89 (1998), pp. 59–73.

[7] V. Chvátal, *Linear Programming*, W. H. Freeman, New York, 1983.

[8] V. Chvátal, *Star-cutsets and perfect graphs*, J. Combin. Theory Ser. B, 39 (1985), pp. 189–199.

[9] V. Chvátal, *Problems Concerning Perfect Graphs*, Department of Computer Science, Rutgers University, 1993, manuscript.

[10] T. Feder, P. Hell, S. Klein, and R. Motwani, *Complexity of Graph Partition Problems*, manuscript.

[11] S. Foldes and P. L. Hammer, *Split graphs*, in Proceedings 8th Southeastern Conf. on Combinatorics, Graph Theory and Computing, Louisiana State University, Baton Rouge, LA, 1977, Congress. Numer. 19, Utilitas Math., Winnipeg, Manitoba, 1977, pp. 311–315.

[12] M. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[13] M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica, 1 (1981), pp. 169–197.

[14] P. L. Hammer and B. Simeone, *The splittance of a graph*, Research Report CORR 77-39, Department of Combinatorics and Optimization, University of Waterloo, Ontario, Canada, 1977.

[15] C. W. H. Lam, S. Swiercz, L. Thiel, and E. Regener, *A computer search for* $(\alpha, \omega)$*-graphs*, in Proc. 9th Manitoba Conference on Numerical Mathematics and Computation, Congress. Numer. 27, Utilitas Math., Winnipeg, Manitoba, 1980, pp. 285–289.

[16] L. Lovász, *A characterization of perfect graphs*, J. Combin. Theory Ser. B, 13 (1972), pp. 95–98.

[17] H. Meyniel, *A new property of critical imperfect graphs and some consequences*, European J. Combin., 8 (1987), pp. 313–316.

[18] M. Padberg, *Perfect zero-one matrices*, Math. Programming, 6 (1974), pp. 180–196.

[19] A. Tucker, *Critical perfect graphs and perfect* 3*-chromatic graphs*, J. Combin. Theory Ser. B, 23 (1977), pp. 143–149.

SIAM J. Discrete Math. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 1, pp. 56–63

# A BEST POSSIBLE DETERMINISTIC ON-LINE ALGORITHM FOR MINIMIZING MAXIMUM DELIVERY TIME ON A SINGLE MACHINE*

J. A. HOOGEVEEN† AND A. P. A. VESTJENS‡

**Abstract.** We consider a single-machine on-line scheduling problem where jobs arrive over time. A set of independent jobs has to be scheduled on the machine, where preemption is not allowed and the number of jobs is unknown in advance. Each job becomes available at its release date, which is not known in advance, and its characteristics, i.e., processing requirement and delivery time, become known at its arrival. The objective is to minimize the time by which all jobs have been delivered. We propose and analyze an on-line algorithm based on the following idea: As soon as the machine becomes available for processing, choose an available job with highest priority, and schedule it if its processing requirement is not too large. Otherwise, postpone the start of this job. We prove that our algorithm has performance bound $(\sqrt{5}+1)/2 \approx 1.61803$, and we show that there cannot exist a deterministic on-line algorithm with a better performance ratio for this problem.

**Key words.** on-line algorithm, single-machine scheduling, worst-case analysis

**AMS subject classification.** 90B35

**PII.** S0895480196296823

## 1. Introduction.

Until a few years ago, one of the basic assumptions made in deterministic scheduling was that all of the information needed to define the problem instance was known in advance. This assumption is usually not valid in practice, however. Abandoning it has led to the rapidly emerging field of on-line scheduling; for an overview, see Sgall [1998]. Two on-line models have been proposed. The first one assumes that there are no release dates and that the jobs arrive in a list. The on-line algorithm has to schedule the first job on this list before it sees the next job on the list (e.g., see Graham [1966] and Chen, Van Vliet, and Woeginger [1994]). The second model assumes that jobs arrive over time. Besides the presence of release dates, the main difference between the models is that in the second model, jobs do not have to be scheduled immediately upon arrival. At each time that the machine is idle, the algorithm decides which one of the available jobs is scheduled, if any. Within the second model, we can make a further distinction depending on whether the characteristics of a job, e.g., its processing requirement, become known at its arrival or not. The first variant has been adopted by Phillips, Stein, and Wein [1998], Vestjens [1997], and Hall, et al. [1997]; Motwani, Phillips, and Torng [1994] and Shmoys, Wein, and Williamson [1995] study the second variant. In this paper we consider a single-machine on-line scheduling problem that falls in the first category of the second model; that is, the jobs arrive over time, and the characteristics of each job become known at its arrival time.

Our objective is to minimize the time by which all jobs have been delivered. In this problem, after their processing on the machine, the jobs need to be delivered,

*Received by the editors January 2, 1996; accepted for publication (in revised form) April 9, 1999; published electronically January 13, 2000. The research leading to this paper was conducted when both authors were working at Eindhoven University of Technology.
http://www.siam.org/journals/sidma/13-1/29682.html

†Department of Computer Science, Utrecht University, P.O. Box 80089, 3508 TB Utrecht, The Netherlands (slam@cs.uu.nl).

‡Center for Quantative Methods CQM, P.O. Box 414, 5600 AK Eindhoven, The Netherlands (Vestjens@cqm.nl).

which takes a certain *delivery time*. The corresponding off-line problem is strongly NP-hard, but the preemptive version can be solved in polynomial time through an on-line algorithm (e.g., see Lawler et al. [1993]). A well known on-line algorithm for this problem is the LDT rule: choose from among the available jobs the one with the largest delivery time; this algorithm solves the problem if all jobs are released at the same time. For the case with unequal release dates, Kise, Iberaki, and Mine [1979] prove that LDT has a performance guarantee of 2. The question is, of course, can we do better from a worst-case point of view? In the remainder of the paper we show that the answer is yes: we present a deterministic algorithm with worst-case ratio $(\sqrt{5}+1)/2 \approx 1.61803$. Moreover, we show that we cannot do any better: every deterministic on-line algorithm has a worst-case ratio of at least that value. Recently, Seiden (1998) has developed a randomized algorithm with worst-case bound 1.55370. This algorithm randomly selects one of two deterministic algorithms and runs it. Furthermore, Seiden has shown that no such algorithm can do better than 1.5.

Throughout the paper we use $J_j$ to denote job $j$, and $r_j$, $p_j$, and $q_j$ to denote the release date, processing requirement, and delivery time of $J_j$, respectively. We denote by $S_j(\sigma)$, $C_j(\sigma)$, and $L_j(\sigma)$, the starting time, completion time, and the time by which $J_j$ is delivered in schedule $\sigma$, respectively. We use $\sigma$ to denote the schedule produced by the heuristic and $\pi$ to denote an optimal schedule.

## 2. Maximum delivery time.

In this section, we present an on-line $\alpha$-approximation algorithm for the single-machine scheduling problem of minimizing the time by which all jobs have been delivered, where $\alpha = (\sqrt{5}+1)/2$, and show that no on-line algorithm can do better from a worst-case point of view. We start with the latter. We prove the lower bound on the worst-case ratio by means of an example for which any on-line algorithm will have at least the required ratio. Let $L_{\max}(\pi)$ denote the minimum time by which all jobs can be delivered, and let $L_{\max}(\sigma)$ denote the time by which all jobs are delivered in schedule $\sigma$, where $\sigma$ is obtained through some on-line algorithm.

THEOREM 2.1. *Any on-line algorithm has a worst-case ratio of at least $\alpha$.*

*Proof.* Consider the following situation. The first job arrives at time 0 and has processing requirement $p_1 = 1$ and delivery time $q_1 = 0$. The on-line algorithm decides to schedule the job at time $S$. Depending on $S$, either no jobs arrive any more or one job with processing requirement $p_2 = 0$ and delivery time $q_2 = 1$ arrives at time $r_2 = S + \epsilon$. In the first case it is optimal to start the only job at time zero, giving $L_{\max}(\pi) = 1$, and we get a ratio of $L_{\max}(\sigma)/L_{\max}(\pi) = (S+1)/1$; this implies that to beat the ratio of 2, we must choose $S < 1$. In the second case it is optimal to schedule $J_2$ before $J_1$, because $S < 1$, $L_{\max}(\pi) = S + 1 + \epsilon$, and we get a ratio of $L_{\max}(\sigma)/L_{\max}(\pi) = (S+2)/(S+1+\epsilon)$. Hence,

$$\frac{L_{\max}(\sigma)}{L_{\max}(\pi)} \geq \max\left\{S+1, \frac{S+2}{S+1+\epsilon}\right\}.$$

If the on-line algorithm knew beforehand that the adversary would feed it one of these two instances, then it could account for this optimally by choosing $S$ so as to minimize this expression. Obviously, any algorithm lacking this information does at least as badly. Some simple algebra shows that the best choice for $S$ is

$$S = \frac{1}{2}(\sqrt{5+2\epsilon+\epsilon^2} - 1 - \epsilon).$$

This implies a worst-case ratio of

$$\frac{L_{\max}(\sigma)}{L_{\max}(\pi)} \geq \frac{1}{2}(\sqrt{5+2\epsilon+\epsilon^2}+1-\epsilon).$$

If we let $\epsilon$ tend to zero, then we get the desired ratio of $\alpha$. □

We can use the example of Theorem 2.1 to show that any on-line algorithm that schedules a job as soon as the machine is available will have a worst-case ratio of at least 2. Note that a simple algorithm like LDT already achieves this bound. If an algorithm wants to guarantee a better performance bound, then it needs a waiting strategy, which sometimes allows the algorithm to wait for more information; this decreases the impact of a "wrong" decision by the algorithm, as the adversary has to release jobs further into the future to disturb the algorithm. Therefore, we modify the LDT rule and call the new rule the delayed LDT rule (D-LDT). The basic idea behind the algorithm is that, if none of the available jobs has a "large" processing requirement, then we schedule the job with the largest delivery time; otherwise, we decide whether to schedule the big job, the job with the largest delivery time, or no job at all.

Throughout this section, we use the following notation:

- $p(S)$ denotes the total processing time of all jobs in $S$;
- $A(t)$ is the set containing all jobs that have arrived at or before time $t$ and that have not been started by time $t$;
- $B(t)$ is the set containing all jobs that arrived at or before time $t$ and that were not completed before the last idle time period before time $t$; if there is no idle time before time $t$, then $B(t)$ contains all jobs;
- $p(t)$ denotes the index of the job with the largest processing time in $A(t)$;
- $q(t)$ denotes the index of the job with the largest delivery time in $A(t)$.

The set $A(t)$ can be interpreted as the set of jobs that still need to be entirely processed at time $t$. The set $B(t)$ can be interpreted as the set of all jobs in $A(t)$ plus the jobs scheduled in the block of contiguously scheduled jobs that is still being processed at time $t$. We call a job $J_j$ *big* at time $t$ if $p_j > (\alpha-1)p(B(t))$; this is equivalent to $\alpha p_j > p(B(t))$, as $\alpha(\alpha-1)=1$. Since $(\alpha-1) > \frac{1}{2}$, there can be no more than one big job at any time.

ALGORITHM D-LDT

```
STEP 0. If the machine is idle and a job is available at time $t$, determine $q(t)$ and $p(t)$; otherwise, wait until the machine is idle and a job is available.
STEP 1. If $p_{p(t)} \leq (\alpha-1)p(B(t))$, then schedule $J_{q(t)}$ and go to STEP 0.
STEP 2. If $t + p(A(t)) > r_{p(t)} + \alpha p_{p(t)}$, then
    STEP 2A. if $q_{q(t)} > (\alpha-1)p_{p(t)}$, then schedule $J_{q(t)}$;
    STEP 2B. if $q_{q(t)} \leq (\alpha-1)p_{p(t)}$, then schedule $J_{p(t)}$.
STEP 3. If $t + p(A(t)) \leq r_{p(t)} + \alpha p_{p(t)}$, then
    STEP 3A. if $q(t) \neq p(t)$, then schedule $J_{q(t)}$;
    STEP 3B. if $q(t) = p(t)$, then schedule any job but $J_{p(t)}$.
STEP 4. Go to STEP 0.
```

Note that if only one job $J_j$ is available and is marked as big at time $t$, then the algorithm schedules no job and waits either until time $r_j + (\alpha-1)p_j$ or until a new job arrives, whichever happens first. If no job is big at the decision time $t$, then the algorithm simply schedules job $J_{q(t)}$, i.e., the job with the largest delivery time. If job

$J_{p(t)}$ is big at time $t$, then the algorithm checks whether the available jobs can keep the machine busy until time $r_{p(t)} + \alpha p_{p(t)}$. If this is not the case, then the algorithm schedules the job with maximum delivery time, unless this is the big job; it schedules some other job then. If the jobs can keep the machine busy long enough, then we can permit scheduling the big job $J_{p(t)}$, unless $J_{q(t)}$ has a huge delivery time (more than $(\alpha - 1)p_{p(t)}$).

We will prove that D-LDT has worst-case ratio $\alpha$. In our proof, we work with a smallest counterexample, where size is measured in terms of the number of jobs. Let $\mathcal{I}$ be such a smallest counterexample, and let $\sigma$ be the schedule created by D-LDT for $\mathcal{I}$. We abuse the notation $\mathcal{I}$ to denote the instance as well as the set of jobs in the instance. Let $J_l$ denote the first job in $\sigma$ that assumes the value $L_{\max}(\sigma)$. To prove that such a counterexample cannot exist, we start by proving four structure properties that the schedule $\sigma$ for an alleged smallest counterexample must satisfy.

OBSERVATION 2.2. *The schedule $\sigma$ consists of a single block: it starts with a nonnegative period of idle time after which all jobs are executed contiguously.*

*Proof.* Suppose to the contrary that $\sigma$ does not have this form. We then show that either we can find a counterexample that consists of a smaller number of jobs, or that this alleged counterexample is not a counterexample at all.

Suppose that $\sigma$ consists of more than one block. Let $B$ be the block in which $J_l$ is scheduled. Since the algorithm bases its choices on the set $B(t)$, the existence of the jobs that are completed before the start of block $B$ does not influence the start time of $B$ and the order in which the jobs are executed. Therefore, we can remove all jobs that are completed before the start of block $B$ without changing the value $L_{\max}(\sigma)$ and without increasing $L_{\max}(\pi)$. Similarly, we can remove all jobs from $\mathcal{I}$ that are released after the start of $J_l$ in $\sigma$. Therefore, we may assume that our counterexample consists of the jobs from block $B$ and the jobs that are available at the start of $J_l$ in $\sigma$ but that are scheduled in another block.

Let $S_B$ and $C_B$ denote the start time of the first job and the completion time of the last job in $B$. Since D-LDT inflicts unnecessary idle time only if there is one job available that is marked big, we know that there is only one job in $\mathcal{I}$ that does not belong to $B$ and that this job, which we denote by $J_i$, is marked big at time $C_B$, i.e., $\alpha p_i > p(B(C_B)) = p_i + p(B)$.

Let $J_0$ be the job that arrives first, which may be equal to $J_i$. Due to the working of the algorithm, $S_B = \min\{r_0 + (\alpha - 1)p_0, r_1\}$, where $r_1$ denotes the release date of the second available job. Since $J_l$ is a job in $B$, we know that $L_{\max}(\sigma) = L_l(\sigma) = C_l(\sigma) + q_l \le C_B + q_l$. If $J_0$ is the first job scheduled in $\pi$, then $L_{\max}(\pi) \ge L_l(\pi) \ge r_0+p_0+q_l > S_B+q_l$, from which we derive that $L_{\max}(\sigma)-L_{\max}(\pi) < C_B-S_B = p(B)$. If $J_0$ is not the first job in $\pi$, then the first job in $\pi$ cannot start before time $r_1 \ge S_B$, which implies that $L_{\max}(\pi) \ge L_l(\pi) \ge S_B+p_l+q_l$, and we again have that $L_{\max}(\sigma) - L_{\max}(\pi) \le C_B - S_B = p(B)$. Above, we have already shown that $\alpha p_i > (p(B) + p_i)$, which implies that $L_{\max}(\sigma) - L_{\max}(\pi) \le p(B) < (\alpha - 1)p_i \le (\alpha - 1)L_{\max}(\pi)$, and we conclude that $\mathcal{I}$ then does not correspond to a counterexample. ◻

From now on, let $J_0$ denote the job that arrives first in $\mathcal{I}$. Without loss of generality we may assume that $r_0 = 0$.

OBSERVATION 2.3. *For all $J_j \in \mathcal{I} \setminus \{J_0\}$, we have that $p_j \le (\alpha - 1)p(\mathcal{I})$.*

*Proof.* Suppose to the contrary that there does exist a job $J_j$ with $r_j \ge r_0$ that has $p_j > (\alpha-1)p(\mathcal{I})$, i.e., $\alpha p_j > p(\mathcal{I})$. Then at time $r_j$ there are at least two jobs available, which implies that the algorithm starts a job if it has not done so already. On the basis of Observation 2.2, we may conclude that there is no idle time in the remainder of the schedule. But since $J_j$ is marked as big by the algorithm for any time $t$, this can be the case only if the other jobs are able to keep the machine busy from time $r_j$ to time $r_j+(\alpha-1)p_j$. In that case, however, $(\alpha-1)p_j \leq p(\mathcal{I})-p_j < \alpha p_j - p_j = (\alpha-1)p_j$, which is a contradiction. □

Let $J_k$ denote the last job scheduled in $\sigma$ before $J_l$ with a delivery time smaller than $q_l$, if it exists. Let $G(l)$ denote the set of jobs containing $J_l$ and all jobs between $J_k$ and $J_l$ in $\sigma$ if $J_k$ exists, and all jobs scheduled before and including $J_l$ in $\sigma$ otherwise. Note that, due to the definition of $J_k$, all jobs in $G(l)$ have delivery time greater than or equal to $q_l$. As in Potts [1980], we shall refer to $J_k$ as the *interference job* for the schedule $\sigma$ as it may delay the times at which the jobs in $G(l)$ are delivered by occupying the machine when at least one of the jobs in $G(l)$ becomes available for processing. We now show that the schedule $\sigma$ contains an interference job and, furthermore, that this job has a considerable processing requirement.

OBSERVATION 2.4. *$\sigma$ contains an interference job $J_k$.*

*Proof.* Due to the definition of $G(l)$, we have that

$$L_{\max}(\pi) \geq \sum_{j \in G(l)} p_j + q_l.$$

The first job in $\sigma$ starts at time $(\alpha-1)p_0$, at the latest. If there is no interference job, then $G(l)$ contains all the jobs that have been scheduled in $\sigma$ before time $C_l(\sigma)$ and, consequently,

$$L_{\max}(\sigma) = C_l(\sigma) + q_l \leq (\alpha-1)p_0 + \sum_{j \in G(l)} p_j + q_l.$$

Hence, $L_{\max}(\sigma) - L_{\max}(\pi) \leq (\alpha-1)p_0 \leq (\alpha-1)L_{\max}(\pi)$, which contradicts the fact that we consider a counterexample. □

OBSERVATION 2.5. *$p_k > (\alpha-1)p(\mathcal{I})$.*

*Proof.* Suppose that $(\alpha-1)p(\mathcal{I}) \geq p_k$. We show that $\mathcal{I}$ then does not correspond to a counterexample.

To show that $\mathcal{I}$ does not correspond to a counterexample, we must prove that $L_{\max}(\sigma) = C_l + q_l = S_k(\sigma) + p_k + p(G(l)) + q_l \leq \alpha L_{\max}(\pi)$. The difficulty lies in the term $(S_k(\sigma) + p_k)$; we need to bound this. Therefore, we take a closer look at the situation in $\sigma$ at time $S_k(\sigma)$.

There are three possible reasons why the algorithm selected $J_k$ and not one of the jobs from $G(l)$:

(1) All jobs in $G(l)$ have a release date larger than $S_k(\sigma)$.

(2) $J_k$ is big at time $S_k(\sigma)$, and all of the jobs from $G(l)$ that are available at time $S_k(\sigma)$ have a delivery time of at most $(\alpha-1)p_k$; this corresponds to the situation described in STEP 2B.

(3) There is one job from $G(l)$ available at $S_k(\sigma)$, which we denote by $J_i$, that is marked as big, and the algorithm is not allowed to start it; this corresponds to the situation described in STEP 3.

*Case* 1. In this case, we have that

$$L_{\max}(\pi) \geq \min_{j \in G(l)} r_j + \sum_{j \in G(l)} p_j + q_l > S_k(\sigma) + \sum_{j \in G(l)} p_j + q_l.$$

Hence, we deduce that $L_{\max}(\sigma) - L_{\max}(\pi) < p_k \leq (\alpha-1)p(\mathcal{I}) \leq (\alpha-1)L_{\max}(\pi)$.

*Case* 2. Since the available jobs from $G(l)$ have a delivery time of size no more than $(\alpha-1)p_k$ and $J_l$ is the job with minimum delivery time in $G(l)$, we know that $q_l \le (\alpha-1)p_k$.

Now consider $\pi$. If $\pi$ does not have $J_0$ as its first job, then processing in $\pi$ will start at the same time or later than in $\sigma$, which implies that $C_{\max}(\sigma) \le C_{\max}(\pi)$. In this case, we derive

$$L_{\max}(\sigma) = L_l(\sigma) = C_l(\sigma) + q_l \le C_{\max}(\sigma) + (\alpha-1)p_k$$

$$\le C_{\max}(\pi) + (\alpha-1)p_k \le L_{\max}(\pi) + (\alpha-1)p_k \le \alpha L_{\max}(\pi).$$

Hence, $\mathcal{I}$ does not correspond to a counterexample in Case 2, unless $J_0$ is scheduled first in $\pi$. Then we have that $C_{\max}(\sigma) \le C_{\max}(\pi) + (\alpha-1)p_0$, as the first job in $\sigma$ is started no later than time $(\alpha-1)p_0$, and we obtain

$$L_{\max}(\sigma) = L_l(\sigma) = C_l(\sigma) + q_l \le C_{\max}(\sigma) + (\alpha-1)p_k$$

$$\le C_{\max}(\pi) + (\alpha-1)p_0 + (\alpha-1)p_k \le L_{\max}(\pi) + (\alpha-1)(p_0+p_k).$$

If $J_0 \ne J_k$, then $p_0 + p_k \le p(I) \le L_{\max}(\pi)$, which implies that $\mathcal{I}$ does not correspond to a counterexample then. Hence, we assume that $J_0 = J_k$. Since $\pi$ starts with $J_0$, we find that $L_{\max}(\pi) \ge p_0 + p(G(l)) + q_l$, which implies that $L_{\max}(\sigma) - L_{\max}(\pi) \le S_0(\sigma)$. Since $\mathcal{I}$ forms a counterexample, we must have $S_0(\sigma) > (\alpha-1)p_0$. But this implies that D-LDT must have applied STEP 2A at least once. Consider the set $Q$ containing the jobs that were processed from time $T$ to time $S_0(\sigma)$ in $\sigma$, where $T$ is the start time of the first job that is completed after time $(\alpha-1)p_0$ in $\sigma$. Since the jobs in $Q$ were preferred to $J_0$, all these jobs have delivery time more than $(\alpha-1)p_0 > q_l$. On basis of the jobs in $Q \cup G(l)$, we derive the following lower bound on $L_{\max}(\pi)$, where we keep in mind that $J_0$ is scheduled first in $\pi$:

$$L_{\max}(\pi) \ge p_0 + \sum_{j\in Q} p_j + \sum_{j\in G(l)} p_j + \min_{j\in(Q\cup G(l))} q_j = p_0 + p(Q) + p(G(l)) + q_l.$$

Since $L_{\max}(\sigma) = T + p(Q) + p_0 + p(G(l)) + q_l$, we obtain

$$L_{\max}(\sigma) - L_{\max}(\pi) \le T \le (\alpha-1)p_0,$$

and we conclude that $\mathcal{I}$ does not correspond to a counterexample in Case 2. Note that the proof of Case 2 does not rely on the assumption that $(\alpha-1)p(\mathcal{I}) \ge p_k$.

*Case* 3. The algorithm would have selected $J_i$ if given a chance, as $q_i \ge q_l > q_k$. Therefore, we must be in the situation of STEP 3 of D-LDT, which implies that $S_k(\sigma) + p_k \le r_i + (\alpha-1)p_i$. We obtain

$$L_{\max}(\pi) \ge \min_{j\in G(l)} r_j + \sum_{j\in G(l)} p_j + q_l = r_i + p(G(l)) + q_l.$$

Since $L_{\max}(\sigma) = C_l + q_l = S_k(\sigma) + p_k + p(G(l)) + q_l$, we deduce that

$$L_{\max}(\sigma) - L_{\max}(\pi) \le S_k(\sigma) + p_k - r_i \le r_i + (\alpha-1)p_i - r_i \le (\alpha-1)L_{\max}(\pi).$$

Since none of these three cases corresponds to a counterexample, we conclude that our assumption was wrong and hence that $J_k$ must have $p_k > (\alpha-1)p(\mathcal{I})$. ∎

COROLLARY 2.6. *$J_k = J_0$.* □

So far, we proved several structural properties that the heuristic schedule for a smallest counterexample must satisfy. Furthermore, we proved that the job that arrives first is big for all times $t$, has a smaller delivery time than $J_l$, and is the last job scheduled before $J_l$ with a smaller delivery time than $J_l$. Finally, none of the other jobs is big at any time $t$, and all jobs are scheduled contiguously. For our analysis in Theorem 2.8, we still need one more lemma that proves a specific property on the optimal schedule for the alleged smallest counterexample.

LEMMA 2.7. *Either $J_0$ is the first job in $\pi$, or $C_{\max}(\pi) \geq C_{\max}(\sigma) \geq \alpha p_0$.*

*Proof.* Let $J_1$ be the first job other than $J_0$ that becomes available. Since there are two jobs available at time $r_1$, the algorithm starts one of the jobs if the machine is still idle. Therefore, if $J_0$ is not the first job in $\pi$, the first job in $\sigma$ starts no later than the first job in $\pi$, and since there is no idle time in $\sigma$, we then have $C_{\max}(\sigma) \leq C_{\max}(\pi)$. It is easily checked that $C_{\max}(\sigma) \geq \alpha p_0$. □

THEOREM 2.8. *The deterministic on-line algorithm* D-LDT *for minimizing $L_{\max}$ on a single machine has performance bound $(\sqrt{5}+1)/2$.*

*Proof.* Suppose to the contrary that there exists an instance for which the algorithm finds a schedule $\sigma$ with $L_{\max}(\sigma) > \alpha L_{\max}(\pi)$. Obviously, then there exists a counterexample $\mathcal{I}$ with a minimum number of jobs. On basis of Observations 2.2 through 2.5, we may assume that the first job available in $\mathcal{I}$, which is defined to be $J_0$, has $p_0 > (\alpha - 1)p(\mathcal{I})$. Note that, due to Corollary 2.6, $J_0$ is the last job before $J_l$ in $\sigma$ with delivery time smaller than $q_l$.

Just like in the proof of Observation 2.5, we will show that $L_{\max}(\sigma) = C_l(\sigma) + q_l = S_k(\sigma) + p_k + p(G(l)) + q_l \leq \alpha L_{\max}(\pi)$. Again, we take a closer look at the situation in $\sigma$ at time $S_k(\sigma)$.

There are two possible reasons for starting $J_0$ instead of a job from $G(l)$ at time $S_0(\sigma)$. The first one is that simply none of the jobs in $G(l)$ was available, i.e., $r_j > S_0(\sigma)$ for all $J_l \in G(l)$. The second one is that the available jobs in $G(l)$ have a delivery time at most equal to $(\alpha - 1)p_0$ (the situation in STEP 2B). We cover both cases by distinguishing between

(1) $r_j > S_0(\sigma)$ for all $J_j \in G(l)$, and

(2) $q_j \leq (\alpha - 1)p_0$ for some $J_j \in G(l)$.

*Case* 1. Since none of the jobs in $G(l)$ is available at time $S_0(\sigma)$,

$$L_{\max}(\pi) > S_0(\sigma) + \sum_{j \in G(l)} p_j + q_l, \quad \text{and } L_{\max}(\sigma) = S_0(\sigma) + p_0 + \sum_{j \in G(l)} p_j + q_l.$$

Hence, $L_{\max}(\sigma) - L_{\max}(\pi) < p_0$. If $J_0$ is not the first job in $\pi$, then, according to Lemma 2.7, $L_{\max}(\pi) \geq C_{\max}(\pi) \geq \alpha p_0$, which implies that $L_{\max}(\sigma) - L_{\max}(\pi) < (\alpha - 1)L_{\max}(\pi)$. Therefore, we assume that $J_0$ is the first job in $\pi$. Moreover, we may assume that there is no job in $\mathcal{I}$ with delivery time greater than $(\alpha - 1)p_0$, as we would obtain $L_{\max}(\pi) \geq \alpha p_0$ then, from which we derive $p_0 \leq (\alpha - 1)L_{\max}(\pi)$. This implies that STEP 2A will not occur, and hence $J_0$ is started as soon as it is eligible, which event occurs at time $(\alpha - 1)p_0$ at the latest. All this leads to

$$L_{\max}(\pi) \geq p_0 + \sum_{j \in G(l)} p_j + q_l,$$

and hence, $L_{\max}(\sigma) - L_{\max}(\pi) \leq S_0(\sigma) \leq (\alpha - 1)p_0 \leq (\alpha - 1)L_{\max}(\pi)$, which ruins our counterexample.

*Case* 2. Since all jobs in $G(l)$ have a delivery time that is at least as large as $q_l$, we have that $q_l \leq (\alpha - 1)p_0$. This leads to the same situation as examined in Case 2 of Observation 2.5 for which we concluded that it was incapable of producing a counterexample.

Since we have eliminated all possibilities, we conclude that there is no counterexample to Theorem 2.8. $\square$

**Acknowledgments.** The authors want to thank Steve Seiden for pointing out a hole in the proof of Observation 2.5 and two anonymous referees for their constructive comments.

## REFERENCES

[1] B. Chen, A. van Vliet, and G. J. Woeginger (1994), *New lower and upper bounds for on-line scheduling*, Oper. Res. Lett., 16, pp. 221–230.

[2] R. L. Graham (1966), *Bounds for certain multiprocessing anomalies*, Bell System Tech. J., 45, pp. 1563–1581.

[3] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein (1997), *Scheduling to minimize average completion time: Off-line and on-line approximation algorithms*, Math. Oper. Res., 22, pp. 513–544.

[4] H. Kise, T. Iberaki, and H. Mine (1979), *Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times*, J. Oper. Res. Soc. Japan, 22, pp. 205–224.

[5] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys (1993), *Sequencing and scheduling: Algorithms and complexity*, in Logistics of Production and Inventory; Handbooks Oper. Res. Management Sci. 4, S. C. Graves, P. H. Zipkin, and A. H. G. Rinnooy Kan, eds., North-Holland, Amsterdam, pp. 445–522.

[6] R. Motwani, S. Phillips, and E. Torng (1994), *Non-clairvoyant scheduling*, Theoret. Comput. Sci., 130, pp. 17–47.

[7] C. Phillips, C. Stein, and J. Wein (1998), *Minimizing average completion time in the presence of release dates*, Math. Programming (B), 28, pp. 199–223.

[8] C. N. Potts (1980), *Analysis of a heuristic for one machine sequencing with release dates and delivery times*, Oper. Res. 28, pp. 1436–1441.

[9] S. S. Seiden (1998), *Randomized Online Scheduling with Delivery Times*, Report Woe-24, Technical University Graz, Graz, Austria.

[10] J. Sgall (1998), *On-line scheduling*, in A. Fiat and G. J. Woeginger, eds., On-Line Algorithms: The State of the Art, Lecture Notes in Comput. Sci. 14421, Springer, Berlin.

[11] D. B. Shmoys, J. Wein, and D. P. Williamson (1995), *Scheduling parallel machines on-line*, SIAM J. Comput. 24, pp. 1313–1331.

[12] A. P. A. Vestjens (1997), *On-line Machine Scheduling*, Ph.D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

SIAM J. Discrete Math.
Vol. 13, No. 1, pp. 64–78

# MULTIPROCESSOR SCHEDULING WITH REJECTION[*]

YAIR BARTAL[†], STEFANO LEONARDI[‡], ALBERTO MARCHETTI-SPACCAMELA[†], JIŘÍ SGALL[§], AND LEEN STOUGIE[¶]

**Abstract.** We consider a version of multiprocessor scheduling with the special feature that jobs may be rejected at a certain penalty. An instance of the problem is given by $m$ identical parallel machines and a set of $n$ jobs, with each job characterized by a processing time and a penalty. In the on-line version the jobs become available one by one and we have to schedule or reject a job before we have any information about future jobs. The objective is to minimize the makespan of the schedule for accepted jobs plus the sum of the penalties of rejected jobs.

The main result is a $1 + \phi \approx 2.618$ competitive algorithm for the on-line version of the problem, where $\phi$ is the golden ratio. A matching lower bound shows that this is the best possible algorithm working for all $m$. For fixed $m$ we give improved bounds; in particular, for $m = 2$ we give a $\phi \approx 1.618$ competitive algorithm, which is best possible.

For the off-line problem we present a fully polynomial approximation scheme for fixed $m$ and a polynomial approximation scheme for arbitrary $m$. Moreover, we present an approximation algorithm which runs in time $O(n \log n)$ for arbitrary $m$ and guarantees a $2 - \frac{1}{m}$ approximation ratio.

**Key words.** multiprocessor scheduling, on-line algorithms, competitive analysis, approximation algorithms

**AMS subject classification.** 68Q25

**PII.** S0895480196300522

## 1. Introduction.

Scheduling jobs on parallel machines is a classical problem that has been widely studied for more than three decades [6, 12]. In this paper we consider a version of the problem that has the special feature that jobs can be rejected at a certain price.

We call this problem *multiprocessor scheduling with rejection* and use the abbreviation MSR. Given are $m$ identical machines and $n$ jobs. Each job is characterized by its *processing time* and its *penalty*. A job can be either rejected, in which case its penalty is paid, or scheduled on one of the machines, in which case its processing time contributes to the completion time of that machine. The processing time is the same

[*]Received by the editors March 13, 1996; accepted for publication (in revised form) May 7, 1999; published electronically January 13, 2000. A preliminary version of this paper appeared in the *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, Atlanta, GA, January 28–30, 1996, SIAM, Philadelphia, 1996, pp. 95–103.
http://www.siam.org/journals/sidma/13-1/30052.html

[†]Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel (yairb@math.tau.ac.il). The research of this author was supported in part by the Ben Gurion Fellowship, Israel Ministry of Science and Arts.

[‡]Dipartimento di Informatica Sistemistica, Università di Roma "La Sapienza," via Salaria 113, 00198-Roma, Italia (leonardi@dis.uniroma1.it, marchetti@dis.uniroma1.it). The research of this author was partly supported by ESPRIT BRA Alcom II under contract 7141, and by Italian Ministry of Scientific Research Project 40% "Algoritmi, Modelli di Calcolo e Strutture Informative."

[§]Mathematical Institute, AS CR, Žitná 25, 115 67 Prague 1, Czech Republic, and Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic (sgall@math.cas.cz). The research of this author was partially supported by grants A1019602 and A1019901 of GA AV ČR, postdoctoral grant 201/97/P038 of GA ČR, and cooperative research grant INT-9600919/ME-103 from the NSF (USA) and the MŠMT (Czech Republic). Part of this work was done at the Institute of Computer Science, Hebrew University, Jerusalem, Israel and was supported in part by a Golda Meir Postgraduate Fellowship.

[¶]Department of Mathematics, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands (leen@win.tue.nl). The research of this author was supported by the Human Capital Mobility Network DONET of the European Community.

for all the machines, as they are identical. Preemption is not allowed; i.e., each job is assigned to a single machine and once started is processed without interruption. The objective is to minimize the *sum of the makespan and the penalties* of all rejected jobs. Makespan (the length of the schedule) is defined as the maximum completion time taken over all machines.

In the on-line version of MSR jobs become available one by one, and the decision to either reject a job or to schedule it on one of the machines has to be made before any information about following jobs is disclosed. In particular, there may be no other jobs. On-line algorithms are evaluated by the *competitive ratio*; an on-line algorithm is $c$-competitive if for each input the cost of the solution produced by the algorithm is at most $c$ times the cost of an optimal solution (cf. [14]).

The main goal of an on-line MSR algorithm is to choose the correct balance between the penalties of the jobs rejected and the increase in the makespan for the accepted jobs. At the beginning, it might have to reject some jobs if the penalty for their rejection is small compared to their processing time. However, at a certain point it would have been better to schedule some of the previously rejected jobs since the increase in the makespan due to scheduling those jobs in parallel is less than the total penalty incurred. In this scenario the on-line MSR problem can be seen as a nontrivial generalization of the well-known Rudolph's *ski rental problem* [11]. (In that problem, a skier has to choose whether to rent skis for the cost of 1 per trip or to buy them for the cost of $c$, without knowing the future number of trips. The best possible deterministic strategy is to rent for the first $c$ trips and buy afterwards. In our problem, rejecting jobs is analogous to renting, while scheduling one job is analogous to buying, as it allows us to schedule $m - 1$ more jobs of no bigger processing time without extra cost.)

Our main result is a best possible, $1 + \phi \approx 2.618$ competitive algorithm for the on-line MSR problem, where $\phi = (1 + \sqrt{5})/2$ is the golden ratio. We prove that no deterministic algorithm that receives $m$ as input can achieve a better competitive ratio independent of $m$.

For small values of $m$ we give better upper and lower bounds. In particular, for $m = 2$ we obtain a best possible, $\phi \approx 1.618$ competitive algorithm. For $m = 3$ we obtain 2-competitive algorithms and show a lower bound of 1.839.

Our results should be compared with the current knowledge about on-line algorithms for the classical multiprocessor scheduling problem. In that problem, each job has to be scheduled; hence it is equivalent to a special case of our problem where each penalty is larger than the corresponding processing time. Graham's *list scheduling* algorithm schedules each job on the currently least loaded machine and is $2 - \frac{1}{m}$ competitive [7]. It is known that for $m > 3$, list scheduling is not optimal [5], and in fact there exist $2 - \varepsilon$ competitive algorithms for small constant $\varepsilon > 0$ [2, 10, 1]. The best *possible* competitive ratio is known to be between 1.85 and 1.92 (see [1]), but its precise value is unknown. In contrast, for the more general on-line MSR problem we do find the optimal competitive ratio. More surprisingly, our algorithms achieving the optimal competitive ratio schedule the accepted jobs using list scheduling, which is inferior when rejections are not allowed!

Next we consider the off-line MSR problem. We present an approximation algorithm with a $2 - \frac{1}{m}$ worst-case approximation ratio running in time $O(n \log n)$ for arbitrary $m$. We also present a *fully polynomial approximation scheme* for MSR for any fixed $m$ and a *polynomial approximation scheme* for arbitrary $m$, i.e., where $m$ is part of the input.

More explicitly, the approximation schemes give algorithms with running time either polynomial in $n$ and $1/\epsilon$ but exponential in $m$, or polynomial in $n$ and $m$ but exponential in $1/\epsilon$, where $\epsilon$ is the maximal error allowed. This implies that for the more general problem with possible rejection of jobs we have algorithms that are essentially as good as those known for the classical problem without rejection. In fact, our algorithms are based on the techniques used for the problem without rejection, namely, on the fully polynomial approximation scheme for fixed $m$ [9] (based on a dynamic programming formulation of the problem) and the polynomial approximation scheme for arbitrary $m$ [8].

Obviously, the MSR problem on a single machine is easily solved exactly by scheduling every job whose processing time does not exceed its penalty, and for $m \geq 2$ it is NP-hard to find the optimal solution, similarly as in the classical case without rejections.

The on-line algorithms and lower bounds are presented in sections 3 and 4. Section 5 contains the results of the off-line problem.

**2. Notation.** An instance of the MSR problem consists of a *number of machines* $m$ and a *set of jobs* $J$, $|J| = n$. We abuse the notation and denote the $j$th job in the input sequence by $j$. Each job $j \in J$ is characterized by a pair $(p_j, w_j)$, where $p_j$ is its *processing time* and $w_j$ is its *penalty.*

For a set of jobs $X \subseteq J$, $W(X) = \sum_{j\in X} w_j$ is the total penalty of jobs in $X$, and $M(X) = \sum_{j\in X} p_j/m$ is the sum of the loads of the jobs in $X$, where the *load* of a job $j$ is defined by $p_j/m$. The set $B = \{j \mid w_j \leq p_j/m\}$ contains jobs with penalty less than or equal to their load.

Given a solution produced by an on-line or approximation algorithm, $R$ denotes the set of all rejected jobs, $A$ denotes the set of all accepted job, and $T$ denotes the largest processing time of all accepted jobs. For their analogues in the optimal solution we use $R^{OPT}$, $A^{OPT}$, $T^{OPT}$, respectively. $Z^{OPT}$ denotes the total cost of the optimal solution for a given instance of the problem, and $Z^H$ is the cost achieved by algorithm $H$. An on-line algorithm $ON$ is $c$-competitive if $Z^{ON} \leq c \cdot Z^{OPT}$ for every input instance.

The golden ratio is denoted by $\phi = (\sqrt{5}+1)/2 \approx 1.618$. We will often use the property of the golden ratio that $\phi - 1 = 1/\phi$.

Using list scheduling, the makespan of a schedule is bounded from above by the processing time of the job that finishes last plus the sum of the loads of all other scheduled jobs [7]. We denote this bound by $C^{LS}(X)$ for a set $X$ of scheduled jobs. If $\ell$ is the job in $X$ that finishes last, then

$$C^{LS}(X) = M(X - \{\ell\}) + p_\ell \leq M(X) + \left(1 - \frac{1}{m}\right) T. \tag{2.1}$$

**3. On-line scheduling with rejections.** In the first part of this section we present an on-line MSR algorithm which works for arbitrary $m$ and achieves the best possible competitive ratio in that case. The corresponding lower bound is given in section 4.2. For fixed $m \geq 3$ this algorithm gives the best competitive ratio we are able to achieve; however, we are not able to prove a matching lower bound. In the second part we present a different algorithm which is best possible for the case of two machines. The corresponding lower bound is given in section 4.1.

**3.1. Arbitrary number of machines.** Our algorithm uses two simple rules. First, all jobs in the set $B$ are rejected, which seems advantageous since their penalty is smaller than their load. The second rule is inspired by the relation of MSR to the ski rental problem and states that a job is rejected unless its penalty added to the total penalty of the hitherto rejected jobs would be higher than some prescribed fraction of its processing time. This fraction parameterizes the algorithm; we denote it by $\alpha$.

ALGORITHM RTP($\alpha$) (REJECT-TOTAL-PENALTY($\alpha$)).

(i) If a job from $B$ becomes available, reject it.

(ii) Let $W$ be the total penalty of all jobs from $J - B$ rejected so far. If a job $j = (p_j, w_j) \notin B$ becomes available, reject it if $W + w_j \leq \alpha p_j$, otherwise accept it and schedule it on a least loaded machine.

In Theorem 3.1 we will prove that for given $m$, the algorithm is $c$-competitive if $c$ and $\alpha > 0$ satisfy

$$c \geq 1 + \left(1 - \frac{1}{m}\right)\frac{1}{\alpha}, \qquad (3.1)$$
$$c \geq 2 + \alpha - \frac{2}{m}.$$

To obtain a best possible algorithm for arbitrary $m$, we use $\alpha = \phi - 1 \approx 0.618$. Then $c = 1 + \phi$ satisfies the inequalities above. For a fixed $m$, the best $c$ is obtained if equality is attained in both cases. For $m = 2$ this leads to $\alpha = \sqrt{2}/2 \approx 0.707$ and $c = 1 + \sqrt{2}/2 \approx 1.707$, and for $m = 3$ we get $\alpha = 2/3$ and $c = 2$. For general $m$ we obtain

$$\alpha = \frac{-(1 - \frac{2}{m}) + \sqrt{5 - \frac{8}{m} + \frac{4}{m^2}}}{2},$$
$$c = 1 + \frac{(1 - \frac{2}{m}) + \sqrt{5 - \frac{8}{m} + \frac{4}{m^2}}}{2}.$$

THEOREM 3.1. *The algorithm* RTP($\alpha$) *for $m$ machines is $c$-competitive if $c$ and $\alpha$ satisfy* (3.1).

*Proof.* First we notice that since our algorithm uses list scheduling for the accepted jobs, its makespan is bounded by $C^{LS}(A) = (1 - \frac{1}{m})T + M(A)$ (cf. (2.1)). Hence,

$$Z^{ON} \leq \left(1 - \frac{1}{m}\right)T + M(A) + W(R).$$

For any set $S \subseteq R$, the right-hand side of this inequality can be rewritten as a sum of two terms:

$$Z^{ON} \leq (M(A) + W(R - S) + M(S)) + \left(\left(1 - \frac{1}{m}\right)T + W(S) - M(S)\right). \qquad (3.2)$$

Now, we fix an off-line optimal solution. We use the above inequality for the set $S = (R - B) \cap A^{OPT}$, the set of all jobs rejected by the algorithm in step (ii) and accepted in the optimal solution. First, we bound the first term in (3.2). Notice that

$$M(A) = M(A \cap A^{OPT}) + M(A \cap R^{OPT}) \leq M(A \cap A^{OPT}) + W(A \cap R^{OPT}), \qquad (3.3)$$

since no job of the set $B$ is accepted by the algorithm, and thus the load of each job accepted by the algorithm is smaller than its penalty. Next we notice that $S \subseteq A^{OPT}$, implying that

$$M(S) = M(A^{OPT} \cap S). \qquad (3.4)$$

Since $R^{OPT}$ and $A^{OPT}$ is a partition of the set of all jobs, and $B \subseteq R$, we obtain

$$\begin{aligned} R - S &= [(R \cap R^{OPT}) \cup (R \cap A^{OPT})] - [(R - B) \cap A^{OPT}] \\ &= (R \cap R^{OPT}) \cup (B \cap A^{OPT}). \end{aligned} \tag{3.5}$$

From (3.5) and the definition of $B$ we have

$$W(R-S) = W(B \cap A^{OPT}) + W(R \cap R^{OPT}) \leq M(B \cap A^{OPT}) + W(R \cap R^{OPT}). \tag{3.6}$$

Inequalities (3.3), (3.4), and (3.6) together imply that

$$M(A) + W(R-S) + M(S) \leq M(A^{OPT}) + W(R^{OPT}) \leq Z^{OPT}.$$

To finish the proof, it is now sufficient to show

$$\left(1 - \frac{1}{m}\right) T + W(S) - M(S) \leq \left(1 + \alpha - \frac{2}{m}\right) T^{OPT} + \left(1 - \frac{1}{m}\right) \frac{1}{\alpha} W(R^{OPT}), \tag{3.7}$$

and notice that under our conditions (3.1) on $c$ this is at most

$$(c-1)T^{OPT} + (c-1)W(R^{OPT}) \leq (c-1)Z^{OPT}.$$

All jobs in $S$ are scheduled in the optimal solution and hence have processing time at most $T^{OPT}$. The algorithm never rejects such a job if this would increase the penalty above $\alpha T^{OPT}$, and hence

$$W(S) \leq \alpha T^{OPT}. \tag{3.8}$$

For any job $j$ that was rejected by step (ii) of the algorithm we have $w_j \leq \alpha p_j$. Summing over all jobs in $S$ we obtain $W(S) \leq \alpha m M(S)$, and hence

$$W(S) - M(S) \leq \left(1 - \frac{1}{\alpha m}\right) W(S) \leq \left(1 - \frac{1}{\alpha m}\right) \alpha T^{OPT} = \left(\alpha - \frac{1}{m}\right) T^{OPT}. \tag{3.9}$$

Thus, if $T \leq T^{OPT}$, (3.7) follows. If $T > T^{OPT}$, let $W$ be the penalty incurred by the jobs rejected in step (ii) of the algorithm until it schedules the first job with processing time $T$, job $j$ say, having penalty $w_j$. By the condition in step (ii) of the algorithm, $\alpha T \leq W + w_j$. Conversely, $W + w_j \leq W(S) + W(R^{OPT})$, as all jobs rejected in step (ii) are in $S \cup R^{OPT}$, and also the job with processing time $T$ is in $R^{OPT}$, since $T > T^{OPT}$. Thus,

$$\begin{aligned} \left(1 - \frac{1}{m}\right) T &\leq \left(1 - \frac{1}{m}\right) \frac{1}{\alpha} (W(S) + W(R^{OPT})) \\ &\leq \left(1 - \frac{1}{m}\right) T^{OPT} + \left(1 - \frac{1}{m}\right) \frac{1}{\alpha} W(R^{OPT}), \end{aligned} \tag{3.10}$$

using (3.8). Adding (3.9) to (3.10) we obtain (3.7), which finishes the proof. $\square$

Choosing $\alpha = \phi - 1$ and $c = \phi + 1$, both inequalities in (3.1) are satisfied for any $m$, which yields our main result. For arbitrarily large $m$ these values are the best possible.

THEOREM 3.2. *Algorithm* RTP$(\phi - 1)$ *is* $(1+\phi)$*-competitive.*

For any choice of $m$ and $\alpha$ the bounds on $c$ given by the inequalities (3.1) give a tight analysis of Algorithm RTP$(\alpha)$, as shown by the following two examples. First, consider the sequence of two jobs $(1-\frac{1}{\alpha m}, \alpha - \frac{1}{m})$ and $(1-\varepsilon, \frac{1}{m})$ with $\epsilon > 0$ arbitrarily small. RTP$(\alpha)$ rejects the first job and accepts the second job, while in the optimal solution both jobs are rejected. The competitive ratio attained on this sequence is $(1-\varepsilon+(\alpha-\frac{1}{m}))/\alpha$, which for any $\alpha > 0$ and $m$ can be made arbitrarily close to the first inequality of (3.1). Second, consider the sequence formed by one job $(1, \alpha)$, $m-2$ jobs $(1, \frac{1}{m})$, and one job $(1,1)$. RTP$(\alpha)$ rejects the first $m-1$ jobs and accepts job $(1,1)$, while the optimal solution accepts all jobs. The competitive ratio is $2+\alpha-\frac{2}{m}$, leading to the second inequality of (3.1).

**3.2. Two machines.** To obtain a best possible, $\phi$-competitive algorithm for two machines we use another approach. We simply reject all jobs with penalty at most $\alpha$ times their processing time, where $\alpha$ is again a parameter of the algorithm. Again the optimal value is $\alpha = \phi - 1 \approx 0.618$.

ALGORITHM RP$(\alpha)$ (REJECT-PENALTY$(\alpha)$). If a job $j = (p_j, w_j)$ becomes available, reject it if $w_j \leq \alpha p_j$, otherwise accept it and schedule it on a least loaded machine.

THEOREM 3.3. *The algorithm* RP$(\phi - 1)$ *is* $\phi$*-competitive for two machines.*

*Proof.* If the algorithm does not schedule any job, then

$$Z^{ON} = W(J) \leq 2(\phi-1)M(A^{OPT}) + W(R^{OPT}) \leq 2(\phi-1)Z^{OPT} \leq \phi Z^{OPT},$$

and the theorem is proved.

Otherwise denote by $\ell$ a job that is finished last by the on-line algorithm. Since the algorithm uses list scheduling, the makespan is bounded by $C^{LS}(A) = M(A - \{\ell\}) + p_\ell$, and therefore we have

$$Z^{ON} \leq W(R) + M(A - \{\ell\}) + p_\ell. \tag{3.11}$$

Notice that

$$W(R) = W(R \cap R^{OPT}) + W(R \cap A^{OPT}) \leq W(R \cap R^{OPT}) + 2(\phi-1)M(R \cap A^{OPT}) \tag{3.12}$$

by direct application of the rejection rule of algorithm RP$(\phi - 1)$.

For any job that is accepted by the algorithm, the rejection rule of RP$(\phi - 1)$ implies that its load is not greater than its penalty. Therefore,

$$\begin{aligned} M(A - \{\ell\}) &= M((A - \{\ell\}) \cap A^{OPT}) + M((A - \{\ell\}) \cap R^{OPT}) \\ &\leq M((A - \{\ell\}) \cap A^{OPT}) + W((A - \{\ell\}) \cap R^{OPT}). \end{aligned} \tag{3.13}$$

Invoking (3.12) and (3.13) in (3.11) yields

$$Z^{ON} \leq W(R^{OPT} - \{\ell\}) + 2(\phi-1)M(A^{OPT} - \{\ell\}) + p_\ell. \tag{3.14}$$

We distinguish two cases. In the first case the optimal solution rejects job $\ell$. Since $\ell$ is scheduled by the algorithm, we have $p_\ell \leq \phi w_\ell$, and therefore

$$Z^{ON} \leq \phi W(R^{OPT}) + 2(\phi-1)M(A^{OPT}) \leq \phi Z^{OPT}.$$

In the second case $\ell$ is accepted in the optimal solution. Then, we use the identity $p_\ell = 2(\phi-1)M(\{\ell\}) + (1-(\phi-1))p_\ell$ in (3.14) to obtain

$$\begin{aligned} Z^{ON} &\leq (2-\phi)p_\ell + W(R^{OPT}) + 2(\phi-1)M(A^{OPT}) \\ &\leq (2-\phi)Z^{OPT} + 2(\phi-1)Z^{OPT} \; = \; \phi Z^{OPT}, \end{aligned}$$

which completes the proof. □

The same approach can be used for larger $m$ as well. However, for $m > 3$ this is worse than the previous algorithm. An interesting situation arises for $m = 3$. Choosing $\alpha = 1/2$ we obtain a 2-competitive algorithm, which matches the competitive ratio of the algorithm RTP(2/3) for $m = 3$ in the previous subsection. Whereas RP(1/2) rejects all jobs with penalty up to 1/2 of their processing time, RTP(2/3) rejects all jobs with penalty up to 1/3 of their processing time and also jobs with larger penalty as long as the total penalty paid (by the jobs with smaller or equal processing times) remains at most 2/3 times the processing time. We can combine these two approaches and show that for any $1/3 \leq \alpha \leq 1/2$, the algorithm that rejects each job with penalty at most $\alpha$ times its *processing time*, and also if the total penalty is up to $1-\alpha$ times its processing time, is 2-competitive, too. However, no such combined algorithm is better.

## 4. Lower bounds for on-line algorithms.

In the first part of this section we give the lower bound for a small number of machines. In particular it shows that the algorithm presented in section 3.2 is best possible for $m = 2$. In the second part we exhibit the lower bound for algorithms working for all $m$.

### 4.1. Small number of machines.

Assume that there exists a $c$-competitive on-line algorithm for $m$ machines. We prove that $c$ satisfies $c \geq \rho$, where $\rho$ is the solution of the following equation:

$$\rho^{m-1} + \rho^{m-2} + \cdots + 1 = \rho^m. \tag{4.1}$$

For $m = 2$ we get $\rho = \phi$, and hence prove that the algorithm RP$(\phi - 1)$ is best possible. For $m = 3$ we get $\rho \approx 1.839$, and so on. Notice that for arbitrary $m$ this proves only that the competitive ratio is at least 2.

THEOREM 4.1. *For any $c$-competitive algorithm for MSR on $m$ machines, it holds that $c \geq \rho$, where $\rho$ satisfies* (4.1).

*Proof.* Given $m$, let $\rho$ be the solution of (4.1). Consider an adversary providing a sequence of jobs, all with processing time 1. The first job given has penalty $w_1 = 1/\rho$. If the on-line algorithm accepts this job, the sequence stops and the algorithm is $\rho$-competitive. Otherwise, a second job is given by the adversary with penalty $w_2 = 1/\rho^2$. Again, accepting this job by the on-line algorithm makes the sequence stop and the competitive ratio is $\rho$. Rejection makes the sequence continue with a third job. This process is repeated for at most $m-1$ jobs with penalties $w_j = 1/\rho^j$ for $1 \leq j = m-1$. If the on-line algorithm accepts any job in this sequence, job $k$ say, the adversary stops the sequence at that job, yielding a competitive ratio of the on-line algorithm on this sequence of $k$ jobs of

$$\frac{Z^{ON}}{Z^{OPT}} = \frac{1 + \sum_{j=1}^{k-1} \frac{1}{\rho^j}}{\sum_{j=1}^{k} \frac{1}{\rho^j}} = \rho,$$

since for any such $k \leq m-1$ in the optimal solution all jobs are rejected.

Otherwise, if none of the first $m-1$ jobs are accepted by the on-line algorithm, another job is presented with penalty $w_m = 1$. In the optimal solution all $m$ jobs are accepted and scheduled in parallel, giving cost 1. The on-line cost is equal to the sum of the penalties of the first $m-1$ jobs plus 1, independent of whether the last job is accepted or rejected. Thus,

$$\frac{Z^{ON}}{Z^{OPT}} = 1 + \sum_{j=1}^{m-1} \frac{1}{\rho^j}.$$

By (4.1), this is exactly $\rho$, and the theorem follows. $\square$

COROLLARY 4.2. *For two machines, no on-line algorithm has competitive ratio less than $\phi$.*

**4.2. Arbitrary number of machines.** Now we prove the lower bound on algorithms working for arbitrary $m$. The sequence of jobs starts as in the previous section, but additional ideas are necessary.

THEOREM 4.3. *There exists no on-line algorithm that is $\beta$-competitive for some constant $\beta < 1 + \phi$ and all $m$.*

*Proof.* All jobs in the proof have processing time 1. All logarithms are base 2. For contradiction, we assume that the on-line algorithm is $\beta$-competitive for a constant $\beta < 1 + \phi$, and $m$ is a sufficiently large power of two. Let $a_i = (\log m)^{i+1}$, and let $k$ be the largest integer such that $\log m + \sum_{i=1}^{k} a_i < m$. Calculation gives $k = \lfloor \log m / \log\log m \rfloor - 1$.

Consider again an adversary that intends to provide the following sequence of at most $m$ jobs (all with processing time 1):

$$\begin{array}{rll}
1 & \text{job with penalty} & 1/(1+\phi), \\
1 & \text{job with penalty} & 1/(1+\phi)^2, \\
 & \vdots & \\
1 & \text{job with penalty} & 1/(1+\phi)^{\log m}, \\
a_1 & \text{jobs with penalty} & 1/a_1, \\
 & \vdots & \\
a_k & \text{jobs with penalty} & 1/a_k.
\end{array}$$

As in the proof of Theorem 4.1 we argue that if the on-line algorithm accepts one of the first $\log m$ jobs, the adversary stops the sequence and the competitive ratio is $1 + \phi$. Therefore, any $\beta$-competitive algorithm has to reject the first $\log m$ jobs. Now, let $b_i$ be the number of jobs with penalty $1/a_i$ that are rejected by the $\beta$-competitive algorithm. The penalty the algorithm pays on those jobs is $b_i/a_i$. Since there are less than $m$ jobs, the optimal cost is at most 1. Thus the total penalty incurred by the on-line algorithm has to be at most $\beta$, and in particular there has to exist $\ell \le k$ such that $b_\ell / a_\ell \le \beta/k < 3/k$. Fix such $\ell$.

Now consider the following modified sequence of at most $2m$ jobs (again all with processing time 1):

$$\begin{array}{rll}
1 & \text{job with penalty} & 1/(1+\phi), \\
1 & \text{job with penalty} & 1/(1+\phi)^2, \\
 & \vdots & \\
1 & \text{job with penalty} & 1/(1+\phi)^{\log m}, \\
a_1 & \text{jobs with penalty} & 1/a_1, \\
 & \vdots & \\
a_\ell & \text{jobs with penalty} & 1/a_\ell, \\
M & \text{jobs with penalty} & 6,
\end{array}$$

where $M = m + 1 - \sum_{i=1}^{\ell}(a_i - b_i)$.

The sequence is identical up to the jobs with penalty $1/a_\ell$, and hence the on-line algorithm behaves identically on this initial subsequence. In particular, it also rejects all first $\log m$ jobs paying a penalty of at least $\sum_{j=1}^{\log m}(1+\phi)^{-j} = (1-(1+\phi)^{-\log m})/\phi \geq \phi - 1 - 1/m$ for them. Then it also rejects $b_i$ jobs with penalty $1/a_i$, for $i \leq \ell$, paying penalty $\sum_{i=1}^{\ell} b_i/a_i$ for them.

The on-line algorithm has to accept all jobs with penalty 6, since the adversary will present at most $2m$ jobs, and hence scheduling them all would lead to a cost of at most 2. By summing the numbers, it follows that the on-line algorithm schedules exactly $m+1$ jobs. Thus, its makespan is at least 2, and its total cost is at least $1 + \phi - 1/m$.

To finish the proof, it is sufficient to present a solution with cost $1+o(1)$. Consider the solution that rejects

$$\begin{array}{rll}
1+\log m & \text{jobs with penalty} & 1/a_1, \\
b_1 & \text{jobs with penalty} & 1/a_2, \\
b_2 & \text{jobs with penalty} & 1/a_3, \\
 & \vdots & \\
b_{\ell-2} & \text{jobs with penalty} & 1/a_{\ell-1}, \\
b_{\ell-1} + b_\ell & \text{jobs with penalty} & 1/a_\ell
\end{array}$$

and schedules all remaining jobs optimally. First we verify that this description is legal, i.e., there are always sufficiently many jobs with given penalty. By definition, $b_i \leq a_i \leq a_{i+1}$. For sufficiently large $m$, we have $1+\log m < a_1$, and due to our choice of $\ell$, we also have $b_{\ell-1} + b_\ell \leq a_{\ell-1} + 3a_\ell/k \leq a_\ell$.

In the presented schedule one more job is rejected than in the solution produced by the on-line algorithm, and hence there are only $m$ jobs to be scheduled. Thus, the makespan is 1. The penalty paid is

$$\frac{1+\log m}{a_1} + \sum_{i=1}^{\ell-1} \frac{b_i}{a_{i+1}} + \frac{b_\ell}{a_\ell} = \frac{1+\log m}{(\log m)^2} + \frac{1}{\log m}\sum_{i=1}^{\ell-1}\frac{b_i}{a_i} + \frac{b_\ell}{a_\ell}.$$

The sum in the second term is less than the penalty paid by the on-line algorithm, and hence this term is bounded by $O(1/\log m)$. The last term is bounded due to our choice of $\ell$; namely, it is $O(1/k) = O(\log\log m/\log m)$. Thus, the total penalty paid is $O(\log\log m/\log m) = o(1)$, and the total cost is $1 + o(1)$. $\square$

## 5. Off-line scheduling with rejection.

**5.1. An approximation algorithm for arbitrary number of machines.** In this section we give a $(2-\frac{1}{m})$-approximation algorithm for MSR on $m$ machines. Our lower bounds imply that such a ratio cannot be achieved by an on-line algorithm. The algorithm rejects all jobs in the set $B = \{j \mid w_j \leq p_j/m\}$. From all other jobs it accepts some number of jobs with the smallest processing time and chooses the best among such solutions.

ALGORITHM APPROX.

(i) Sort all jobs in $J - B$ according to their processing times in nondecreasing order.

(ii) Let $S_i$, $0 \leq i \leq |J-B|$, be the solution that schedules the first $i$ jobs from $J - B$ using list scheduling and rejects all other jobs. Choose the solution $S_i$ with the smallest cost.

Note that step (ii) of the algorithm takes time $O(n \log m)$ (or $O(n)$ in case $m \geq n$), as we can build the schedules incrementally, and the bookkeeping of penalties for rejected jobs is simple. Thus, the whole algorithm runs in time $O(n \log n)$, independent of $m$. A performance analysis leads to the following worst-case ratio.

THEOREM 5.1. *Algorithm APPROX achieves $Z^H \leq (2-\frac{1}{m})Z^{OPT}$, where $Z^H$ is the cost of the solution found by the algorithm.*

*Proof.* We assume that the jobs from $J-B$ are ordered $1, 2, \ldots, |J-B|$, according to the ordering given by step (i) of the algorithm. If the optimal solution rejects all jobs from $J - B$, by the definition of $B$ it is optimal to reject all jobs from $B$ as well. Thus the solution $S_0$ that rejects all jobs is optimal and $Z^H = Z^{OPT}$.

Otherwise let $\ell$ be the last job from $J - B$ accepted in the optimal solution. Consider the solution $S_\ell$, which schedules all jobs up to $\ell$. Let $A = \{1, \ldots, \ell\}$ be the set of all jobs scheduled in $S_\ell$. Job $\ell$ has the largest running time of all scheduled jobs, and since we use list scheduling, the makespan of $S_\ell$ is at most

$$C^{LS}(A) = M(A) + \left(1 - \frac{1}{m}\right) p_\ell \leq M(A) + \left(1 - \frac{1}{m}\right) Z^{OPT}.$$

Since the cost of the algorithm is at most the cost of $S_\ell$, we have

$$\begin{aligned} Z^H &\leq W(J-A) + M(A) + \left(1 - \frac{1}{m}\right) Z^{OPT} \\ &= W(A^{OPT} \cap (J-A)) + W(R^{OPT} \cap (J-A)) + \\ &\quad M(R^{OPT} \cap A) + M(A^{OPT} \cap A) + \left(1 - \frac{1}{m}\right) Z^{OPT}. \end{aligned}$$

By the choice of $\ell$, $A^{OPT} \cap (J-A) \subseteq B$, and thus $W(A^{OPT} \cap (J-A)) \leq M(A^{OPT} \cap (J-A))$. Moreover, since $A$ does not contain any job of $B$, $M(R^{OPT} \cap A) \leq W(R^{OPT} \cap A)$. These observations inserted in the above inequality yield

$$Z^H \leq W(R^{OPT}) + M(A^{OPT}) + \left(1 - \frac{1}{m}\right) Z^{OPT} \leq \left(2 - \frac{1}{m}\right) Z^{OPT}. \qquad \square$$

That the ratio is tight is shown by the following instance with $m$ jobs (and $m$ machines): $p_1 = \cdots = p_m = 1$, $w_1 = 1 - \epsilon$, and $w_2 = \cdots = w_m = \frac{1}{m}(1-\epsilon)$. The heuristic will reject all jobs resulting in $Z^H = (1 + \frac{m-1}{m})(1-\epsilon)$. In the optimal solution all jobs are accepted; hence $Z^{OPT} = 1$. Therefore, $Z^H/Z^{OPT}$ can be made arbitrarily close to $2 - \frac{1}{m}$.

This example also shows that any heuristic that rejects all jobs in the set $B$ has a worst-case ratio no better than $2 - \frac{1}{m}$, since there is no scheduling at all involved in it. Thus, the only way in which an improvement might be obtained is by also possibly accepting jobs in the set $B$.

**5.2. A fully polynomial approximation scheme for fixed *m*.** For the off-line MSR problem there exists a fully polynomial approximation scheme for fixed $m$. The proof uses a rounding technique based on dynamic programming, as was developed in [9] for the classical makespan problem.

LEMMA 5.2. *The MSR problem with integer processing times and penalties can be solved in time polynomial in n and* $(Z^{OPT})^m$.

*Proof.* We use dynamic programming. Let $M_i$ represent the current load of machine $i$, $i = 1, \ldots, m$. We compute for each $M_1, \ldots, M_m \leq Z^{OPT}$ the minimal value of total penalty to be paid that can be achieved with these loads. We denote this value after the first $j$ jobs are rejected or scheduled by $W_j(M_1, \ldots, M_m)$ and define it to be $\infty$ whenever $M_i < 0$ for some $i$. At the same time we compute the minimal cost of a schedule that can be achieved with given loads $M_1, \ldots, M_m$, denoted $Z(M_1, \ldots, M_m)$. For $M_1, \ldots, M_m \geq 0$ these values can be computed recursively as follows:

$$\begin{aligned}
W_0(M_1, \ldots, M_m) &= 0, \\
W_j(M_1, \ldots, M_m) &= \min\{\, w_j + W_{j-1}(M_1, \ldots, M_m), \\
&\qquad \min_i W_{j-1}(M_1, \ldots, M_{i-1}, M_i - p_j, M_{i+1}, \ldots, M_m)\}, \\
Z(M_1, \ldots, M_m) &= W_n(M_1, \ldots, M_m) + \max_i M_i.
\end{aligned}$$

We compute the values in the order of increasing $\max_i M_i$. As soon as $\max_i M_i$ reaches the cost of the current optimal solution, which is the smallest value of $Z$ computed so far, we stop, as we know it is a global optimum. ∎

THEOREM 5.3. *For any* $\varepsilon \geq 0$, *there exists an* $\varepsilon$*-approximation algorithm for the MSR problem that runs in time polynomial in the size of the input instance,* $n^m$ *and* $1/\varepsilon^m$.

*Proof.* Given an instance $I$ of the MSR problem with $n$ jobs and $m$ machines, we first use the approximation algorithm from section 5.1 to obtain the cost $Z^H$. Now we define an instance $I'$ by rounding the processing times and the penalties of the jobs in $I$. Namely, the processing time $p'_j$ and the penalty $w'_j$ of job $j$ in $I'$ are $p'_j = \lfloor p_j/k \rfloor$ and $w'_j = \lfloor w_j/k \rfloor$, where $k = \varepsilon Z^H/2n$. We obtain the optimal solution of $I'$ by the dynamic programming algorithm presented in the proof of Lemma 5.2 and derive an approximate solution for $I$ by scheduling the respective jobs on the same machines as in the optimal solution for $I'$.

The cost $Z^{A(k)}$ of the approximate solution deviates from the optimal solution for $I$ by at most $nk = \varepsilon Z^H/2$. Therefore, by applying the lower bound $Z^{OPT} \geq Z^H/2$ we obtain

$$\frac{|Z^{A(k)} - Z^{OPT}|}{Z^{OPT}} \leq \frac{2nk}{Z^H} = \varepsilon.$$

By Lemma 5.2 it follows that the running time of the approximation algorithm is polynomial in $n$ and $(Z^{OPT}(I'))^m$. The theorem follows since $Z^{OPT}(I') \leq Z^{OPT}(I)/k \leq 2Z^H/k$, and hence $Z^{OPT}(I') \leq 4n/\varepsilon$. ∎

**5.3. A polynomial approximation scheme for arbitrary $m$.** For arbitrary $m$ we will design a polynomial approximation scheme (PAS) based on the PAS for the makespan problem in [8].

Given an instance with $n$ jobs, $m$ machines, and $\epsilon > 0$, we are to find an $\epsilon$-approximate solution. As an upper bound $U$ on the solution value we use the outcome $Z^H$ of the heuristic presented in section 5.1. Notice that all jobs with $p_j > U$ will be rejected. Thus, all jobs that are possibly scheduled have processing times in the interval $[0, U]$.

From Theorem 5.1 we have a lower bound on the optimal solution that we denote by $L = Z^H/2 = U/2$. We define the set $S = \{j \mid p_j \in [0, \epsilon L/3]\}$, a set of jobs with relatively small processing times. Let $D = \{j \mid j \notin S\}$. The remaining interval $(\epsilon L/3, U]$ is partitioned into $s \leq 18\lceil 1/\epsilon^2 \rceil$ subintervals $(l_1, l_2]$, $(l_2, l_3]$, $\ldots$, $(l_s, l_{s+1}]$ of length $\epsilon^2 L/9$ each, with $l_1 = \epsilon L/3$ and $l_{s+1} \geq U$. Let $D_i$ be the set of jobs with processing time in the interval $(l_i, l_{i+1}]$, and let the jobs in each such set be ordered so that the penalties are nonincreasing. As before, define the set $B = \{j \mid w_j \leq p_j/m\}$.

First we will describe how, for any subset $\Delta$ of $D$, we generate an approximate solution with value $Z^{H(\epsilon)}(\Delta)$. For any such set $\Delta$ we determine a schedule for all the jobs in $\Delta$ with an $\epsilon/3$-approximate makespan using the PAS in [8]. All other jobs in $D$, i.e., all jobs in $D - \Delta$, are rejected. Jobs in the set $S$ that have $w_j \geq \frac{1}{m} p_j$, i.e., jobs in the set $S - B$, are scheduled in any order according to the list scheduling rule starting from the $\epsilon/3$-approximate schedule determined before. The remaining jobs, $j \in S \cap B$, are considered in any order. Each next job is rejected if its assignment to a least loaded machine would cause an increase of the makespan; otherwise it is assigned to a least loaded machine as indicated by list scheduling.

This procedure is applied to every set $D(y_1, \ldots, y_s) \subseteq D$, where $D(y_1, \ldots, y_s)$ denotes the set that is composed of the first $y_i$ elements in the ordered set $D_i$, $i = 1, \ldots, s$. In this way an approximate solution $Z^{H(\epsilon)}(D(y_1, \ldots, y_s))$ is found for each set $D(y_1, \ldots, y_s)$. The minimum value over all these sets,

$$Z^{H(\epsilon)} = \min_{(y_1, \ldots, y_s)} Z^{H(\epsilon)}(D(y_1, \ldots, y_s)),$$

is taken as the output of our procedure.

THEOREM 5.4. *For any $\epsilon > 0$ the algorithm $H(\epsilon)$ described above runs in time polynomial in $n$ and $m$ and yields*

$$\frac{Z^{H(\epsilon)}}{Z^{OPT}} \leq 1 + \epsilon.$$

*Proof.* The proof consists of two steps. First, consider the set $A^{OPT} \cap D$ of jobs in $D$ that are accepted in the optimal solution. Applying the heuristic procedure described above to this set of jobs yields the approximate solution $Z^{H(\epsilon)}(A^{OPT} \cap D)$. We will prove that

$$\frac{Z^{H(\epsilon)}(A^{OPT} \cap D)}{Z^{OPT}} \leq 1 + \frac{\epsilon}{3}. \tag{5.1}$$

In the second step we analyze how much the set $A^{OPT} \cap D$ may differ from $D(y_1, \ldots, y_s)$. Assume that for $i = 1, \ldots, s$, $A^{OPT} \cap D$ consists of $y_i^{OPT}$ jobs from the set $D_i$. These $y_i^{OPT}$ jobs are not necessarily the first $y_i^{OPT}$ jobs in the ordered set $D_i$, but we will show that

$$Z^{H(\epsilon)}(D(y_1^{OPT}, \ldots, y_s^{OPT})) \leq Z^{H(\epsilon)}(A^{OPT} \cap D) + \frac{2}{3}\epsilon L. \tag{5.2}$$

Inequalities (5.1) and (5.2) imply that

$$\frac{Z^{H(\epsilon)}(D(y_1^{OPT},\dots,y_s^{OPT}))}{Z^{OPT}} \le 1+\epsilon.$$

Since, obviously, $Z^{H(\epsilon)} \le Z^{H(\epsilon)}(D(y_1^{OPT},\dots,y_s^{OPT}))$, the theorem follows.

In order to prove inequality (5.1) two cases are distinguished.

(1) The completion times of the various machines (in the heuristic solution corresponding to $Z^{H(\epsilon)}(A^{OPT}\cap D)$) differ by no more than $\epsilon L/3$. The resulting makespan is the same as the makespan after scheduling the jobs in $A^{OPT}\cap D$ and $S-B$, and due to our assumption it is at most $M(A^{OPT}\cap D)+M(S-B)+\epsilon L/3$. The weight of all rejected jobs is at most $W(S\cap B)+W(D-A^{OPT})$. Thus

$$Z^{H(\epsilon)}(A^{OPT}\cap D) \le M(A^{OPT}\cap D)+M(S-B)+\frac{\epsilon L}{3}+W(S\cap B)+W(D-A^{OPT}).$$

Using the definition of the set $B$, we have for the optimal solution

$$Z^{OPT} \ge M(A^{OPT}\cap D)+M(S-B)+W(S\cap B)+W(D-A^{OPT}).$$

From these two inequalities (5.1) follows immediately.

(2) The completion times of the machines differ by more than $\epsilon L/3$. Since the processing time of each job in $S$ is less than $\epsilon L/3$, we know that no job in the set $S\cap B$ is rejected, and scheduling all jobs in $S$ has not increased the makespan computed for the set $A^{OPT}\cap D$. Let $C^{H(\epsilon)}(A^{OPT}\cap D)$ and $C^{OPT}(A^{OPT}\cap D)$ denote, respectively, the $\epsilon/3$-approximate and the optimal makespan for the jobs in $A^{OPT}\cap D$. In this case

$$Z^{H(\epsilon)}(A^{OPT}\cap D) = C^{H(\epsilon)}(A^{OPT}\cap D)+W(D-A^{OPT})$$

and

$$Z^{OPT} \ge C^{OPT}(A^{OPT}\cap D)+W(D-A^{OPT}).$$

Moreover, since we have used an $\epsilon/3$-approximate algorithm for scheduling the jobs in $A^{OPT}\cap D$, we have

$$C^{H(\epsilon)}(A^{OPT}\cap D) \le \left(1+\frac{\epsilon}{3}\right) C^{OPT}(A^{OPT}\cap D).$$

Inequality (5.1) results from the above three inequalities.

In order to prove (5.2) we need to bound the extra error that might occur due to the fact that $A^{OPT}\cap D \ne D(y_1^{OPT},\dots,y_s^{OPT})$. Notice that, for any $D_i$, $i=1,\dots,s$, the difference in processing time between any two jobs in $D_i$ is at most $\epsilon^2 L/9$, and that $D(y_1^{OPT},\dots,y_s^{OPT})$ contains the jobs with larger penalties in $D_i$. The latter implies that the extra error can be due only to the fact that the first $y_i^{OPT}$ jobs in $D_i$ have longer processing times than those in $A^{OPT}\cap D_i$. Since the processing time of a job in $D$ is at least $\epsilon L/3$ and $U \le 2L$, no more than $6/\epsilon$ jobs from $D$ are scheduled on any machine. Therefore the overall extra contribution to the makespan due to the fact that $A^{OPT}\cap D \ne D(y_1^{OPT},\dots,y_s^{OPT})$ can be no more than $(6/\epsilon)(\epsilon^2 L/9) = 2\epsilon L/3$, which implies inequality (5.2).

This completes the proof of correctness of the approximation.

The running time of the algorithm is dominated by the time required to compute the heuristic $Z^{H(\epsilon)}(D(y_1,\dots,y_s))$ for each possible set of values $y_1,\dots,y_s$, such that

$0 \leq y_i \leq |D_i|$, $i = 1, \dots, s$. Since $y_i$, $i = 1, \dots, s$, satisfies $1 \leq y_i \leq n$, there are at most $n^s = O(n^{18\lceil 1/\epsilon^2 \rceil})$ possible sets of values $y_1, \dots, y_s$.

For each of these sets an $\epsilon$-approximate schedule is computed using the algorithm in [8], taking $O((n/\epsilon)^{\lceil 9/\epsilon^2 \rceil})$; attaching the jobs in the set $S$ just adds $O(n^2)$ time to each of these computations. Hence, the overall running time of the algorithm is $O((n^3/\epsilon)^{\lceil 9/\epsilon^2 \rceil})$. This establishes that the algorithm is a polynomial approximation scheme for the problem with arbitrary $m$. $\square$

**6. Open problems and recent developments.** Some open problems remain. For the on-line problem tight algorithms for the case of fixed $m$ other than $m = 2$ are still to be established. For the off-line problem perhaps better heuristics may be found by improving the rejection strategy proposed in the algorithm in section 5.1.

Seiden [13] has proved new results related to our problem. For the variant of deterministic *preemptive* scheduling with rejection he gives a $(4 + \sqrt{10})/3 \approx 2.387$ competitive algorithm for any number of machines, thus showing that allowing preemption can provably be exploited. Interestingly, this yields yet another 2-competitive algorithm for three machines. Also, Seiden notes that our Theorem 4.1 yields a lower bound for preemptive scheduling as well and hence yields a lower bound of 2 for general number of machines. For two machines, this shows that our algorithm RP$(\phi - 1)$ is best possible even among all preemptive algorithms. For three machines, an interesting open problem is to establish whether preemption allows a better competitive ratio. The best upper bound of 2 and the best lower bound of 1.839 for preemptive algorithms still coincide with those shown in this paper for nonpreemptive algorithms.

Seiden [13] also studies randomized scheduling with rejection, both preemptive and nonpreemptive. He gives algorithms which are better than deterministic for a small number of machines, and in particular are 1.5-competitive for two machines, both preemptive and nonpreemptive; this is best possible for two machines. In both cases the question of whether randomized algorithms for any number of machines can be better than their deterministic counterparts remains open.

Epstein and Sgall [4] presented polynomial time approximation schemes for related machines for various objectives, including MSR, thus generalizing the polynomial time approximation scheme given in this paper.

Engels et al. [3] study scheduling with rejection where, in the objective, the makespan is replaced by the sum of the completions times.

**Acknowledgments.** We thank Giorgio Gallo for having drawn our attention to this scheduling problem. We thank the anonymous referees for numerous helpful comments.

## REFERENCES

[1] S. Albers, *Better bounds for online scheduling*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, ACM, New York, 1997, pp. 130–139.

[2] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra, *New algorithms for an ancient scheduling problem*, J. Comput. Systems Sci., 51 (1995), pp. 359–366.

[3] D. W. Engels, D. R. Karger, S. G. Kolliopoulos, S. Sengupta, R. N. Uma, and J. Wein, *Techniques for scheduling with rejection*, in Proceedings of the 6th Annual European Symposium on Algorithms, Lecture Notes in Comput. Sci. 1461, Springer-Verlag, New York, 1998, pp. 490–501.

[4] L. Epstein and J. Sgall, *Approximation Schemes for Scheduling on Uniformly Related and Identical Parallel Machines*, Technical Report KAM-DIMATIA Series 98-414, Charles University, Prague, Czech Republic, 1998.

[5] G. Galambos and G. J. Woeginger, *An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling*, SIAM J. Comput., 22 (1993), pp. 349–355.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP–completeness*, W. H. Freeman, San Francisco, CA, 1979.

[7] R. L. Graham, *Bounds for certain multiprocessor anomalies*, Bell System Tech., 45 (1966), pp. 1563–1581.

[8] D. S. Hochbaum and D. B. Shmoys, *Using dual approximation algorithms for scheduling problems: Theoretical and practical results*, J. Assoc. Comput. Mach., 34 (1987), pp. 144–162.

[9] E. Horowitz and S. Sahni, *Exact and approximate algorithms for scheduling non-identical processors*, J. Assoc. Comput. Mach., 23 (1976), pp. 317–327.

[10] D. R. Karger, S. J. Phillips, and E. Torng, *A better algorithm for an ancient scheduling problem*, J. Algorithms, 20 (1996), pp. 400–430.

[11] R. M. Karp, *On-line algorithms versus off-line algorithms: How much is it worth to know the future?*, in Proceedings of the IFIP 12th World Computer Congress. Vol. 1: Algorithms, Software, Architecture, J. van Leeuwen, ed., Elsevier Science Publishers, Amsterdam, 1992, pp. 416–429.

[12] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *Sequencing and scheduling: Algorithms and complexity*, in Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory, S. C. Graves, A. H. G. Rinnooy Kan, and P. Zipkin, eds., North–Holland, Amsterdam, 1993, pp. 445–552.

[13] S. S. Seiden, *More Multiprocessor Scheduling with Rejection*, Technical Report Woe-16, Department of Mathematics, TU Graz, Graz, Austria, 1997.

[14] D. D. Sleator and R. E. Tarjan, *Amortized efficiency of list update and paging rules*, Comm. Assoc. Comput. Mach., 28 (1985), pp. 202–208.

SIAM J. DISCRETE MATH.
Vol. 13, No. 1, pp. 79–104

# THE WEIGHT HIERARCHY OF HERMITIAN CODES[*]

ANGELA I. BARBERO[†] AND CARLOS MUNUERA[‡]

**Abstract.** We compute the complete weight hierarchies of all Hermitian codes. The tools used are the arithmetic of Hermitian curves and the order bound on the generalized Hamming weights.

**Key words.** linear codes, generalized Hamming weights, Hermitian codes

**AMS subject classification.** 94B27

**PII.** S089548019834342X

**1. Introduction.** Let $\mathbf{F}_q$ be the finite field with $q$ elements. The *support* of a linear code $\mathcal{C}$ over $\mathbf{F}_q$ is defined by

$$\operatorname{supp}(\mathcal{C}) = \{i \mid x_i \neq 0 \text{ for some } \mathbf{x} \in \mathcal{C}\}.$$

If $\mathcal{C}$ is an $[n,k]$ code, for $1 \leq r \leq k$, the $r$th *generalized Hamming weight* of $\mathcal{C}$ is defined by

$$d_r(\mathcal{C}) = \min\{\#\operatorname{supp}(D) \mid D \text{ is a linear subcode of } \mathcal{C} \text{ with } \dim(D) = r\}.$$

The sequence $d_1(\mathcal{C}), \dots, d_k(\mathcal{C})$, of generalized Hamming weights, or *weight hierarchy* of $\mathcal{C}$, was independently introduced by Helleseth, Kløve, and Mykkeleveit in [2], and by Wei in [8], motivated by applications from cryptography. Since the publication of the latter paper, the interest of coding theorists in the weight hierarchy has been continually increasing and it now plays a central role in coding theory. Much is known about it for several classes of codes: Hamming codes, Golay codes, cyclic codes, algebraic geometric codes, etc. For an excellent survey on this field, we refer the reader to [7].

Among algebraic geometric codes, those coming from Hermitian curves (*Hermitian codes*) have been of primary interest. The reason for this interest comes from the good properties of Hermitian curves: they have many rational points and a simple arithmetic. Thus, Hermitian codes are seen, in some sense, as a prototype of algebraic geometric codes. Their minimum distances were first obtained by Yang and Kumar in [9]; later, Yang, Kumar, and Stichtenoth [10] and Munuera [4], [5] gave the values of many higher weights in a number of cases. However, many other weights still remain unknown.

Recently, Heijnen and Pellikaan [1] gave a new bound on higher Hamming weights. This so-called *order bound*, which is a generalization of the Feng–Rao bound on the minimum distance, allowed the determination of the complete hierarchy of $q$-ary Reed–Muller codes. In this paper, by an exhaustive computation of the order bound, we are able to obtain the weight hierarchy in full for all Hermitian codes.

[*]Received by the editors August 10, 1998; accepted for publication (in revised form) August 2, 1999; published electronically January 13, 2000. This work was supported by the Junta de Castilla y León under project VA 22/96 and DGICYT PB 95-063-0002-02.
http://www.siam.org/journals/sidma/13-1/34342.html

[†]Department of Applied Mathematics (ETS de Ingenieros Industriales), University of Valladolid, Paseo del Cauce SN, 47005 Valladolid, Castilla, Spain (angbar@wmatem.eis.uva.es).

[‡]Department of Applied Mathematics (ETS de Arquitectura), University of Valladolid, Avda de Salamanca SN, 47014 Valladolid, Castilla, Spain (cmunuera@modulor.arq.uva.es).

The organization of the paper is as follows. Hermitian codes and the main arithmetical properties that we shall need are stated in section 2. In section 3 we obtain a duality formula for the weight hierarchy. In section 4 we compute the order bound. By using these tools we are able to determine the complete weight hierarchies of all Hermitian codes. Then, abundant codes are treated in section 5 and nonabundant ones in section 6. The hardest part of the paper is the computation of the order bound. In [1] this is done for Reed–Muller codes by using deep results from extremal poset theory. Here we present a direct proof, which can be generalized to a wide range of codes and curves.

The paper ends with two appendices. Appendix A contains some tables that show the obtained results. Finally, Appendix B contains an index of some notation used, which can be useful to have at hand when reading through the paper.

**2. Hermitian codes and the weight hierarchy.** Let $\mathcal{H}$ be the curve defined over $\mathbf{F}_{q^2}$ by the affine equation

$$v^q + v = u^{q+1}.$$

$\mathcal{H}$ is called the *Hermitian curve.* This is a nonsingular plane curve of genus $g = q(q-1)/2$. Over $\mathbf{F}_{q^2}$, $\mathcal{H}$ has $n = q^3$ affine points, denoted $P_1, \dots, P_n$, plus one point at infinity, denoted by $Q$. Thus, over $\mathbf{F}_{q^2}$, $\mathcal{H}$ has the maximum possible number of points allowed by the Weyl bound: it is a maximal curve.

Hermitian codes are defined as follows (see [4], [10]): Let $\mathcal{L}$ be the set of rational functions on $\mathcal{H}$ over $\mathbf{F}_{q^2}$ having poles only at $Q$. For a nonnegative integer $m$ let $\mathcal{L}(mQ)$ be the subset of $\mathcal{L}$ of functions having a pole of order at most $m$. $\mathcal{L}(mQ)$ is a vector space whose dimension, $l(mQ)$, can be computed by using the Riemann–Roch theorem. Finally, let $D$ be the divisor obtained as the sum of all affine points on $\mathcal{H}$. For $1 \le m \le n + 2g - 2$, the *Hermitian code* $\mathcal{C}(m)$ is the image of the evaluation map

$$ev : \mathcal{L}(mQ) \longrightarrow \mathbf{F}_{q^2}^n;\ f \mapsto (f(P_1), \dots, f(P_n)).$$

In order to compute the weight hierarchy of Hermitian codes, we shall use the arithmetic of the curve $\mathcal{H}$, in particular, the Weierstrass semigroup of $\mathcal{H}$ at $Q$ and the order bound on the higher Hamming weights. This section is devoted to recalling the facts from the arithmetic of $\mathcal{H}$ we shall need throughout the paper. Although these results can be found in [10] and [4], we shall explain them briefly for the convenience of the reader. The order bound will be treated in section 4.

We begin with two simple facts: For $a \in \mathbf{F}_{q^2}$, it is easy to verify that

$$\operatorname{div}(u-a) = \sum_{b^q+b=a^{q+1}} (a,b) - qQ.$$

Analogously, for $b \in \mathbf{F}_{q^2}$, such that $b^q + b \neq 0$,

$$\operatorname{div}(v-b) = \sum_{a^{q+1}=b^q+b} (a,b) - (q+1)Q.$$

From these relations we can deduce two important properties of Hermitian codes: First, the Weierstrass semigroup $S$ of $\mathcal{H}$ at the point $Q$ contains $q = -v_Q(u)$ and $q+1 = -v_Q(v)$. Thus, the semigroup generated by $q$ and $q+1$ is contained in $S$, $\langle q, q+1 \rangle \subseteq S$, and since both semigroups have the same genus $g$, we conclude that

$S = \langle q, q+1\rangle$. In particular $S$ is symmetric, so $(2g-2)Q$ is a canonical divisor. Then, according to [6], the codes $\mathcal{C}(m)$ and $\mathcal{C}(n+2g-2-m)$ are dual to each other.

Second, let us recall that the number $\alpha = l(mQ-D) = \dim\ker(ev)$ is called the *abundance* of the code. Codes of positive abundance are called *abundant*. Since the divisors $D$ and $nQ$ are equivalent, the codes $\mathcal{C}(m)$ are abundant for $m \geq n$.

We shall usually write $S$ as an enumeration of its elements in increasing order, $S = \{\rho_1, \rho_2, \rho_3, \ldots\}$, with $\rho_1 = 0, \rho_2 = q, \rho_3 = q+1, \ldots$. Each of these $\rho$'s is called a *pole number* (or a *nongap*) at $Q$. It can be written uniquely as a linear combination of $q$ and $q+1$ with nonnegative integer coefficients $\rho = xq + y(q+1)$, $0 \leq x, 0 \leq y < q$.

Let us now deal with the weight hierarchy. Its three fundamental properties were stated by Wei [8].

PROPOSITION 2.1. *If $\mathcal{C}$ is a linear code of length $n$ and dimension $k$, then*

(a) *(the monotonicity) $1 \leq d_1(\mathcal{C}) < d_2(\mathcal{C}) < \cdots < d_k(\mathcal{C}) \leq n$;*

(b) *(the duality) if $d_1(\mathcal{C}^\perp), \ldots, d_{n-k}(\mathcal{C}^\perp)$ is the weight hierarchy of the dual of $\mathcal{C}$, then $\{d_1(\mathcal{C}^\perp), \ldots, d_{n-k}(\mathcal{C}^\perp)\} \cup \{n+1-d_1(\mathcal{C}), \ldots, n+1-d_k(\mathcal{C})\} = \{1, \ldots, n\}$;*

(c) *(the generalized Singleton bound) for $r = 1, \ldots, k$, $d_r(\mathcal{C}) \leq n-k+r$.*

According to the duality property, it is enough to compute the generalized weights of Hermitian codes $\mathcal{C}(m)$ for $m \geq \lfloor (n+2g-2)/2 \rfloor$. We shall return to this in more detail in the next section.

For the sake of simplicity, from now on the $r$th weight of the code $\mathcal{C}(m)$ will be written as $d_r(m)$ instead of $d_r(\mathcal{C}(m))$. As previously stated, these generalized weights admit an arithmetical interpretation (concerning the arithmetic of $\mathcal{H}$).

PROPOSITION 2.2. *Let $\mathcal{C}(m)$ be a Hermitian code of dimension $k$ and abundance $\alpha$. For every $r$, $1 \leq r \leq k$, we have*

$$d_r(m) = \min\{n - \deg(D') \mid 0 \leq D' \leq D, l(mQ-D') \geq r+\alpha\}.$$

We say that an integer $m$ has the property (*) if $m$ is a pole number at $Q$, $m = xq + y(q+1)$ with $x, y \in \mathbf{Z}$, $0 \leq x, 0 \leq y < q$, and either $x \leq q^2-q-1$ or $y = 0$. If $m$ verifies the property (*), then there is a divisor $D'$, $0 \leq D' \leq D$, such that $D' \sim mQ$. From this fact and the previous arithmetical interpretation, we have the following result ([4, Proposition 12]).

PROPOSITION 2.3. *Let $\mathcal{C}(m)$ be a Hermitian code of dimension $k$ and abundance $\alpha$. For $r = 1, \ldots, k$, we have*

(a) *$d_r(m) \geq n - m + \rho_{r+\alpha}$;*

(b) *if $n - m + \rho_{r+\alpha}$ has the property (*), then $d_r(m) = n - m + \rho_{r+\alpha}$.*

*Proof* (sketch). For every positive integer $r$, we define the $r$th-gonality of $\mathcal{H}$ as

$$\gamma_r = \min\{\deg(A) \mid A \in Div(\mathcal{H}), l(A) \geq r\}.$$

Let $D' \leq D$ be an effective divisor such that $d_r(m) = n - \deg(D')$. Then $l(mQ-D') \geq r+\alpha$, so $m - \deg(D') \geq \gamma_{r+\alpha}$ and thus $d_r(m) \geq n - m + \gamma_{r+\alpha}$. Since for Hermitian curves we have $\gamma_s = \rho_s$ for all $s$ (see [4, Proposition 3]), then (a) is proved. Let us prove (b). If $n - m + \rho_{r+\alpha}$ has the property (*), then, by using the expressions for $\operatorname{div}(u-a)$ and $\operatorname{div}(v-b)$ given previously, we can obtain a divisor $D'$, $0 \leq D' \leq D$ such that $(n-m+\rho_{r+\alpha})Q \sim D'$ (see [4, Proposition 11]). Since $D \sim nQ$, we have $D'' = D - D' \sim (m - \rho_{r+\alpha})Q$, so $l(mQ - D'') = l(\rho_{r+\alpha}Q) = r+\alpha$, and according to Proposition 2.2, we have $d_r(m) \leq n - \deg(D'') = \deg(D') = n - m + \rho_{r+\alpha}$. $\square$

An immediate consequence of this proposition and the generalized Singleton bound is the following.

COROLLARY 2.4. *Let $\mathcal{C}(m)$ be a Hermitian code of dimension $k$ and abundance $\alpha$. If $r+\alpha > g$, then $d_r(m) = n-k+r$.*

Thus, we have to compute $d_r(m)$ only for $r = 1, \dots, g-\alpha$. We shall also use the nested character of the codes $\mathcal{C}(m)$ in the following way.

PROPOSITION 2.5. *For $1 \le m \le n+2g-2$ and $1 \le r \le \dim \mathcal{C}(m) - 1$, we have*

$$d_r(m) \le d_r(m-1) \le d_{r+1}(m).$$

**3. Duality and the weight hierarchy.** Let $\mathcal{C}$ be a linear $[n,k]$ code and let $\mathcal{C}^\perp$ be its dual. If $d_1, \dots, d_k$ and $d_1^\perp, \dots, d_{n-k}^\perp$ are, respectively, the hierarchies of $\mathcal{C}$ and $\mathcal{C}^\perp$, according to the Wei's duality formula, given in Proposition 2.1, we have

$$\{d_1^\perp, \dots, d_{n-k}^\perp\} = \{1, \dots, n\} \setminus \{n+1-d_1, \dots, n+1-d_k\}$$

that uniquely determines the hierarchy of $\mathcal{C}^\perp$ once the hierarchy of $\mathcal{C}$ is known. However, this formula has the disadvantage that it does not explicitly provide the values $d_r^\perp$. In this section we deduce another duality formula which provides such values.

For $r = 1, \dots, k$, let us consider subcodes $V_1, \dots, V_k \subseteq \mathcal{C}$ such that $\dim V_r = r, \#\mathrm{supp} V_r = d_r$. Furthermore, for every subset $I \subseteq \{1, \dots, n\}$, let

$$\begin{aligned} \mathcal{C}(I) &= \{\mathbf{x} \in \mathcal{C} \mid \mathrm{supp}\ (\mathbf{x}) \subseteq I\}, \\ k(I) &= \dim \mathcal{C}(I), \\ k^\perp(I) &= \dim \mathcal{C}^\perp(I). \end{aligned}$$

LEMMA 3.1. *For all $r$, $1 \le r \le k$, we have $V_r = \mathcal{C}(\mathrm{supp} V_r)$.*

*Proof.* The proof follows from the definition of $V_r$. ☐

Thus, the subcodes $V_r$ are determined by their supports, $I_r = \mathrm{supp} V_r$. Given a set $I \subseteq \{1, \dots, n\}$ we shall denote by $I^*$ the complement of $I$, that is, $I^* = \{1, \dots, n\} \setminus I$.

PROPOSITION 3.2. *For every set $I \subseteq \{1, \dots, n\}$, we have $k(I) = k - n + \#I + k^\perp(I^*)$.*

*Proof.* Let $L = \bigcap_{i \notin I} (x_i = 0)$. Then $\mathcal{C}(I) = \mathcal{C} \cap L$ and $(\mathcal{C} + L)^\perp = \mathcal{C}^\perp(I^*)$. The statement follows from the dimension's formula applied to the subspaces $\mathcal{C}$ and $L$. ☐

DEFINITION 3.3. *We say that a set $I \subseteq \{1, \dots, n\}$ is a* minimum support *for $\mathcal{C}$ (respectively, for $\mathcal{C}^\perp$) if $\#I = d_{k(I)}$.*

In order to simplify the exposition, from now on in this section, let us define $d_0 = 0$ and $d_{k+1} = n+1$.

PROPOSITION 3.4. *Let $I \subseteq \{1, \dots, n\}$ and let $r = k(I)$. If $d_{r+1} > \#I + 1$, then $I^*$ is a minimum support for $\mathcal{C}^\perp$.*

*Proof.* Let us first assume $r < k$. According to Proposition 3.2, $k^\perp(I^*) = n-k+r-\#I$. If $I^*$ is not a minimum support for $\mathcal{C}^\perp$, then there exists $J \subseteq \{1, \dots, n\}$, such that $k^\perp(J^*) = n-k+r-\#I$ and $\#J^* = \#I^* - t$, for some $t > 0$, so $k(J) = r+t$ and $d_{r+t} \le \#J = \#I + t$. In particular, by the monotonicity of the weight hierarchy, we have $d_{r+1} \le \#I + 1$, contradicting our assumption. In the case $r = k$, it suffices to prove that when $k(I) = k$ and $\#I < n$, then $I^*$ is a minimum support for $\mathcal{C}^\perp$. If this were not true, the same reasoning as in the former case proves that there exists $J$ such that $k(J) = k+t$ for some $t > 0$, which is absurd. ☐

For $0 \le r \le k$, we shall write $s_r = d_{r+1} - d_r$. Clearly, $s_r \ge 1$. If $s_r \ge 2$, a subset $I \subseteq \{1, \dots, n\}$ is said to be *associated* with $I_r$ whenever $I_r \subseteq I$ and $\#(I_r \setminus I) \le s_r - 2$.

PROPOSITION 3.5. *If $s_r \ge 2$ and $I$ is associated with $I_r$, then $I^*$ is a mimimum support for $\mathcal{C}^\perp$.*

*Proof.* Since $I_r \subseteq I$ and $\#I \leq d_{r+1} - 2$, then we have $k(I) = r$, and the result follows from Proposition 3.4. ☐

For every $r$, $0 \leq r \leq k$, such that $s_r \geq 2$, let us fix a set of $s_r - 1$ supports $I_r^0, \ldots, I_r^{s_r-2}$ associated with $I_r$, such that $I_r^0 = I_r$ and $\#(I_r^t \setminus I_r) = t$, $0 \leq t \leq s_r - 2$. Let $\mathcal{J} = \{I_r^t \mid 0 \leq r \leq k \text{ with } s_r \geq 2, 0 \leq t \leq s_r - 2\}$.

THEOREM 3.6. *The weight hierarchy of $\mathcal{C}^\perp$ is the set*

$$\{\#I^* \mid I \in \mathcal{J}\} = \{n - \#I \mid I \in \mathcal{J}\}.$$

*Proof.* Every $I^*$, $I \in \mathcal{J}$, is a minimum support for $\mathcal{C}^\perp$ according to Proposition 3.4. Furthermore, all of them are different, because they have different cardinality. Thus, it suffices to prove that $\#\mathcal{J} = n - k$. Since

$$\#\mathcal{J} = \sum_{r=0}^{k}(s_r - 1) = \sum_{r=0}^{k}(d_{r+1} - d_r - 1) = d_{k+1} - d_0 - (k+1) = n - k,$$

the result is proved. ☐

COROLLARY 3.7. *If $0 \leq r \leq k$ and $s_r \geq 2$, then for $0 \leq t \leq s_r - 2$, we have*

$$d^\perp_{n-k-d_r+r-t} = n - d_r - t.$$

*Proof.* The proof follows from Proposition 3.2 and Theorem 3.6. ☐

*Remark* 3.1. Note that, according to Theorem 3.6, the formula given in the corollary provides the whole weight hierarchy of $\mathcal{C}^\perp$. Observe, furthermore, that Wei's duality formula also follows from the theorem. Another property that can be derived from the theorem is that $\mathcal{C}^\perp$ verifies the chain condition whenever $\mathcal{C}$ does too.

**4. Computing the order bound.** As mentioned previously, the order bound on the generalized Hamming weights has been introduced by Heijnen and Pellikaan in [1]. Let us explain briefly how it works.

Let $\{f_1, f_2, \ldots\}$ be a basis of $\mathcal{L}$ such that $-v_Q(f_i) < -v_Q(f_{i+1})$ for all $i$, where $v_Q$ is the valuation at $Q$. Note that $-v_Q(f_i) = \rho_i$, and thus $S = \{-v_Q(f) \mid f \in \mathcal{L}\}$.

For every positive integer $l$ let us consider the Hermitian code

$$C_l = \langle ev(f_1), \ldots, ev(f_l)\rangle^\perp$$

and the set

$$A(l) = \{\rho_i \in S \mid \rho_i + \rho_j = \rho_{l+1} \text{ for some } \rho_j \in S\}.$$

For $l_1 < \cdots < l_r$, define the set $A(l_1, \ldots, l_r)$ by

$$A(l_1, \ldots, l_r) = A(l_1) \cup \cdots \cup A(l_r).$$

DEFINITION 4.1. *Let $l$ be a positive integer. The number*

$$d_r^{ORD}(l) = \min\{\#A(l_1, \ldots, l_r) \mid l \leq l_1 < \cdots < l_r\}$$

*is called the* order bound *on the $r$th generalized weight of $C_l$.*

THEOREM 4.2 (Heijnen and Pellikaan [1]). *For $1 \leq l \leq n+g$ and $1 \leq r \leq \dim C_l$, we have*

$$d_r(C_l) \geq d_r^{ORD}(l).$$

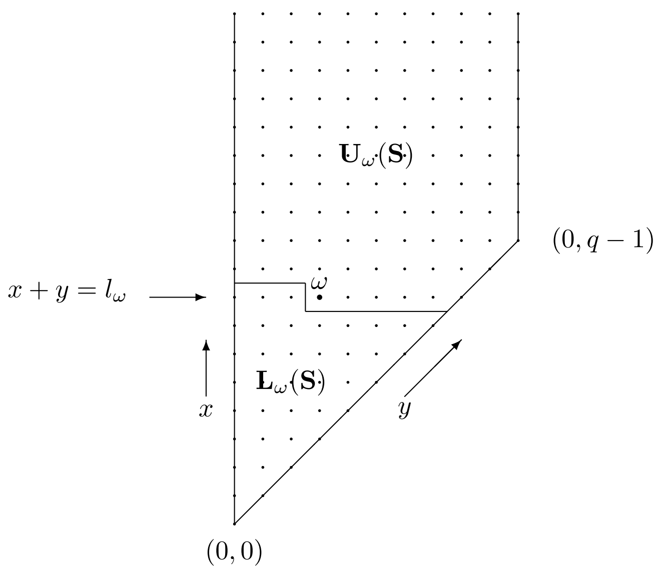

$\mathbf{U}_{\omega}(\mathbf{S})$
$(0, q-1)$
$x + y = l_\omega$
$\omega$
$\mathbf{L}_{\omega}(\mathbf{S})$
$x$
$y$
$(0, 0)$

FIG. 4.1. *Graphical representation of $S$.*

The proof of this theorem can be found in [1, Theorem 3.14]. Note that $C_l = (0)$ if $l > n + g$.

By using the notation stated in the previous section, given a Hermitian code $\mathcal{C}(m)$ of dimension $k$, let $m^\perp = n + 2g - 2 - m$. Since $\mathcal{C}(m^\perp) = \mathcal{C}(m)^\perp$ we have $\mathcal{C}(m) = C_l$ for $l = l(m^\perp Q)$. In particular, when $m > 2g-2$ (the case we consider in this paper) it holds that $l(m^\perp Q) = \dim \mathcal{C}(m^\perp) = n - k$. Thus, in this range, $m > 2g - 2$, Theorem 4.2 can be written as

$$d_r(m) \geq d_r^{ORD}(n-k)$$

for $r = 1, \ldots, k$.

As we know, the Weierstrass semigroup, $S$, of $\mathcal{H}$ at the point $Q$, is generated by $q$ and $q + 1$, so every pole in $S$ can be written uniquely as $\rho = xq + y(q + 1)$ with $0 \leq x$, $0 \leq y < q$. In the following we shall denote poles by means of two-dimensional coordinates, $\rho = (x, y)$. In this way we can identify $S$ with a plane region, as Figure 4.1 shows. For a better understanding, we recommend the reader to follow this reasoning by using this graphical interpretation.

For an integer $l$, $0 \leq l \leq n + g$, let $\omega = \rho_{l+1} = (w_1, w_2)$ and let $l_\omega = w_1 + w_2$. Our goal in this section is to compute $d_r^{ORD}(l)$ for some convenient values of $r$ and $l$. To that end, and in order to simplify the exposition, if $\rho$ is the $(s + 1)$th pole of $S$, $\rho = \rho_{s+1}$, from now on we shall write $A[\rho]$ instead of $A(s)$. Thus, $A[\rho] = \{p \in S \mid \text{ there exists } p' \in S \text{ such that } p + p' = \rho\} = \{p \in S \mid \rho - p \in S\}$. $A[\rho]$ is called the *A-set* corresponding to the pole $\rho$. We shall also write $A[p_1, \ldots, p_r] = A[p_1] \cup \cdots \cup A[p_r]$.

DEFINITION 4.3. *Let $\rho = (x, y)$ be a pole.*

(a) *We say that $\rho$ is a* type I *pole if $x \leq q$. Otherwise we say that $\rho$ is a* type II *pole.*

(b) *If $\rho$ is a type* II *pole, the* conjugate *of $\rho$ is the pole $\overline{\rho} = (x - q - 1, q - 1)$.*

Observe that all poles on the line $x = \text{constant} > q$ have the same conjugate.

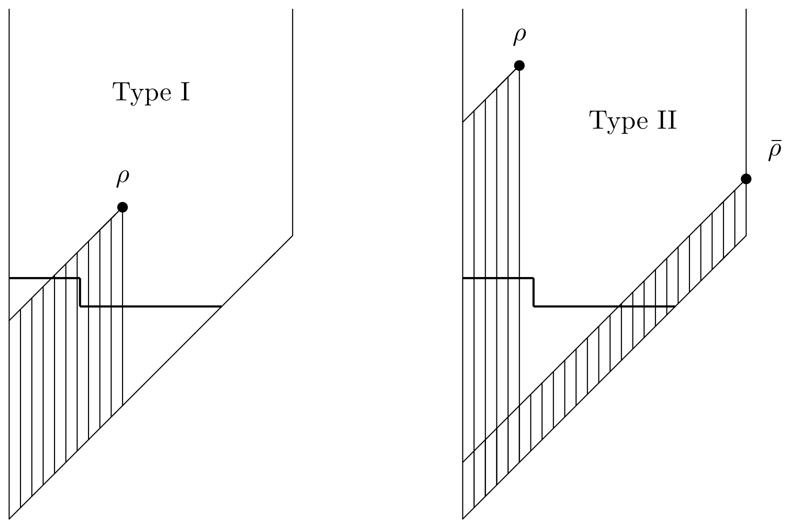


FIG. 4.2. *A-sets of poles.*

DEFINITION 4.4. *If $\rho = (x, y)$ is a pole, the* (direct) shadow *of $\rho$ is the subset*

$$S[\rho] = \{(\alpha, \beta) \in S \mid \alpha \leq x, \beta \leq y\}.$$

*If $p = (x, y)$ is a point in $\mathbf{Z}^2$ (not necessarily a pole), the* inverse shadow *of $p$ is the subset*

$$S^{-1}[p] = \{(\alpha, \beta) \in S \mid \alpha > x, \beta > y\}.$$

*Remark* 4.1. As we have seen, the direct shadow is defined only for poles, whereas the inverse shadow is defined for points in general. From now on we shall consider only inverse shadows of points in the set $\overline{S} = \{(x, y) \in \mathbf{Z}^2 \mid -1 \leq x, -1 \leq y \leq q-1\}$. We shall denote by $\pi$ the point $\pi = (-1, -1)$. Observe that $S^{-1}[\pi] = S$.

PROPOSITION 4.5. *Let $\rho = (x, y)$ be a pole. Then*

$$A[\rho] = \begin{cases} S[\rho] & \text{if } \rho \text{ is type I;} \\ S[\rho] \cup S[\overline{\rho}] & \text{if } \rho \text{ is type II.} \end{cases}$$

*Proof.* The inclusion $S[\rho] \subseteq A[\rho]$ is clear. Conversely, if $\rho = xq + y(q + 1) = (x_1q+y_1(q+1))+(x_2q+y_2(q+1)) = p_1+p_2$, then either $y_1+y_2 = y$ or $y_1+y_2 = y+q$. In the first case, $x_1 + x_2 = x$, so $p_1, p_2 \in S[\rho]$; in the second case (which can happen only when $\rho$ is type II), $y + 1 \leq y_1, y_2 \leq q - 1$, $x = x_1 + x_2 + q + 1$, so $A[\rho] = S[\rho] \cup S[\overline{\rho}]$. □

*Remark* 4.2. According to the preceding proposition, the A-set of a type I pole is just a shadow, whereas the A-set of a type II pole is not (See Figure 4.2). This difference comes not only from the semigroup structure itself but also from the representation of $S$ we are using. In fact, we can represent $S$ not as a plane set, as Figure 4.1 shows, but embedded in the cylinder obtained from $\overline{S}$, by identifying the lines

$y=-1$ and $y=q-1$ and the points $(x,-1)$ with $(x-q-1,q-1)$, $x>q$. In this way, every A-set is always an (extended) shadow.

LEMMA 4.6. *If $\rho=(x,y)$ is a pole, then $\#S[\rho]=(x+1)(y+1)$.*

*Proof.* The proof is obvious. □

LEMMA 4.7. *Let $\rho=(x,y),\rho'=(x',y')$ be two poles. Then $S[\rho]\cap S[\rho']=S[\rho'']$, where $\rho''=(\min\{x,x'\},\min\{y,y'\})$. In particular, if $\rho$ is a type* II *pole, then*

$$S[\rho]\cap S[\overline{\rho}]=S[p],$$

*where $p=(x-q-1,y)$.*

*Proof.* The proof is a simple computation. □

PROPOSITION 4.8. *Let $\rho=(x,y)$ be a pole. Then*

$$\#A[\rho]=\begin{cases}(x+1)(y+1) & \text{if } \rho \text{ is type I;}\\ (x+1)(y+1)+(x-q)(q-1-y) & \text{if } \rho \text{ is type II.}\end{cases}$$

*Proof.* The proof is a direct consequence of Proposition 4.5 and Lemmas 4.6 and 4.7. □

DEFINITION 4.9. *Let $T$ be a subset of $S$ and $\omega$ as above.*

(a) *The $\omega$-*lower part *of $T$ is defined as*

$$L_\omega(T)=\{\rho\in T \mid \rho<\omega\}.$$

*The $\omega$-*upper part *of $T$ is defined as*

$$U_\omega(T)=\{\rho\in T \mid \rho\geq\omega\}.$$

(b) *If $p\in\overline{S}$, the $\omega$-*lower part of $T$ with respect to $p$ *is defined as*

$$L_\omega^{(p)}(T)=L_\omega(T)\cap S^{-1}[p].$$

*The $\omega$-*upper part of $T$ with respect to $p$ *is defined as*

$$U_\omega^{(p)}(T)=U_\omega(T)\cap S^{-1}[p].$$

Graphically, $L_\omega(T)$ is the part of $T$ below the line $x+y=l_\omega$ plus the points $(x,l_\omega-x)\in T$, $0\leq x\leq w_1$. $U_\omega(T)$ is the rest of $T$.

For $r=1,2,\dots,$ and $p\in\overline{S}$, let

$$\mathcal{P}_\omega(r,p)=\{\{p_1,\dots,p_r\}\subseteq U_\omega^{(p)}(S) \mid \text{ the } p_i\text{'s are all distinct}\}.$$

Then, by definition

$$d_r^{ORD}(l)=\min\{\#A[p_1,\dots,p_r] \mid \{p_1,\dots,p_r\}\in\mathcal{P}_\omega(r,\pi)\},$$

thus, if $d_r^{ORD}(l)=\#A[p_1,\dots,p_r]$, roughly speaking, we say that $\#A[p_1\dots,p_r]$ is *minimum in $\mathcal{P}_\omega(r,\pi)$.*

LEMMA 4.10. *If $r_1<r_2$, then $d_{r_1}^{ORD}(l)<d_{r_2}^{ORD}(l)$.*

*Proof.* For the proof it suffices to consider the case $r_1=r,r_2=r+1$.

If $d_{r+1}^{ORD}(l)=\#A[p_1,\dots,p_r,p_{r+1}]$ with $p_1<\cdots<p_r<p_{r+1}$, then $p_{r+1}\notin A[p_1,\dots,p_r]$, so $\#A[p_1,\dots,p_r,p_{r+1}]>\#A[p_1,\dots,p_r]\geq d_r^{ORD}(l)$. □

PROPOSITION 4.11. *If $d_r^{ORD}(l)=\#A[p_1,\dots,p_r]$, then, for all $i=1,\dots,r$, we have $U_\omega(A[p_i])\subseteq\{p_1,\dots,p_r\}$.*

*Proof.* If there exists $p \in U_\omega(A[p_i]) \setminus \{p_1, \dots, p_r\}$ for some $i$, then since $p \in A[p_1, \dots, p_r]$, it holds that $A[p_1, \dots, p_r] = A[p_1, \dots, p_r, p]$, so $\#A[p_1, \dots, p_r]$ cannot be minimum in $\mathcal{P}_\omega(r, \pi)$, according to Lemma 4.10. □

Thus, if $\#A[p_1, \dots, p_r]$ is minimum in $\mathcal{P}_\omega(r, \pi)$, then $\{p_1, \dots, p_r\}$ is a union of $U_\omega$-parts of A-sets. Throughout the paper we shall consider only sets $\{p_1, \dots, p_r\} \in \mathcal{P}_\omega(r, \pi)$ having the property of being $U_\omega$-parts of union of A-sets, and we shall write

$$\mathcal{P}^*{}_\omega(r,p) = \{\{p_1, \dots, p_r\} \in \mathcal{P}_\omega(r,p) \mid \{p_1, \dots, p_r\} = U_\omega(A[p_1, \dots, p_r])\}.$$

The poles $p_i \in \{p_1, \dots, p_r\}$ such that $p_i \notin A[p_j]$ for all $j = 1, \dots, r$, with $i \neq j$, are called *maximal elements* in the set. If $p_{i_1}, \dots, p_{i_t}$ are the maximals in $\{p_1, \dots, p_r\}$, then $A[p_1, \dots, p_r] = A[p_{i_1}, \dots, p_{i_t}]$. Conversely, if $\#A[p_1, \dots, p_r]$ is minimum in $\mathcal{P}^*{}_\omega(r, \pi)$, then

$$d_r^{ORD}(l) = \#A[p_1, \dots, p_r] = r + \#L_\omega(A[p_1, \dots, p_r])$$

so $\#A[p_1, \dots, p_r]$ is minimum if and only if $\#L_\omega(A[p_1, \dots, p_r])$ is minimum as well. In the rest of this section we shall investigate this condition.

Definition 4.12. *Let $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$ be two poles. A pole $p = (x, y) \in A[p_1] \cap A[p_2]$ such that $x + y \geq l_\omega - 1$ is called a* link *between $p_1$ and $p_2$. If, furthermore, $y = y_2$, then the link $p$ is said to be* from $p_1$ to $p_2$. *If there exists a link between $p_1$ and $p_2$, we say that these poles are* joined *(with respect to $\omega$). In any other case we say that these poles are* separated *(with respect to $\omega$).*

Definition 4.13. *Let $\{p_1, \dots, p_r\} \subseteq U_\omega(S)$ be a set of poles and let $p_{i_1}, \dots, p_{i_t}$ be the maximals of this set. $\{p_1, \dots, p_r\}$ is called a* block *if*

(a) *$\{p_1, \dots, p_r\} \in \mathcal{P}^*{}_\omega(r, \pi)$; and*

(b) *there is a reordering, $p_1^*, \dots, p_t^*$ of $p_{i_1}, \dots, p_{i_t}$ such that $p_{j-1}^*$ and $p_j^*$ are joined for all $j = 2, \dots, t$.*

*If $T \in \mathcal{P}^*{}_\omega(s, \pi)$, a block $\{p_1, \dots, p_r\} \subseteq T$ is called* maximal in $T$ *(or simply* maximal *when the set $T$ is intended) if there is no other block in $T$ containing it.*

Let $B = \{p_1, \dots, p_r\}$ be a block with maximals $p_{i_1}, \dots, p_{i_t}$. In what follows we shall order these maximals as $p_1^*, \dots, p_t^*$, with $p_j^* = (x_j^*, y_j^*)$, in a way such that

(i) for all $j = 2, \dots, t$, there is a link from $p_{j-1}^*$ to $p_j^*$; and

(ii) $y_{j-1}^* > y_j^*$ for all $j = 2, \dots, t$ except at most in one case;

that is, we fix the order given by the links from the right to the left in the cylindrical representation of $S$. Thus, either $y_1^* > y_2^* > \cdots > y_t^*$, or there exists $j$ such that $y_j^* > y_{j+1}^* > \cdots > y_t^* > y_1^* > \cdots > y_{j-1}^*$. An enumeration of the maximals in $B$ verifying (i) and (ii) is called *well-ordered.* From now on, all enumerations of maximals will be well-ordered.

Definition 4.14. *Let $B = \{p_1, \dots, p_r\}$ be a block with maximals $p_1, \dots, p_t$, well-ordered. Let $\beta = y_1$ and let*

$$\alpha = \begin{cases} x_t & \text{if } y_1 > y_2 > \cdots > y_t; \\ x_t + q + 1 & \text{if } y_{j-1} < y_j \text{ for some (unique) } j. \end{cases}$$

*The pole $b = (\alpha, \beta)$ is called a* vertex *of the block $B$.*

*Remark* 4.3. If $p_1, \dots, p_t$ is a well-ordered enumeration of the maximals of some block, and there is a link from $p_t$ to $p_1$, then every cyclic permutation of $p_1, \dots, p_t$ is also well-ordered. Thus, in general, a well-ordered enumeration of the maximals of one block is not unique, so the vertex of this block is also not uniquely defined. Then, given a block, we can speak of a "vertex of the block with respect to a certain well-ordered enumeration of its maximals." However, note that the only case in which a well-ordered enumeration of the maximals is not unique is the one mentioned above, and in such a case $L_\omega(A[p_1,\dots,p_t]) = L_\omega(S)$, so the computation of the order bound is immediate. We remark, finally, that if all poles in the block are type I, then the vertex is uniquely defined and it is also a type I pole.

Some important properties of blocks and their vertices are stated in the next propositions.

PROPOSITION 4.15. *If $B = \{p_1,\dots,p_r\}$ is a block with vertex $b$, then $A[p_1,\dots,p_r] \subseteq A[b]$.*

*Proof.* Let $p_1 = (x_1,y_1),\dots,p_t = (x_t,y_t)$ be a well-ordered enumeration of the maximals in $B$, and let $b = (\alpha,\beta)$ be the vertex of $B$ with respect to this enumeration. We shall proceed by induction on $t$. If $t = 1$, then $B = U_\omega(A[p_1])$ and the result is clear. Let us assume the result is true for $t-1$ and prove it for $t > 1$. Let $B' = U_\omega(A[p_1,\dots,p_{t-1}])$. Since the enumeration $p_1,\dots,p_{t-1}$ is well-ordered, then $B'$ is a block. Let $b' = (\alpha',\beta')$ be a vertex of this block with respect to this enumeration. By an induction hypothesis, $B' \subseteq A[b']$, so it suffices to prove $p_t, b' \in A[b]$. We have to consider three cases:

*Case* 1. If $y_1 > \dots > y_t$, then $x_1 < \dots < x_t$, and $\alpha = x_t, \alpha' = x_{t-1}, \beta = \beta' = y_1$. Thus $p_t, b' \in S[b] \subseteq A[b]$.

*Case* 2. If $y_t > y_1 > \dots > y_{t-1}$, then $x_t < x_1 < \dots < x_{t-1}$, and $\alpha = x_t+q+1, \alpha' = x_{t-1}, \beta = \beta' = y_1$. Furthermore, the link from $p_{t-1}$ to $p_t$ is in $S[\overline{p_{t-1}}] \cap S[p_t]$, so $p_{t-1}$ is a type II pole. By definition of maximal, $p_t \notin S[\overline{p_{t-1}}]$, so $x_t > x_{t-1} - q - 1$; thus $b' \in S[b]$ and $p_t \in S[\bar{b}]$.

*Case* 3. If $y_j > \dots > y_t > y_1 > \dots > y_{j-1}$ for some $j \le t-1$, then $x_j < \dots < x_t < x_1 < \dots < x_{j-1}$ and $\alpha = x_t + q + 1, \alpha' = x_{t-1} + q + 1, \beta = \beta' = y_1$, so $b' \in S[b], p_t \in S[\bar{b}]$. □

PROPOSITION 4.16. *If $B = \{p_1,\dots,p_r\}$ is a block with vertex $b$, then $L_\omega(A[b]) = L_\omega(A[p_1,\dots,p_r])$.*

*Proof.* Let $p_1 = (x_1,y_1),\dots,p_t = (x_t,y_t)$ be a well-ordered enumeration of the maximals in $B$, and let $b = (\alpha,\beta)$ be its vertex with respect to this enumeration. After the above proposition, it is enough to prove $L_\omega(A[b]) \subseteq L_\omega(A[p_1,\dots,p_t])$. Seeking a contradiction, assume that there is a pole $p = (x,y) \in L_\omega(A[b]) \setminus L_\omega(A[p_1,\dots,p_t])$. We shall consider three cases.

*Case* 1. If $b$ is a type I pole, then all poles in $B$ are also type I; henceforth $y_1 > \dots > y_t$, so $x_1 < \dots < x_t$. Since $p \in A[b] = S[b]$, we have $x \le x_t, y \le y_1$. If $y \le y_t$, then $p \in S[p_t]$, which is a contradiction. Thus, there exists $j$ such that $y_{j-1} \ge y > y_j$. Since $p \notin S[p_{j-1}]$, we have $x_{j-1} < x$, so $x_{j-1} + y_j \le x + y - 2 \le l_\omega - 2$, because $p \in L_\omega(S)$. Conversely, and since there is a link from $p_{j-1}$ to $p_j$, $x_{j-1} + y_j \ge l_\omega - 1$, which is again a contradiction.

*Case* 2. If $b$ is a type II pole and $y_1 > \dots > y_t$, since $S[\bar{b}] = S[\overline{p_t}]$, then $p \notin S[\bar{b}]$, and the same reasoning as in the previous case proves the result.

*Case* 3. If $b$ is a type II pole and $y_j > \dots > y_t > y_1 > \dots > y_{j-1}$ for some $j$, then $x_j < \dots < x_t < x_1 < \dots < x_{j-1}$. Since $y_{j-1} < y_j$, then $(x_{j-1}-q-1, y_j)$ is a link from $p_{j-1}$ to $p_j$, hence $x_{j-1}-2 \ge x_{j-1}-q-1+y_j \ge l_\omega-1$, that is, $x_{j-1} \ge l_\omega+1$. Let us first assume $p \in S[b]$. Let $B'$ be the block $B' = U_\omega(A[p_1,\dots,p_{j-1}])$ and $b' = (x_{j-1}, y_1)$ its vertex. Since $p \in S[b]$, we have $y \le y_1$. If $x > x_{j-1}$, then $l_\omega + 1 \le x_{j-1} < x \le x + y \le l_\omega$, a contradiction. Then $x \le x_{j-1}$, hence $p \in S[b'] \setminus A[p_1,\cdots,p_{j-1}]$ and we can apply the previous Case 2, getting a contradiction. Finally, if $p \in S[\bar{b}]$, we can consider the block $B'' = U_\omega(A[\overline{p_{j-1}}, p_j,\cdots,p_t])$; with a similar reasoning we fall in a contradiction again. □

Proposition 4.17. *Let $T \in \mathcal{P}_\omega^*(s,\pi)$. Every two maximal blocks in $T$ have separated vertices.*

*Proof.* If $B_1, B_2$ are maximal blocks in $T$, having joined vertices $b_1$ and $b_2$, we can find maximals $p_1$ in $B_1$ and $p_2$ in $B_2$ such that $p_1$ and $p_2$ are joined, in contradiction with the fact of the blocks being maximal. Details, which are simple but rather tedious, are left to the reader. □

If $d_r^{ORD}(l) = \#A[p_1, \cdots, p_r]$, we know that the set $\{p_1, \ldots, p_r\}$ is a union of $U_\omega$-parts of A-sets, so we are interested in finding the (minimum) value of $\#L_\omega(A[p_1, \cdots, p_r])$. Now, according to Propositions 4.15, 4.16, and 4.17, it arises that the simplest way for handling these poles is to consider them as grouped in blocks. Next, we shall investigate how many blocks the set $T = \{p_1, \ldots, p_r\}$ can have. To do this we shall distinguish two cases according to $l_\omega$. From now on, all the blocks we consider will be maximal. For the sake of simplicity, we shall simply write "block" instead of "maximal block in $\{p_1, \ldots, p_r\}$".

**4.1. The case $l_\omega \le q-1$.** The first case we consider is $l_\omega \le q-1$. We begin with a somewhat technical lemma.

Lemma 4.18. *Assume $l_\omega \le q-1$ and let $p = (x,y) \in \overline{S}$ be such that $L_\omega^{(p)}(S) \neq \emptyset$. For a positive integer $t$, let us consider the set*

$$\mathcal{S}_\omega(t,p) = \{\rho \in U_\omega^{(p)}(S) \mid \#U_\omega^{(p)}(S[\rho]) \ge t\}.$$

*If the function $f(\rho) = \#L_\omega^{(p)}(S[\rho])$, defined in $\mathcal{S}_\omega(t,p)$, attains its minimum at $\rho \in \mathcal{S}_\omega(t,p)$, then either $(x+1, l_\omega - x - 1) \in S[\rho]$ or $(l_\omega - y - 1, y+1) \in S[\rho]$.*

*Proof.* Up to changing coordinates, we can assume $p = \pi = (-1,-1)$, so that $S^{-1}[p] = S$. If $\rho = (\alpha, \beta)$, we have to prove $\alpha \ge l_\omega$ or $\beta \ge l_\omega$. Seeking a contradiction, suppose that $\alpha < l_\omega$ and $\beta < l_\omega$. Let $\rho' = (\alpha', \beta') = (\alpha + \beta - l_\omega, l_\omega)$. Note that $\rho' \in S$, since $l_\omega \le q-1$ and $\alpha + \beta \ge l_\omega$, because $\rho \in U_\omega(S)$. Furthermore, since $\alpha' + \beta' = \alpha + \beta$ and $\beta' > \beta$, we have $\rho' > \rho$, hence $\rho' \in U_\omega(S)$. On the other hand, since $\beta' = l_\omega$, the same argument $\alpha' + \beta' = \alpha + \beta$ implies that $\#U_\omega(S[\rho']) \ge \#U_\omega(S[\rho]) \ge t$. Then $\rho' \in \mathcal{S}_\omega(t,p)$. To end the proof we shall show that $f(\rho) > f(\rho')$. Since $f(\rho) = (\alpha+1)(\beta+1) - \#U_\omega(S[\rho])$ and $f(\rho') = (\alpha'+1)(\beta'+1) - \#U_\omega(S[\rho']) \le (\alpha'+1)(\beta'+1) - \#U_\omega(S[\rho])$, it is enough to prove $(\alpha+1)(\beta+1) > (\alpha'+1)(\beta'+1)$. From elementary calculus, we know that the function $\phi(u,v) = uv$ defined for positive $u, v$ on the line $u + v = constant > 0$ is increasing as $|u - v|$ decreases. Since $|\alpha' - \beta'| = |\alpha + \beta - 2l_\omega| = 2l_\omega - (\alpha + \beta) > |\alpha - \beta|$, then $f$ cannot have a minimum at $\rho$. □

Proposition 4.19. *Assume $l_\omega \le q-1$. If $d_r^{ORD}(l) = \#A[p_1, \ldots, p_r]$, then*

(a) *there are at most two (maximal) blocks in the set $\{p_1, \ldots, p_r\}$.*

(b) *If there are type* II *poles among $p_1, \ldots, p_r$, then $\{p_1, \ldots, p_r\}$ form one block.*

(c) *In the case where there is one block with vertex $b$, then either $(0, l_\omega) \in S[b]$ or $(l_\omega, 0) \in S[b]$.*

(d) *In the case where there are two blocks with vertices $b_1$ and $b_2$, then $(0, l_\omega) \in S[b_1]$ and $(l_\omega, 0) \in S[b_2]$.*

*Proof.* (a) Assume there are $h$ blocks in $\{p_1, \ldots, p_r\}$, with vertices $b_1 = (\alpha_1, \beta_1), \ldots, b_h = (\alpha_h, \beta_h)$. The property of the blocks being maximal implies that the $b_i$'s are separated. In particular, all the $\alpha$'s are different and all the $\beta$'s are different. Up to reordering, write $\alpha_1 < \alpha_2 < \cdots < \alpha_h$, so $\beta_1 > \beta_2 > \cdots > \beta_h$. Let us prove that there are at most two blocks. If, on the contrary, $h \ge 3$, take $i$, $1 < i < h$.

Since $b_i$ and $b_{i+1}$ are separated, $\alpha_i + \beta_{i+1} < l_\omega - 1$ so, being $\alpha_{i-1} < \alpha_i$, we have $\alpha_{i-1} + \beta_{i+1} < l_\omega - 2$. Let $p = (\alpha_{i-1}, \beta_{i+1})$. Since $(\alpha_{i-1}+1, \beta_{i+1}+1) \in L_\omega^{(p)}(S)$, it holds that $L_\omega^{(p)}(S) \neq \emptyset$. Since $\#A[p_1,\dots,p_r] = r + \#L_\omega(A[p_1,\dots,p_r])$ is minimum, then so is $\#L_\omega(A[p_1] \cup \cdots \cup A[p_r]) = \#(L_\omega(A[b_1]) \cup \cdots \cup L_\omega(A[b_h]))$. Note that $L_\omega(A[b_1]) \cup \cdots \cup L_\omega(A[b_h]) = L_\omega(S[b_1]) \cup \cdots \cup L_\omega(S[b_h])$ if all vertices are type I, and $L_\omega(A[b_1]) \cup \cdots \cup L_\omega(A[b_h]) = L_\omega(S[b_1]) \cup \cdots \cup L_\omega(S[b_h]) \cup L_\omega(S[\overline{b_h}])$ if there are type II vertices. In any case, $S[b_i] \cap S^{-1}[p]$ must be minimum in $\mathcal{P}_\omega^*(\#U_\omega^{(p)}(S[b_i]), p)$; that is, by using the notation as in Lemma 4.18, if $s = \#U_\omega^{(p)}(S[b_i]) = \#U_\omega(S[b_i])$, the function $f$, defined in $\mathcal{S}_\omega(s,p)$, attains its minimum at $b_i$. Then, according to this lemma, either $(\alpha_{i-1}+1, l_\omega - \alpha_{i-1} - 1) \in S[b_i]$ or $(l_\omega - \beta_{i+1} - 1, \beta_{i+1}+1) \in S[b_i]$. If $(\alpha_{i-1}+1, l_\omega - \alpha_{i-1} - 1) \in S[b_i]$, then $\beta_i \geq l_\omega - \alpha_{i-1} - 1$, so $\rho = (\alpha_{i-1}, \beta_i)$ is a link between $b_{i-1}$ and $b_i$, and hence these poles are joined, in contradiction to the fact of the blocks being maximal. If $(l_\omega - \beta_{i+1} - 1, \beta_{i+1}+1) \in S[b_i]$, a similar reasoning proves that $b_i$ and $b_{i+1}$ are joined, again in contradiction to the fact of the blocks being maximal. Since this argument can be applied to every $i$, $1 < i < h$, then there are at most two blocks: those with vertices $b_1$ and $b_h$. This proves (a).

(b) If there are type II poles in the set $\{p_1,\dots,p_r\}$, then $b_h$ must be type II, so $L_\omega(A[b_1]) \cup \cdots \cup L_\omega(A[b_h]) = L_\omega(S[b_1]) \cup \cdots \cup L_\omega(S[b_h]) \cup L_\omega(S[\overline{b_h}])$, and the same reasoning as done in part (a) for the $i$th vertex, $1 < i < h$, can be extended to $1 \leq i < h$, proving that there exists just one block.

(c) If there is one block with vertex $b$, and $b$ is type I, take $p = (-1,-1)$, and Lemma 4.18 implies the result. If $b$ is type II, then clearly $(l_\omega, 0) \in S[b]$ and $(0, l_\omega) \in S[\overline{b}]$.

(d) Finally, if there are two blocks, then all poles $p_1,\dots,p_r$, are type I. Let $b_1 = (\alpha_1, \beta_1)$ and $b_2 = (\alpha_2, \beta_2)$ be vertices of these blocks, with $\alpha_1 < \alpha_2$, and let $p = (-1, \beta_2)$. Again according to Lemma 4.18, we have $(l_w - \beta_2 - 1, \beta_2 + 1) \in S[b_1]$ or $(0, l_\omega) \in S[b_1]$. Since the first possibility cannot happen because the blocks are separated, we obtain $(0, l_\omega) \in S[b_1]$. A similar reasoning proves that $(l_\omega, 0) \in S[b_2]$. $\square$

To compute the cardinalities of the sets $L_\omega$ and $U_\omega$, we shall introduce the following function: for $j = 0, 1, \dots, l_\omega$, define

$$\delta_\omega(j) = \begin{cases} 0 & \text{if } (j, l_\omega - j) \in U_\omega(S); \\ -1 & \text{if } (j, l_\omega - j) \in L_\omega(S). \end{cases}$$

Note that the function $\delta_\omega$ verifies $\delta_\omega(j) \leq \delta_\omega(j-1)$, $j = 1, \dots, l_\omega$. Conversely, for a nonnegative integer $j$, let us define

$$\sigma(j) = 0 + 1 + \cdots + j = \frac{j(j+1)}{2}.$$

The following lemma will be useful in what follows.

Lemma 4.20. *Assume $l_\omega \leq q-1$, and let $p = (x,y)$ be a pole such that $y = q-1$ or $x \geq l_\omega$.*

(a) *If $y = q-1$ and $x \leq l_\omega$, then*

$$\#L_\omega(S[p]) = (x+1)l_\omega - \sigma(x) - \sum_{j=0}^{x} \delta_\omega(j);$$

$$\#U_\omega(S[p]) = (x+1)(q - l_\omega) + \sigma(x) + \sum_{j=0}^{x} \delta_\omega(j).$$

(b) *If $x \geq l_\omega$ and $y \leq l_\omega$, then*

$$\#L_\omega(S[p]) = (y+1)l_\omega - \sigma(y) - \sum_{j=0}^{y} \delta_\omega(l_\omega - j);$$

$$\#U_\omega(S[p]) = (y+1)(x - l_\omega + 1) + \sigma(y) + \sum_{j=0}^{y} \delta_\omega(l_\omega - j).$$

(c) *If $x > q$, and $x + y - q \leq l_\omega$, then*

$$\#L_\omega(A[p]) = (x+y-q+1)l_\omega - \sigma(x+y-q) - \sum_{j=0}^{y} \delta_\omega(l_\omega - j) - \sum_{j=0}^{x-q-1} \delta_\omega(j);$$

$$\#U_\omega(A[p]) = (x+y-q+1)(q-l_\omega) + \sigma(x+y-q) + y + 1 + \sum_{j=0}^{y} \delta_\omega(l_\omega - j) + \sum_{j=0}^{x-q-1} \delta_\omega(j).$$

(d) *If $x + y - q \geq l_\omega$, then $L_\omega(A[p]) = L_\omega(S)$.* □

PROPOSITION 4.21. *Assume $l_\omega \leq q-1$. For $1 \leq r \leq g$, there is a set of poles, $\{p_1, \ldots, p_r\} \in \mathcal{P}^*_\omega(r, \pi)$, forming one block and such that $d_r^{ORD}(l) = \#A[p_1, \ldots, p_r]$.*

*Proof.* Take $r$, $1 \leq r \leq g$, and write $d_r^{ORD}(l) = \#A[p'_1, \ldots, p'_r]$. If these poles form one block, then there is nothing to prove. Otherwise $p'_1, \ldots, p'_r$ are all type I and form two blocks, $B_1, B_2$, with vertices $b_1 = (\alpha_1, \beta_1)$ and $b_2 = (\alpha_2, \beta_2)$, such that $(0, l_\omega) \in B_1$ and $(l_\omega, 0) \in B_2$, so $\alpha_1 < \alpha_2 \leq q$. Let $b = (\alpha_1 + \beta_2 + 1, q-1)$. Since $b_1, b_2$ are separated, we have $\alpha_1 + \beta_2 < l_\omega - 1 \leq q-2$, so $b$ is a type I pole. Since the function $\delta_\omega$ verifies $\delta_\omega(j) \leq \delta_\omega(j-1)$, $j = 1, \ldots, l_\omega$, we find

$$\sum_{j=0}^{\alpha_1} \delta_\omega(j) + \sum_{j=0}^{\beta_2} \delta_\omega(l_\omega - j) \leq \sum_{j=0}^{\alpha_1+\beta_2+1} \delta_\omega(j).$$

Let $b_1^* = (\alpha_1, q-1)$ and $b_2^* = (q, \beta_2)$. Then, for $i = 1, 2$ we have $L_\omega(S[b_i]) = L_\omega(S[b_i^*])$ and $U_\omega(S[b_i]) \subseteq U_\omega(S[b_i^*])$; hence, from the previous expressions for $\#U_\omega(S[b_1^*])$, $\#U_\omega(S[b_2^*])$, and $\#U_\omega(S[b])$, given in Lemma 4.20, a simple computation gives

$$\#U_\omega(S[b]) \geq \#U_\omega(S[b_1^*]) + \#U_\omega(S[b_2^*]) \geq \#B_1 + \#B_2 = r.$$

Then, we can take a block $B = \{p_1, \ldots, p_r\} \subseteq U_\omega(S[b])$. Since $\#L_\omega(A[p'_1, \ldots, p'_r]) = \#L(S[b_1]) + \#L_\omega(S[b_2]) - \#S[p] = \#L(S[b_1^*]) + \#L_\omega(S[b_2^*]) - \#S[p]$ where $p = (\alpha_1, \beta_2)$, and again a simple computation shows that

$$\#L_\omega(S[b_1^*]) + \#L_\omega(S[b_2^*]) - \#S[p] \geq \#L_\omega(S[b]) \geq \#L_\omega(A[p_1, \ldots, p_r]),$$

then $B$ is the set we asked for. □

Then, for all $r \geq 1$, there exist poles $p_1, \ldots, p_r \in U_\omega(S)$, forming one block, with $(l_\omega, 0) \in A[p_1, \ldots, p_r]$ or $(0, l_\omega) \in A[p_1, \ldots, p_r]$, and such that $d_r^{ORD}(l) = \#A[p_1, \ldots, p_r]$. If $b = (\alpha, \beta)$ is a vertex of this block, as we know, $L_\omega(A[p_1, \ldots, p_r]) = L_\omega(A[b])$. Additionally, if $\alpha < q$ and $\beta < q-1$, let $b^* = (q, \beta)$ if $(l_\omega, 0) \in A[b]$, and let $b^* = (\alpha, q-1)$ if $(0, l_\omega) \in A[b]$. Then $L_\omega(A[b]) = L_\omega(A[b^*])$. This fact leads us to consider the set

$$\mathcal{B}_\omega = \{p = (x,y) \in U_\omega(S) \mid x \geq q \text{ or } y = q-1\},$$

and for each $r \geq 1$,

$$\mathcal{B}_\omega(r) = \{p \in \mathcal{B}_\omega \mid \#U_\omega(A[p]) \geq r\}.$$

PROPOSITION 4.22. *Assume $l_\omega \leq q-1$. For a given $r$, $1 \leq r \leq g$, let $\rho \in \mathcal{B}_\omega(r)$ be such that*

$$\#L_\omega(A[\rho]) = \min\{\#L_\omega(A[p]) \mid p \in \mathcal{B}_\omega(r)\}.$$

*Then*

$$d_r^{ORD}(l) = r + \#L_\omega(A[\rho]).$$

*Proof.* If this equality were not true, then there would exist poles $p_1, \dots, p_r \in U_\omega(S)$ forming one block and such that $\#L_\omega(A[p_1, \dots, p_r]) < \#L_\omega(A[\rho])$. Let $b$ be a vertex of this block. If $b \in \mathcal{B}_\omega(r)$, then let $b^* = b$; otherwise, if $b \notin \mathcal{B}_\omega(r)$, let $b^* \in \mathcal{B}_\omega(r)$, defined as before. Then $\#U_\omega(A[b^*]) \geq r$ and $\#L_\omega(A[b^*]) = \#L_\omega(A[p_1, \dots, p_r])$ $< \#L_\omega(A[\rho])$, which is impossible by definition of $\rho$. □

After the above result we are able to compute the order bound for the values of $r$ and $l$ in which we are interested. Let us fix the integer $a$, $-1 \leq a \leq q-3$. For a nonnegative integer $j$, let

$$\tau(a,j) = ja + \sigma(j+1).$$

For $1 \leq j \leq q-a-3$ and $0 \leq i \leq j-1$, let $l = g-(a+1)q+\sigma(a+1)+j-i$ (or, equivalently, let $\omega = (q-a-2)q+(j-i)$). Since in this case $l_\omega = q-a-2 \leq q-1$, by using the previous results, next we shall compute $d^{ORD}_{\tau(a,j)+i}(l)$.

THEOREM 4.23. *For $-1 \leq a \leq q-3$, $1 \leq j \leq q-a-3$, and $0 \leq i \leq j-1$, with the restriction $2j-i \leq q-a-3$, we have*

$$d^{ORD}_{\tau(a,j)+i}(g-(a+1)q+\sigma(a+1)+j-i) = (j+1)(q-1)-a+i.$$

*Proof.* We shall use the criterion given in the above proposition, with the help of the formulas stated in Lemma 4.20. Note that, since $w = (q-a-2)q+j-i$, the first $j-i$ poles on the line $x+y = l_\omega = q-a-2$ are in $L_\omega(S)$, and the remaining $l_\omega + 1 - j + i$ poles on this line are in $U_\omega(S)$. This fact, together with the condition $2j \leq q-a+i-3$, allows us the computation of $\delta_\omega(j)$ and, consequently, of the values for the cardinalities of $L_\omega$-sets and $U_\omega$-sets given in Lemma 4.20. Let us proceed in several steps.

First, the minimum value of $\#L_\omega(A[p])$ among the poles $p$ on the line $y = q-1$ such that $\#U_\omega(A[p]) \geq r = \tau(a,j)+i$ is attained at $p_{-1} = (j, q-1)$, for which

$$\#L_\omega(A[p_{-1}]) = (j+1)(q-a-1)-\sigma(j)-(j+1).$$

Let us now study the lines $x = q+t$, $t \geq 0$. If $0 \leq t \leq i$, the minimum value of $\#L_\omega(A[p])$ among the poles $p \in \mathcal{B}_\omega(r)$ on the line $x = q+t$ is attained at $p_t = (q+t, j-t)$, for which

$$\#L_\omega(A[p_t]) = (j+1)(q-a-1)-\sigma(j)-(i+1).$$

Similarly, for $i < t \leq j$, the minimum value of $\#L_\omega(A[p])$ on the line $x = q+t$ is also attained at $p_t = (q+t, j-t)$, and now

$$\#L_\omega(A[p_t]) = (j+1)(q-a-1) - \sigma(j) - t.$$

Finally, for every pole $p$ in the region $x > q+j$, we have $A[p_{-1}] \subset A[p]$, so $L_\omega(A[p_{-1}]) < L_\omega(A[p])$.

Then, by using the notation as in Proposition 4.22, it holds that $\rho = p_{-1} = (j, q-1)$, and

$$\begin{aligned} d^{ORD}_{\tau(a,j)+i}&(g-(a+1)q+\sigma(a+1)+j-i) \\ &= \tau(a,j)+i+\#L_\omega(A[\rho]) \\ &= \tau(a,j)+i+(j+1)(q-a-2)-\sigma(j) \\ &= (j+1)(q-1)-a+i. \qquad \square \end{aligned}$$

Corollary 4.24. *For $-1 \leq a \leq q-3$, $1 \leq j \leq q-a-3$, and $0 \leq i \leq j-1$, with the restriction $2j-i \leq q-a-3$, we have*

$$d_{\tau(a,j)+i}(n+aq+(q-j+i-1)) \geq (j+1)(q-1)-a+i.$$

*Proof.* The proof follows from Theorems 4.2 and 4.23. $\square$

**4.2. The case $l_\omega \geq q$.** The second case we consider is $l_\omega \geq q$. The process and arguments used here are the same as in the former case: for all $r$, we can find a set of poles $p_1, \dots, p_r$ forming one block and such that $d^{ORD}_r(l) = r + \#L_\omega(A[p_1, \dots, p_r])$. Thus, in order to avoid repetition, we state just the main results.

Theorem 4.25. *For $2 \leq a \leq q$, $0 \leq j \leq q-2$, and $0 \leq i \leq j$, with the restriction $2j-i \leq q-a$, we have*

$$d^{ORD}_{\sigma(j)+i+1}(g+(a-1)q+a+j-i-1) = (a+j)q+i-j.$$

Corollary 4.26. *For $2 \leq a \leq q$, $0 \leq j \leq q-2$, and $0 \leq i \leq j$, with the restriction $2j-i \leq q-a$, we have*

$$d_{\sigma(j)+i+1}(n-(a-1)q-a+i-j) \geq (a+j)q+i-j.$$

**5. The weight hierarchy of abundant codes.** Once the bounds given by the order bound are known for some convenient values of $r$ and $l$, in this section we shall compute the hierarchies of all abundant codes, that is, of codes $\mathcal{C}(m)$ for $m = n, \dots, n+(q-2)q$ (note that $\dim \mathcal{C}(n+(q-2)q) = n-1$). Every $m$ in this range can be written as $m = n+aq+b$, with $0 \leq a \leq q-2$ and $-1 \leq b \leq q-2$. If $\alpha$ is the abundance of $\mathcal{C}(m)$, we have to compute $d_1(m), \dots, d_{g-\alpha}(m)$.

Let us recall that, for a nonnegative integer $j$, we write $\sigma(j) = 0+\cdots+j = j(j+1)/2$. Then $g = \sigma(q-1)$, $\rho_{\sigma(j)} = (j-1)(q+1)$, $j = 1, \dots, q$, and $\rho_{\sigma(j)+1} = jq$, $j = 0, \dots, q$. In particular, the abundance of $\mathcal{C}(n+aq)$ is $\sigma(a)+1$, $a = 0, \dots, q-2$. We shall also use the following notation: For $0 \leq a \leq q-2$ and $0 \leq j \leq q-a-1$, we write $\nu(a,j) = \tau(a,j)-1 = j(a+1)+\sigma(j)$.

Let us begin with the case $m = n+aq$, that is, $b = 0$.

Lemma 5.1. *Let $m = n+aq$, $0 \leq a \leq q-2$. If $\alpha$ is the abundance of $\mathcal{C}(m)$, then $\nu(a, q-a-1) > g-\alpha$. Furthermore, $q-a-1$ is the minimum value of $j$ for which this property holds.*

*Proof.*

$$\begin{aligned}\nu(a,q-a-1)-g+\alpha &= (q-a-1)(a+1)+(1+2+\cdots+(q-a-1))\\ &\quad -(1+2+\cdots+(q-1))+(1+2+\cdots+a)+1\\ &= (q-a-1)(a+1)-(q-a-1)a+1\\ &= q-a>0.\end{aligned}$$

In the same way, we obtain $\nu(a,q-a-2)-g+\alpha=-a$. ∎

Lemma 5.2. *Let $m=n+aq$, $0\le a\le q-2$. Then, for $1\le j\le q-a-1$, we have*

$$d_{\nu(a,j)}(m)=jq+j.$$

*Proof.* Let $\alpha$ be the abundance of $\mathcal{C}(m)$. Then $\rho_\alpha=aq$, so $\rho_{\nu(a,j)+\alpha}=(a+j)q+j$ (note that $a+j\le q-1$). Thus

$$n-m+\rho_{\nu(a,j)+\alpha}=-aq+(j+a)q+j=jq+j$$

which verifies the property (*) (see section 2). Then, the result follows from Proposition 2.3. ∎

Lemma 5.3. *Let $m=n+aq$, $0\le a\le q-2$. For $1\le j\le q-a-1$, we have*

$$d_{\tau(a,j-1)}(m)\ge jq-a.$$

*Proof.* Let $m'=n+aq+a$ and $\alpha=\sigma(a+1)$ be the abundance of $\mathcal{C}(m')$. Since $\mathcal{C}(m)=\mathcal{C}(m')$ and $\rho_{\tau(a,j-1)+\alpha}=(a+j)q$, we have

$$d_{\tau(a,j-1)}(m)=d_{\tau(a,j-1)}(m')\ge n-m'+\rho_{\tau(a,j-1)+\alpha}=jq-a$$

and Proposition 2.3 implies the result. ∎

Proposition 5.4. *Let $m=n+aq+b$, $0\le a\le q-2$, $-1\le b\le a$, and let $\alpha$ be the abundance of $\mathcal{C}(m)$. For every integer $r$, $1\le r\le g-\alpha$, let $j$ be the smallest integer such that $r\le\nu(a,j)$. Then*

$$d_r(m)=j(q-a)-\sigma(j)+r.$$

*Proof.* Since $\mathcal{C}(m)=\mathcal{C}(n+aq)$, we can restrict ourselves to the case $b=0$. According to Lemma 5.1, $j\le q-a-1$. Now, Lemmas 5.2 and 5.3 and the monotonicity of the weight hierarchy imply

$$j+a=\nu(a,j)-\tau(a,j-1)\le d_{\nu(a,j)}(m)-d_{\tau(a,j-1)}(m)\le j+a$$

so we get equality. Thus, since $\tau(a,j-1)\le r\le\nu(a,j)$, we have

$$d_r(m)=d_{\nu(a,j)}(m)-(\nu(a,j)-r)=j(q-a)-\sigma(j)+r. \qquad ∎$$

Let us now deal with the case $b\ge a$. Here we can assume $a\le q-3$. Furthermore, if $\alpha$ is the abundance of $\mathcal{C}(n+aq+b)$, then $\nu(a,q-a-2)=g-\alpha$, so it is enough to consider $j$ in the range $1\le j\le q-a-2$.

Lemma 5.5. *Let $m=n+aq+b$ with $0\le a\le q-3$ and $a\le b\le q-2$. For $1\le j\le q-a-2$, we have*

$$d_{\nu(a,j)}(m)=\begin{cases} jq+j+a-b & \text{if } a\le b\le a+j;\\ jq & \text{if } b\ge a+j.\end{cases}$$

*Proof.* Let us assume first $a \le b \le a + j$, and let $\alpha$ be the abundance of $\mathcal{C}(m)$. Then $\rho_\alpha = aq + a$ and $\rho_{\nu(a,j)+\alpha} = (a + j)q + (a + j)$, so

$$n - m + \rho_{\nu(a,j)+\alpha} = jq + j + a - b$$

verifies the condition (*); thus the result follows from Proposition 2.3. If $b \ge a + j$, according to this result and Proposition 5.4,

$$jq = d_{\nu(a,j)}(n + aq + a + j) \ge d_{\nu(a,j)}(m) \ge d_{\nu(a,j)}(n + (a + 1)q) = jq$$

and we have equality. $\square$

PROPOSITION 5.6. *For $0 \le a \le q - 3$, $2 \le j \le q - a - 2$, and $0 \le i \le j - 2$ such that $2j < q - a + i$, we have*

$$d_{\nu(a,j)-a-j+i}(n + aq + (q - j + i)) \ge j(q - 1) - a + i.$$

*Proof.* Since $\tau(a, j - 1) = \nu(a, j) - a - j$, this follows from Corollary 4.24. $\square$

The main result in this section is the following.

THEOREM 5.7. *Let $m = n + aq + b$ with $0 \le a \le q - 2$, $-1 \le b \le q - 2$, and let $\alpha$ be the abundance of $\mathcal{C}(m)$. For every integer $r$, $1 \le r \le g - \alpha$, let $j$ be the smallest integer such that $r \le \nu(a, j)$. Then*

$$d_r(m) = \begin{cases} j(q - a) - \sigma(j) + r & \text{if } -1 \le b \le a; \\ j(q - a) - \sigma(j) + r + a - b & \text{if } a \le b \le a + j; \\ j(q - a - 1) - \sigma(j) + r & \text{if } a + j < b \le \min\{q - 2, q + a + r - \nu(a, j)\}; \\ j(q - a) - \sigma(j) + r + a - b & \text{if } q + a + r - \nu(a, j) < b \le q - 2. \end{cases}$$

*Proof.* The case $b \le a$ was treated in Proposition 5.4. If $a \le b \le a + j$, then $\rho_\alpha = aq + a$, so $\rho_{\tau(a,j-1)+\alpha} = (a + j)q$, and

$$d_{\tau(a,j-1)}(m) \ge n - m + \rho_{\tau(a,j-1)+\alpha} = jq - b.$$

By using the monotonicity of the weight hierarchy and Lemma 5.5,

$$j + a = \nu(a, j) - \tau(a, j - 1) \le d_{\nu(a,j)}(m) - d_{\tau(a,j-1)}(m) \le j + a$$

so we have equality, and since $\tau(a, j - 1) \le r \le \nu(a, j)$,

$$d_r(m) = d_{\nu(a,j)}(m) - (\nu(a, j) - r) = j(q - a) - \sigma(j) + r + a - b.$$

Let us assume now $a + j < b \le \min\{q - 2, q + a + r - \nu(a, j)\}$. If $\nu(a, j) - r \le a + 1$, then $\min\{q - 2, q + a + r - \nu(a, j)\} = q - 2$, and

$$d_r(n + aq + a + j) \ge d_r(m) \ge d_r(n + (a + 1)q).$$

Since $\nu(a, j) - r \le a + 1$ implies $r > \nu(a + 1, j - 1)$, according to the previous results

$$d_r(n + aq + a + j) = d_r(n + (a + 1)q) = j(q - a - 1) - \sigma(j) + r$$

and we have equality in the previous inequality. If $\nu(a, j) - r \ge a + 2$, then $\min\{q - 2, q + a + r - \nu(a, j)\} = q + a + r - \nu(a, j)$. Observe that $\nu(a, 1) = a + 2$, so this case can happen only for $j > 1$. Let $i = r - \nu(a, j) + a + j$. A simple computation shows that $q + a + r - \nu(a, j) = q - j + i$. Furthermore, $0 \le i$ because $j$ is minimum with the property $r \le \nu(a, j)$, and $i \le j - 2$ because $\nu(a, j) - r \ge a + 2$. Since

$r = \nu(a,j) - a - j + i$, we have $q + a + r - \nu(a,j) = q - j + i$. Then, according to Proposition 5.6 and the previous results,

$$\begin{aligned} j(q-a-1) - \sigma(j) + r &= d_r(n + aq + a + j) \geq d_r(m) \\ &= \geq d_r(n + aq + q - j + i) \\ &\geq j(q-a-1) - \sigma(j) + r \end{aligned}$$

and we get equality again.

Finally, let us assume $q + a + r - \nu(a,j) < b \leq q - 2$ (so $\nu(a,j) - r \geq a + 2$ and $j \geq 2$). Since $\nu(a,j) = \tau(a,j-1)+a+j$, we can write $r = \tau(a,j-1)+t$, $0 \leq t \leq j-2$. Conversely, as we know, if $\alpha$ is the abundance of $\mathcal{C}(m)$, then $\rho_{\tau(a,j-1)+\alpha} = (a+j)q$, and hence $\rho_{r+\alpha} = (a+j)q + t$. Consequently,

$$n - m + \rho_{r+\alpha} = (j-1)q + (q - b + t).$$

Now, one easily gets that $0 \leq q - b + t \leq j - 1$, so $n - m + \rho_{r+\alpha}$ verifies the property (*), and

$$d_r(m) = n - m + \rho_{r+\alpha} = j(q-a) - \sigma(j) + r + a - b.$$

This finishes the proof. ∎

*Example* 5.1. The values of $d_r(m)$ in the ranges $n \leq m \leq n+q-1$ and $1 \leq r \leq 9$ are summarized in Table A.1 (Appendix A). The values marked with a star are those obtained with the order bound.

## 6. The weight hierarchy of nonabundant codes.

Let us now study nonabundant codes, that is, codes $\mathcal{C}(m)$ with $m$ in the range $0 \leq m \leq n-1$. In this case it will still be convenient to consider several subcases.

### 6.1. The range $n-q \leq m \leq n-1$.

We can write $m = n-q+b$, $0 \leq b \leq q-1$; that is, we can keep the notation used in the previous section devoted to abundant codes, with $a = -1$.

For $b = q-1$ we have $\mathcal{C}(n-1) = \mathcal{C}(n)$, so the hierarchy of this code is already known.

PROPOSITION 6.1. *For $r = 1, \ldots, g$, we have $d_r(n-1) = \rho_{r+1}$.*

For $0 \leq b \leq q-2$, we can apply the same technique and reasoning as in the case of abundant codes (including the use of Corollary 4.24). Note that $a = -1$ implies $\nu(a,j) = \sigma(j)$. Thus we have the following.

THEOREM 6.2. *Let $m = n - q + b$ with $0 \leq b \leq q-2$. For every integer $r$, $1 \leq r \leq g$, let $j$ be the smallest integer such that $r \leq \sigma(j)$. Then*

$$d_r(m) = \begin{cases} jq - \sigma(j-1) + r - b - 1 & \text{if } 0 \leq b \leq j-1; \\ jq - \sigma(j) + r & \text{if } j - 1 < b \leq \min\{q-2, q-1+r-\sigma(j)\}; \\ jq - \sigma(j-1) + r - b - 1 & \text{if } q - 1 + r - \sigma(j) < b \leq q-2. \end{cases}$$

*Proof.* The proof is straightforward from the proof of Theorem 5.7. ∎

*Example* 6.1. Table A.2 in Appendix A summarizes the values of $d_r(m)$ in the ranges $n - q \leq m \leq n-1$ and $1 \leq r \leq 9$. The values tagged with a star are those obtained with the order bound.

**6.2. The range $n - q^2 \le m < n - q$.** Write $m = n - aq + b$ with $2 \le a \le q$ and $0 \le b \le q - 1$. We have to compute $d_1(m), \dots, d_g(m)$.

LEMMA 6.3. *If $1 \le r \le g$, $n - q^2 \le m < n - q$, and $n - m + \rho_r$ is a pole, then $d_r(m) = n - m + \rho_r$.*

*Proof.* First note that every pole $p = xq + y(q+1) \le n - q^2 + q$ verifies the condition (*), because otherwise $x \ge q^2 - q$ and $y \ge 1$, so

$$p \ge (q^2 - q)q + q + 1 = n - q^2 + q + 1.$$

Since $n - m + \rho_r \le q^2 + \rho_g < n - q^2 + q$, we have the result. ◻

Conversely, according to Corollary 4.26, we have the following.

PROPOSITION 6.4. *For $2 \le a \le q$, $1 \le j \le q-1$, and $1 \le i \le j$ such that $2j < q - a + i$, we have*

$$d_{\sigma(j-1)+i}(n - aq + (q - a + i - j)) \ge (a + j - 1)q + i - j.$$

LEMMA 6.5. *For $1 \le r \le g$, let $j$ be the smallest integer such that $r \le \sigma(j)$. Then $j \le q - 1$.*

*Proof.* $r \le g = \sigma(q-1)$. ◻

THEOREM 6.6. *Let $m = n - aq + b$ with $2 \le a \le q$ and $0 \le b \le q-1$. For every integer $r$, $1 \le r \le g$, let $j$ be the smallest integer such that $r \le \sigma(j)$. Then*

$$d_r(m) = \begin{cases} (a+j-1)q + r - \sigma(j) & \text{if } j - 1 < b \le q - a + r - \sigma(j); \\ (a+j-1)q + r - \sigma(j-1) - b - 1 & \text{otherwise.} \end{cases}$$

*Proof.* If $j$ is the smallest integer such that $r \le \sigma(j)$, then $\rho_r = \rho_{\sigma(j)} - \sigma(j) + r = (j-1)(q+1) - \sigma(j) + r$, and hence $d_r(m) \ge n - m + \rho_r = (a+j-1)q + r - \sigma(j-1) - b - 1$. Let us assume first $0 \le b \le j - 1$. By means of the monotonicity property, it suffices to prove the equality $d_r(m) = n - m + \rho_r$ for $r = \sigma(j)$. Since

$$n - m + \rho_{\sigma(j)} = aq - b + (j-1)q + (j-1) = (a+j-1)q + (j-b-1)$$

is a pole, because $0 \le j - b - 1 \le a + j - 1$, Lemma 6.3 implies the result and equality holds. Let us assume now $b > j - 1$. If $j - 1 < b \le q - a + r - \sigma(j)$, according to Proposition 6.4,

$$\begin{aligned} (a+j-1)q + r - \sigma(j) = d_r(n - aq + j - 1) &\ge d_r(m) \\ &\ge d_r(n - aq + (q - a + r - \sigma(j))) \\ &\ge (a+j-1)q + r - \sigma(j) \end{aligned}$$

and we get equality. If $q - a + r - \sigma(j) < b \le q - a$ (so $r < \sigma(j)$), write $r = \sigma(j-1) + t$, $1 \le t \le j - 1$. Then $\rho_r = (j-1)q + t - 1$ and $n - m + \rho_r = (a+j-2)q + (q - b + t - 1)$. Since $q - a + r - \sigma(j) < b$, then $0 \le q - b + t - 1 \le a + j - 2$, so $n - m + \rho_r$ is a pole and the result follows again from Lemma 6.3. In the same way, if $q - a < b \le q - 1$, then $0 \le q - b \le a - 1$, so $n - m = (a-1)q + (q-b)$ is a pole and thus so is $n - m + \rho_r = (a+j-1)q + r - \sigma(j-1) - b - 1$. ◻

*Example* 6.2. Table A.3 summarizes the values of $d_r(m)$ in the ranges $n - 2q \le m \le n - q - 1$ and $1 \le r \le 9$. The values marked with a star are those obtained with the order bound.

**6.3. The range $2q^2 - 2q < m < n - q^2$.** The weight hierarchy of codes in this range follows very easily from Proposition 2.3. Since $2q^2 - 2q = 2g$, we have to compute $d_1(m), \dots, d_g(m)$.

THEOREM 6.7. *If $2q^2 - 2q < m < n - q^2$, then, for $r = 1, \dots, g$, we have*

$$d_r(m) = n - m + \rho_r.$$

*Proof.* According to Proposition 2.3, it suffices to show that $n - m + \rho_r$ verifies the property (*). Since $n - m + \rho_r \geq n - m > q^2 > 2g$, then $n - m + \rho_r$ is a pole number. Write $n - m + \rho_r = xq + y(q+1)$ with $0 \leq x$, $0 \leq y \leq q - 1$. We have to prove that either $y = 0$ or $x \leq q^2 - q - 1$. If, on the contrary, $y \neq 0$ and $x \geq q^2 - q$, then

$$n - m + \rho_r \geq (q^2 - q)q + q + 1 = n - q^2 + q + 1,$$

and since $\rho_r \leq \rho_g = q^2 - q - 2$,

$$n - m + q^2 - q - 2 \geq n - q^2 + q + 1$$

which leads to $m < 2q^2 - 2q$, in contradiction to our hypothesis on $m$. □

**6.4. The range $0 \leq m \leq 2q^2 - 2q$.** Since $2q^2 - 2q < (n + 2g - 2)/2$, the weight hierarchy of codes in this range can be obtained from the previously computed ones by means of the duality property stated in Corollary 3.7.

*Example* 6.3. The values of $d_r(m) - n + q^2$ in the ranges $2g - 1 \leq m \leq 2g + q - 2$ and $1 \leq r \leq 9$ are summarized in Table A.4. The values marked with a star are those obtained by means of the duality property stated in Corollary 3.7. The remaining values are obtained by using Proposition 2.3. In order to explain how Corollary 3.7 can be used, we shall show the computation of $d_5(2g)$ and $d_6(2g)$. Let $M = n - 2$ and let $K = \dim \mathcal{C}(M) = n - g - 1$. Let $r = \sigma(q - 3) - 2$. According to Theorem 6.2, we have $d_r(M) = q^2 - 3q - 4$ and $d_{r+1}(M) = q^2 - 3q - 1$, so $s_r = 3$ and, according to Corollary 3.7,

$$\begin{aligned} d_5(2g) &= n - d_r(M) - 1 = n - q^2 + 3q + 3, \\ d_6(2g) &= n - d_r(M) = n - q^2 + 3q + 4. \end{aligned}$$

**Appendix A. Tables.**

TABLE A.1

*Values of $d_r(m)$ in the ranges $n \leq m \leq n+q-1$ and $1 \leq r \leq 9$. The values marked with a star are those obtained with the order bound.*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 9 | $3q+3$ | $3q+2$ | $3q+1$ | $3q$ | $3q$ | $\cdots$ | $3q$ | $3q$ | $3q$ |
| 8 | $3q+2$ | $3q+1$ | $3q$ | $3q-1$ | $3q-1$ | $\cdots$ | $3q-1$ | $3q-1$ | $3q-1$ |
| 7 | $3q+1$ | $3q$ | $3q-1$ | $3q-2$ | $3q-2$ | $\cdots$ | $3q-2$ | $3q-2^*$ | $2q+2$ |
| 6 | $3q$ | $3q-1$ | $3q-2$ | $3q-3$ | $3q-3$ | $\cdots$ | $3q-3^*$ | $2q+2$ | $2q+1$ |
| 5 | $2q+2$ | $2q+1$ | $2q$ | $2q$ | $2q$ | $\cdots$ | $2q$ | $2q$ | $2q$ |
| 4 | $2q+1$ | $2q$ | $2q-1$ | $2q-1$ | $2q-1$ | $\cdots$ | $2q-1$ | $2q-1$ | $2q-1$ |
| 3 | $2q$ | $2q-1$ | $2q-2$ | $2q-2$ | $2q-2$ | $\cdots$ | $2q-2$ | $2q-2^*$ | $q+1$ |
| 2 | $q+1$ | $q$ | $q$ | $q$ | $q$ | $\cdots$ | $q$ | $q$ | $q$ |
| 1 | $q$ | $q-1$ | $q-1$ | $q-1$ | $q-1$ | $\cdots$ | $q-1$ | $q-1$ | $q-1$ |
| $r/m$ | $n$ | $n+1$ | $n+2$ | $n+3$ | $n+4$ | $\cdots$ | $n+q-$ | $n+q-2$ | $n+q-1$ |

Table A.2

*Values of $d_r(m)$ in the ranges $n-q \le m \le n-1$ and $1 \le r \le 9$. The values marked with a star are those obtained with the order bound.*

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 9 | $4q+2$ | $4q+1$ | $4q$ | $4q-1$ | $4q-1$ | $\cdots$ | $4q-1$ | $4q-1$ | $4q-1^*$ | $3q+3$ |
| 8 | $4q+1$ | $4q$ | $4q-1$ | $4q-2$ | $4q-2$ | $\cdots$ | $4q-2$ | $4q-2^*$ | $3q+3$ | $3q+2$ |
| 7 | $4q$ | $4q-1$ | $4q-2$ | $4q-3$ | $4q-3$ | $\cdots$ | $4q-3^*$ | $3q+3$ | $3q+2$ | $3q+1$ |
| 6 | $3q+2$ | $3q+1$ | $3q$ | $3q$ | $3q$ | $\cdots$ | $3q$ | $3q$ | $3q$ | $3q$ |
| 5 | $3q+1$ | $3q$ | $3q-1$ | $3q-1$ | $3q-1$ | $\cdots$ | $3q-1$ | $3q-1$ | $3q-1^*$ | $2q+2$ |
| 4 | $3q$ | $3q-1$ | $3q-2$ | $3q-2$ | $3q-2$ | $\cdots$ | $3q-2$ | $3q-2^*$ | $2q+2$ | $2q+1$ |
| 3 | $2q+1$ | $2q$ | $2q$ | $2q$ | $2q$ | $\cdots$ | $2q$ | $2q$ | $2q$ | $2q$ |
| 2 | $2q$ | $2q-1$ | $2q-1$ | $2q-1$ | $2q-1$ | $\cdots$ | $2q-1$ | $2q-1$ | $2q-1^*$ | $q+1$ |
| 1 | $q$ | $q$ | $q$ | $q$ | $q$ | $\cdots$ | $q$ | $q$ | $q$ | $q$ |
| $r/m$ | $n-q$ | $n-q+1$ | $n-q+2$ | $n-q+3$ | $n-q+4$ | $\cdots$ | $n-4$ | $n-3$ | $n-2$ | $n-1$ |

TABLE A.3

*Values of $d_r(m)$ in the ranges $n-2q \leq m \leq n-q-1$ and $1 \leq r \leq 9$. The values marked with a star are those obtained with the order bound.*

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 9 | $5q+2$ | $5q+1$ | $5q$ | $5q-1$ | $\cdots$ | $5q-1$ | $5q-1$ | $5q-1^*$ | $4q+4$ | $4q+3$ |
| 8 | $5q+1$ | $5q$ | $5q-1$ | $5q-2$ | $\cdots$ | $5q-2$ | $5q-2^*$ | $4q+4$ | $4q+3$ | $4q+2$ |
| 7 | $5q$ | $5q-1$ | $5q-2$ | $5q-3$ | $\cdots$ | $5q-3^*$ | $4q+4$ | $4q+3$ | $4q+2$ | $4q+1$ |
| 6 | $4q+2$ | $4q+1$ | $4q$ | $4q$ | $\cdots$ | $4q$ | $4q$ | $4q$ | $4q^*$ | $3q+3$ |
| 5 | $4q+1$ | $4q$ | $4q-1$ | $4q-1$ | $\cdots$ | $4q-1$ | $4q-1$ | $4q-1^*$ | $3q+3$ | $3q+2$ |
| 4 | $4q$ | $4q-1$ | $4q-2$ | $4q-2$ | $\cdots$ | $4q-2$ | $4q-2^*$ | $3q+3$ | $3q+2$ | $3q+1$ |
| 3 | $3q+1$ | $3q$ | $3q$ | $3q$ | $\cdots$ | $3q$ | $3q$ | $3q$ | $3q^*$ | $2q+2$ |
| 2 | $3q$ | $3q-1$ | $3q-1$ | $3q-1$ | $\cdots$ | $3q-1$ | $3q-1$ | $3q-1^*$ | $2q+2$ | $2q+1$ |
| 1 | $2q$ | $2q$ | $2q$ | $2q$ | $\cdots$ | $2q$ | $2q$ | $2q$ | $2q^*$ | $q+1$ |
| $r/m$ | $n-2q$ | $n-2q+1$ | $n-2q+2$ | $n-2q+3$ | $\cdots$ | $n-q-5$ | $n-q-4$ | $n-q-3$ | $n-q-2$ | $n-q-1$ |

TABLE A.4

*Values of $d_r(m) - n + q^2$ in the ranges $2g-1 \leq m \leq 2g+q-2$ and $1 \leq r \leq 9$.*
*The values tagged with a star are those obtained by means of the duality property (section 3).*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 9 | $4q+4$ | $4q+4^*$ | $4q+4^*$ | $4q$ | $\cdots$ | $3q+5$ | $3q+4$ |
| 8 | $4q+3$ | $4q+3^*$ | $4q$ | $4q-1$ | $\cdots$ | $3q+4$ | $3q+3$ |
| 7 | $4q+2$ | $4q$ | $4q-1$ | $4q-2$ | $\cdots$ | $3q+3$ | $3q+2$ |
| 6 | $3q+4$ | $3q+4^*$ | $3q+4^*$ | $3q$ | $\cdots$ | $2q+5$ | $2q+4$ |
| 5 | $3q+3$ | $3q+3^*$ | $3q$ | $3q-1$ | $\cdots$ | $2q+4$ | $2q+3$ |
| 4 | $3q+2$ | $3q$ | $3q-1$ | $3q-2$ | $\cdots$ | $2q+3$ | $2q+2$ |
| 3 | $2q+3$ | $2q+3^*$ | $2q$ | $2q-1$ | $\cdots$ | $q+4$ | $q+3$ |
| 2 | $2q+2$ | $2q$ | $2q-1$ | $2q-2$ | $\cdots$ | $q+3$ | $q+2$ |
| 1 | $q+2$ | $q$ | $q-1$ | $q-2$ | $\cdots$ | 3 | 2 |
| $r/m$ | $2g-1$ | $2g$ | $2g+1$ | $2g+2$ | $\cdots$ | $2g+q-3$ | $2g+q-2$ |

**Appendix B. Some notations used in this paper.**

| | |
|---|---|
| $\mathcal{C}(m)$ | is the Hermitian code, image of the evaluation map that sends $f \in \mathcal{L}(mQ)$ into $(f(P_1), \ldots, f(P_n))$. (section 2) |
| $C_l$ | $= \langle ev(f_1), \ldots, ev(f_l)\rangle^\perp$ Hermitian code. (section 4) |
| $d_r(m)$ | denotes de $r$th generalized weight of the code $\mathcal{C}(m)$. (section 2) |
| $\rho_i$ | is the $i$th element (pole) of the Weierstrass semigroup $S$. (section 4) |
| $\bar{\rho}$ | is the conjugate of a type II pole $\rho$. (Definition 4.3) |
| $A[\rho]$ | $= \{p \in S \mid \text{ there exists } p' \in S \text{ such that } p + p' = \rho\} = \{p \in S \mid \rho - p \in S\}$. $A$-set. (section 4) |
| $A[p_1, \ldots, p_r]$ | $= A[p_1] \cup \cdots \cup A[p_r]$. (section 4) |
| $l_\omega$ | $= w_1 + w_2$, where $\omega = \rho_{l+1} = (w_1, w_2)$. (section 4) |
| $S[\rho]$ | $= \{(\alpha, \beta) \in S \mid \alpha \leq x, \beta \leq y\}$. (Direct) shadow of $\rho = (x, y)$. (Definition 4.4) |
| $S^{-1}[p]$ | $= \{(\alpha, \beta) \in S \mid \alpha > x, \beta > y\}$. Inverse shadow of $p = (x, y) \in \overline{S} = S \cup \pi$, where $\pi = (-1, -1)$. (Definition 4.4) |
| $L_\omega(T)$ | $= \{\rho \in T \mid \rho < \omega\}$. The $\omega$-lower part of a subset $T$ of $S$. (Definition 4.9) |
| $U_\omega(T)$ | $= \{\rho \in T \mid \rho \geq \omega\}$. The $\omega$-upper part of a subset $T$ of $S$. (Definition 4.9) |
| $L_\omega^{(p)}(T)$ | $= L_\omega(T) \cap S^{-1}[p]$. The $\omega$-lower part of $T$ with respect to $p \in \overline{S}$. (Definition 4.9) |
| $U_\omega^{(p)}(T)$ | $= U_\omega(T) \cap S^{-1}[p]$. The $\omega$-upper part of $T$ with respect to $p \in \overline{S}$. (Definition 4.9) |
| $\mathcal{P}_\omega(r, p)$ | $= \{\{p_1, \ldots, p_r\} \subseteq U_\omega^{(p)}(S) \mid \text{ the } p_i\text{'s are all distinct}\}$. (section 4) |
| $\mathcal{P}^*{}_\omega(r, p)$ | $= \{\{p_1, \ldots, p_r\} \in \mathcal{P}_\omega(r, p) \mid \{p_1, \ldots, p_r\} = U_\omega(A[p_1, \ldots, p_r])\}$. (section 4) |
| $d_r^{ORD}(l)$ | $= \min\{\#A[p_1, \ldots, p_r] \mid \{p_1, \ldots, p_r\} \in \mathcal{P}_\omega(r, \pi)\}$. Order bound of the $r$th generalized weight of $C_l$. (section 4) |
| $\mathcal{B}_\omega$ | $= \{p = (x, y) \in U_\omega(S) \mid x \geq q \text{ or } y = q - 1\}$. (subsection 4.1) |
| $\mathcal{B}_\omega(r)$ | $= \{p \in \mathcal{B}_\omega \mid \#U_\omega(A[p]) \geq r\}$. For each $r \geq 1$. (subsection 4.1) |
| $\delta_\omega(j)$ | $= \begin{cases} 0 & \text{if } (j, l_\omega - j) \in U_\omega(S) \\ -1 & \text{if } (j, l_\omega - j) \in L_\omega(S) \end{cases}$ for $j = 0, 1, \ldots, l_\omega$. (subsection 4.1) |
| $\sigma(j)$ | $= 0 + 1 + \cdots + j = \frac{j(j+1)}{2}$, for a nonnegative integer $j$. (subsection 4.1) |
| $\tau(a, j)$ | $= ja + \sigma(j+1)$, for a nonnegative integer $j$, and where $a$ is an integer $-1 \leq a \leq q - 3$. (subsection 4.1) |
| $\nu(a, j)$ | $= \tau(a, j) - 1 = j(a+1) + \sigma(j)$, $0 \leq a \leq q - 2$, and $0 \leq j \leq q - a - 1$. (section 5) |

REFERENCES

[1] P. Heijnen and R. Pellikaan, *Generalized Hamming weights of q-ary Reed-Muller codes*, IEEE Trans. Inform. Theory, 44 (1998), pp. 181–197.

[2] T. Helleseth, T. Kløve, and J. Mykkeleveit, *The weight distribution of irreducible cyclic codes with block lengths $n_1((q^l - 1)/N)$*, Discrete Math., 18 (1977), pp. 179–211.

[3] T. Høholdt, J.H. van Lint, and R. Pellikaan, *Algebraic geometry of codes*, in Handbook of Coding Theory, Vols. I and II, R.A. Brualdi, W.C. Huffman, and V. Pless, eds., North–Holland, Elsevier, Amsterdam, 1998, pp. 871–961.

[4] C. Munuera, *On the generalized Hamming weights of geometric Goppa codes*, IEEE Trans. Inform. Theory, 40 (1994), pp. 2092–2099.

[5] C. Munuera and D. Ramirez, *The second and third generalized Hamming weights of Hermitian codes*, IEEE Trans. Inform. Theory, 45 (1999), pp. 709–713.

[6] H. Stichtenoth, *Self-dual Goppa codes*, J. Pure Appl. Algebra, 55 (1988), pp. 199–211.

[7] M. Tsfasman and S. Vladut, *Geometric approach to higher weights*, IEEE Trans. Inform. Theory, 41 (1995), pp. 1564–1588.

[8] V.K. Wei, *Generalized Hamming weights for linear codes*, IEEE Trans. Inform. Theory, 37 (1991), pp. 1412–1418.

[9] K. Yang and P.V. Kumar, *On the true minimum distance of Hermitian codes*, in Coding Theory and Algebraic Geometry, H. Stichtenoth and M.A. Tsfasman, eds., Lecture Notes in Math. 1518, Springer-Verlag, Heidelberg, 1992, pp. 99–107.

[10] K. Yang, P.V. Kumar, and H. Stichtenoth, *On the weight hierarchy of geometric Goppa codes*, IEEE Trans. Inform. Theory, 40 (1994), pp. 913–920.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 1, pp. 105–138

# WEIGHT DIVISIBILITY OF CYCLIC CODES, HIGHLY NONLINEAR FUNCTIONS ON $\mathbf{F}_{2^m}$, AND CROSSCORRELATION OF MAXIMUM-LENGTH SEQUENCES[*]

ANNE CANTEAUT[†], PASCALE CHARPIN[†], AND HANS DOBBERTIN[‡]

**Abstract.** We study $[2^m-1, 2m]$-binary linear codes whose weights lie between $w_0$ and $2^m-w_0$, where $w_0$ takes the highest possible value. Primitive cyclic codes with two zeros whose dual satisfies this property actually correspond to almost bent power functions and to pairs of maximum-length sequences with preferred crosscorrelation. We prove that, for odd $m$, these codes are completely characterized by their dual distance and by their weight divisibility. Using McEliece's theorem we give some general results on the weight divisibility of duals of cyclic codes with two zeros; specifically, we exhibit some infinite families of pairs of maximum-length sequences which are not preferred.

**Key words.** cyclic codes, weight divisibility, Boolean functions, nonlinearity, almost bent functions, m-sequences, crosscorrelation

**AMS subject classifications.** 11T71, 94B15, 94A55, 94A60

**PII.** S0895480198350057

## 1. Introduction.

This paper presents a new theoretical approach for studying the weight polynomials of binary cyclic codes with two zeros. Using Pless power moment identities [30] and some ideas due to Kasami [18], we point out the essential role played by the weight divisibility of linear codes of length $(2^m-1)$ and dimension $2m$ in the determination of their weight distributions. We especially focus on $[2^m-1, 2m]$-linear codes which are optimal in the following sense: their weights lie in the range $[w_0, \ldots, 2^m-w_0]$, where $w_0$ takes the highest possible value. Most notably we prove that, for odd $m$, these $[2^m-1, 2m]$-optimal codes are completely characterized by their dual distance and by their weight divisibility. The use of McEliece's theorem [24] then reduces the determination of cyclic codes with two zeros whose dual is optimal to a purely combinatorial problem. It especially provides a very fast algorithm for finding such optimal cyclic codes, even for large lengths. These results also enables us to exhibit some infinite families of cyclic codes with two zeros whose dual is not optimal.

This result widely applies in several areas of telecommunications: binary cyclic codes with two zeros whose dual is optimal in the previous sense are especially related to highly nonlinear power functions on finite fields; they also correspond to pairs of maximum-length sequences with preferred crosscorrelation. A function $f$ from $\mathbf{F}_{2^m}$ into $\mathbf{F}_{2^m}$ is said to achieve the highest possible nonlinearity if any nonzero linear combination of its Boolean components is as far as possible from the set of Boolean affine functions with $m$ variables. When $m$ is odd, the highest possible value for the nonlinearity of a function over $\mathbf{F}_{2^m}$ is known and the functions achieving this bound are called almost bent (AB). These functions play a major role in cryptography; in particular, their use in the S-boxes of a Feistel cipher ensures the best resistance to

[*]Received by the editors December 28, 1998; accepted for publication (in revised form) July 6, 1999; published electronically January 13, 2000.
http://www.siam.org/journals/sidma/13-1/35005.html

[†]INRIA, projet CODES, Domaine de Voluceau, BP 105, 78153 Le Chesnay Cedex, France (Anne.Canteaut@inria.fr, Pascale.Charpin@inria.fr).

[‡]German Information Security Agency, P.O. Box 20 03 63, D-53133 Bonn, Germany (dobbertin@skom.rhein.de).

both differential and linear cryptanalyses. It was recently proved [6] that a function $f$ from $\mathbf{F}_{2^m}$ into $\mathbf{F}_{2^m}$ having maximal nonlinearity corresponds to an optimal code of length $(2^m - 1)$ and dimension $2m$. In particular when $f$ is a power function, $f : x \mapsto x^s$, this code is the dual of the cyclic code of length $(2^m - 1)$ whose zeros are $\alpha$ and $\alpha^s$ ($\alpha$ denotes a primitive element of $\mathbf{F}_{2^m}$). Cyclic codes with two zeros whose dual is optimal also appear in the study of maximum-length sequences (also called m-sequences): the exponents $s$ such that the permutation $x \mapsto x^s$ over $\mathbf{F}_{2^m}$ has maximum nonlinearity coincide with the values of the decimations such that the absolute value of the crosscorrelation function between an m-sequence and its decimation by $s$ is minimum. In this case, any two sequences in the set consisting of all time-shifted versions of an m-sequence and all time-shifted versions of its decimation by $s$ can be easily distinguished. Such pairs of m-sequences are then intensively used in many communication systems, especially in code-division multiple access systems.

The next section recalls some properties of binary cyclic codes; it also exhibits the link between the weight distribution of the duals of cyclic codes with two zeros and both concepts of nonlinearity of a power function from $\mathbf{F}_{2^m}$ into $\mathbf{F}_{2^m}$ and of crosscorrelation of a pair of m-sequences. In section 3 we develop a new theoretical tool for studying the weight distribution of some linear codes, which generalizes some ideas due to Kasami. This method provides new characterizations of AB power mappings and of pairs of m-sequences having some 3-valued correlation spectra, which are presented in section 4. These characterizations use the weight divisibility of the duals of cyclic codes with two zeros which can be derived from McEliece's theorem. Section 5 exhibits some general bounds on the weight divisibility of these codes. Most notably we examine the cyclic codes with two zeros whose dual is at most 8-divisible. Section 6 focuses on power functions $x \mapsto x^s$ over $\mathbf{F}_{2^m}$ for odd $m$ when the exponent $s$ can be written as $s = 2^{\frac{m-1}{2}} + 2^i - 1$. This set of exponents contains the values which appear in both Welch's and Niho's conjectures. Here we prove that for most values of $i$, $x \mapsto x^{2^{\frac{m-1}{2}}+2^i-1}$ is not AB on $\mathbf{F}_{2^m}$. In section 7 we give some results on the weight divisibility of the duals of cyclic codes with two zeros of length $(2^m - 1)$ when $m$ is not a prime. This leads to a very simple necessary condition on the values $s$ of the decimations providing preferred pairs of m-sequences; in this case we are able to eliminate most values of $s$. We also give the exact weight divisibility of the duals of the binary cyclic codes of length $(2^m-1)$ with a defining set $\{1, s\}$ when $m$ is even and $s \bmod (2^{\frac{m}{2}} - 1)$ is a power of 2. All of these results notably generalize two recently proved conjectures on the crosscorrelation of pairs of m-sequences [15, 31, 25, 3]. They also imply that the conjectured almost perfect nonlinear (APN) function $x \mapsto x^s$ with $s = 2^{4g}+2^{3g}+2^{2g}+2^g-1$ over $\mathbf{F}_{2^{5g}}$ is not AB. Finally, we give some numerical results which point out that our theoretical results directly provide the weight divisibility of many infinite families of duals of cyclic codes with two zeros.

## 2. Cyclic codes with two zeros and related objects.

We first give two different formulations of the McEliece theorem which enable us to determine the weight divisibility of the dual of a primitive binary cyclic code with two zeros. These results will be extensively used in the paper.

### 2.1. Weight divisibility of cyclic codes.

Here we consider primitive binary cyclic codes. Let $\alpha$ denote a primitive element of $\mathbf{F}_{2^m}$. Any cyclic code $\mathcal{C}$ of length $(2^m - 1)$ can be defined by its generator polynomial whose roots are called the zeros of the code. The defining set of $\mathcal{C}$ is then the set

$$I(\mathcal{C}) = \{i \in \{0, \dots, 2^m - 2\} | \alpha^i \text{ is a zero of } \mathcal{C}\}.$$

Since $\mathcal{C}$ is a binary code, its defining set is a union of 2-cyclotomic cosets modulo $(2^m - 1)$, $Cl(a)$, where $Cl(a) = \{2^j a \bmod (2^m-1)\}$. From now on the defining set of a binary cyclic code of length $(2^m - 1)$ is identified with the representatives of the corresponding 2-cyclotomic cosets modulo $(2^m - 1)$.

Definition 2.1. *A binary code $\mathcal{C}$ is* $2^\ell$-divisible *if the weight of any of its codewords is divisible by $2^\ell$. Moreover, $\mathcal{C}$ is* exactly $2^\ell$-divisible *if, additionally, it contains at least one codeword whose weight is not divisible by $2^{\ell+1}$.*

The following theorem due to McEliece reduces the determination of the exact weight divisibility of binary cyclic codes to a combinatorial problem.

Theorem 2.2 (see [24]). *A binary cyclic code is exactly $2^\ell$-divisible if and only if $\ell$ is the smallest number such that $(\ell+1)$ nonzeros of $\mathcal{C}$ (with repetitions allowed) have product* 1.

We now focus on primitive cyclic codes with two zeros and on the exact weight divisibility of their duals. We denote by $\mathcal{C}_{1,s}$ the binary cyclic code of length $(2^m - 1)$ with defining set $Cl(1) \cup Cl(s)$. The nonzeros of the cyclic code $\mathcal{C}_{1,s}^\perp$ are the elements $\alpha^{-i}$ with

$$i \in Cl(1) \cup Cl(s).$$

Then $(\ell+1)$ nonzeros of $\mathcal{C}_{1,s}^\perp$ have product 1 if and only if there exist $I_1 \subset Cl(s)$ and $I_2 \subset Cl(1)$ with $|I_1| + |I_2| = \ell + 1$ and

$$\begin{aligned} &\prod_{k \in I_1 \cup I_2} \alpha^{-k} = 1 \\ \iff &\sum_{k \in I_1 \cup I_2} k \equiv 0 \bmod (2^m - 1). \end{aligned}$$

We consider both integers $u$ and $v$ defined by their 2-adic expansions: $u = \sum_{i=0}^{m-1} u_i 2^i$ and $v = \sum_{i=0}^{m-1} v_i 2^i$, where

$$u_i = \begin{cases} 1 & \text{if } 2^i s \bmod (2^m-1) \in I_1, \\ 0 & \text{otherwise,} \end{cases}$$

$$v_i = \begin{cases} 1 & \text{if } 2^i \bmod (2^m-1) \in I_2, \\ 0 & \text{otherwise.} \end{cases}$$

Then we have

$$\begin{aligned} \sum_{k \in I_1 \cup I_2} k &\equiv \sum_{i=0}^{m-1} u_i 2^i s + \sum_{i=0}^{m-1} v_i 2^i \bmod (2^m - 1) \\ &\equiv 0 \bmod (2^m - 1). \end{aligned}$$

The size of $I_1$ (resp., $I_2$) corresponds to $w_2(u) = \sum_{i=0}^{m-1} u_i$ (resp., $w_2(v)$) which is the 2-weight of $u$ (resp., $v$). McEliece's theorem can then be formulated as follows.

Corollary 2.3. *The cyclic code $\mathcal{C}_{1,s}^\perp$ of length $(2^m - 1)$ is exactly $2^\ell$-divisible if and only if for all $(u, v)$ such that $0 \le u \le 2^m - 1$, $0 \le v \le 2^m - 1$, and*

$$us + v \equiv 0 \bmod (2^m - 1),$$

*we have $w_2(u) + w_2(v) \ge \ell + 1$.*

Since $v \leq 2^m - 1$, the equation

$$us + v \equiv 0 \bmod (2^m - 1)$$

can be written

$$v = (2^m - 1) - (us \bmod (2^m - 1)).$$

This leads to the following equivalent formulation.

COROLLARY 2.4. *The cyclic code $\mathcal{C}_{1,s}^{\perp}$ of length $(2^m - 1)$ is exactly $2^\ell$-divisible if and only if for all $u$ such that $0 \leq u \leq 2^m - 1$,*

$$w_2(A(u)) \leq w_2(u) + m - 1 - \ell,$$

*where $A(u) = us \bmod (2^m - 1)$.*

**2.2. APN and AB functions.** We now point out that some cryptographic properties of power functions over $\mathbf{F}_{2^m}$ are related to the weight enumerators of cyclic codes with two zeros.

Let $f$ be a function from $\mathbf{F}_2^m$ into $\mathbf{F}_2^m$. For any $(a, b) \in \mathbf{F}_2^m \times \mathbf{F}_2^m$, we define

$$\delta_f(a,b) = \#\{x \in \mathbf{F}_2^m : \; f(x+a) + f(x) = b\},$$
$$\lambda_f(a,b) = |\#\{x \in \mathbf{F}_2^m : \; a \cdot x + b \cdot f(x) = 0\} - 2^{m-1}|,$$

where $\cdot$ is the usual dot product on $\mathbf{F}_2^m$. These values are of great importance in cryptography, especially for measuring the security of an iterated block cipher using $f$ as a round permutation. A differential attack [2, 21] against such a cipher exploits the existence of a pair $(a, b)$ with $a \neq 0$ such that $\delta_f(a, b)$ is high. Similarly, a linear attack [22, 23] is successful if there is a pair $(a, b)$ with $b \neq 0$ such that $\lambda_f(a, b)$ is high. The function $f$ can then be used as a round function of an iterated cipher only if both

$$\delta_f = \max_{a \neq 0} \max_{b} \delta_f(a,b),$$
$$\lambda_f = \max_{a} \max_{b \neq 0} \lambda_f(a,b)$$

are small. Moreover, if $f$ defines the S-boxes of a Feistel cipher, the values of $\delta_f$ and $\lambda_f$ completely determine the complexity of differential and linear cryptanalyses [29, 28].

Since $x$ is a solution of $f(X + a) + f(X) = b$ if and only if $x + a$ is a solution too, $\delta_f$ is always even. Then we have the following.

PROPOSITION 2.5. *For any function $f : \mathbf{F}_2^m \to \mathbf{F}_2^m$,*

$$\delta_f \geq 2.$$

*In the case of equality, $f$ is called almost perfect nonlinear (APN).*

PROPOSITION 2.6 (see [32, 7]). *For any function $f : \mathbf{F}_2^m \to \mathbf{F}_2^m$,*

$$\lambda_f \geq 2^{\frac{m-1}{2}}.$$

*In the case of equality, $f$ is called almost bent (AB).*

Note that this minimum value for $\lambda_f$ can be achieved only if $m$ is odd. For even $m$, some functions with $\lambda_f = 2^{\frac{m}{2}}$ are known and it is highly conjectured that this value is the minimum [31, p. 603].

From now on the vector space $\mathbf{F}_2^m$ is identified with the finite field $\mathbf{F}_{2^m}$. The function $f$ can then be expressed as a unique polynomial of $\mathbf{F}_{2^m}[X]$ of degree at most $(2^m-1)$. Note that the values of $\delta_f$ and $\lambda_f$ are invariant under both right and left compositions by a linear permutation of $\mathbf{F}_{2^m}$. Similarly, if $f$ is a permutation, $\delta_f = \delta_{f^{-1}}$ and $\lambda_f = \lambda_{f^{-1}}$. We can then assume that $f(0) = 0$ without loss of generality. Both APN property and nonlinearity can also be expressed in terms of error-correcting codes. The proofs of the following results are developed by Carlet, Charpin, and Zinoviev in [6].

THEOREM 2.7. *Let $f$ be a function from $\mathbf{F}_{2^m}$ into $\mathbf{F}_{2^m}$ with $f(0) = 0$. Let $\mathcal{C}_f$ be the linear binary code of length $2^m - 1$ defined by the parity-check matrix*

$$H_f = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{2^m-2} \\ f(1) & f(\alpha) & f(\alpha^2) & \cdots & f(\alpha^{2^m-2}) \end{pmatrix}, \tag{2.1}$$

*where each entry is viewed as a binary vector and $\alpha$ is a primitive element of $\mathbf{F}_{2^m}$. Then*

- *If $\dim \mathcal{C}_f = 2^m - 1 - 2m$,*

$$\lambda_f = \max_{c \in \mathcal{C}_f^\perp, c \neq 0} |2^{m-1} - wt(c)|.$$

*In particular, for odd $m$, $f$ is AB if and only if for any nonzero codeword $c \in \mathcal{C}_f^\perp$,*

$$2^{m-1} - 2^{\frac{m-1}{2}} \leq wt(c) \leq 2^{m-1} + 2^{\frac{m-1}{2}}.$$

- *$\lambda_f = 2^{m-1}$ if and only if $\dim \mathcal{C}_f > 2^m - 1 - 2m$ or $\mathcal{C}_f^\perp$ contains the all-one vector.*
- *$f$ is APN if and only if the code $\mathcal{C}_f$ has minimum distance 5.*

In particular, if $f$ is a power function $x \mapsto x^s$ over $\mathbf{F}_{2^m}$, the code $\mathcal{C}_f$ is the cyclic code of length $(2^m-1)$ whose zeros are $\alpha$ and $\alpha^s$.

Tables 2.1 and 2.2 (resp., Tables 2.3 and 2.4) give all known and conjectured values of exponents $s$ (up to equivalence) such that the power function $x \mapsto x^s$ is APN (resp., has the highest known nonlinearity).

TABLE 2.1
*Known and conjectured APN power functions $x^s$ on $\mathbf{F}_{2^m}$ with $m = 2t+1$.*

| | Exponents $s$ | Status |
|---|---|---|
| Quadratic functions | $2^i + 1$ with $\gcd(i, m) = 1$, $1 \leq i \leq t$ | proven [13, 27] |
| Kasami's functions | $2^{2i} - 2^i + 1$ with $\gcd(i, m) = 1$, $2 \leq i \leq t$ | proven [19] |
| Inverse function | $2^{2t} - 1$ | proven [27, 1] |
| Welch's function | $2^t + 3$ | proven [11] |
| Niho's function | $2^t + 2^{\frac{t}{2}} - 1$ if $t$ is even<br>$2^t + 2^{\frac{3t+1}{2}} - 1$ if $t$ is odd | proven [10] |
| Dobbertin's function | $2^{4i} + 2^{3i} + 2^{2i} + 2^i - 1$ if $m = 5i$ | conjectured [10] |

TABLE 2.2
*Known and conjectured APN power functions $x^s$ on $\mathbf{F}_{2^m}$ with $m = 2t$.*

| | Exponents $s$ | Status |
|---|---|---|
| Quadratic functions | $2^i + 1$ with $\gcd(i, m) = 1$, $1 \leq i < t$ | proven [13, 27] |
| Kasami's functions | $2^{2i} - 2^i + 1$ with $\gcd(i, m) = 1$ $2 \leq i < t$ | proven [19, 17] |
| Dobbertin's function | $2^{4i} + 2^{3i} + 2^{2i} + 2^i - 1$ if $m = 5i$ | conjectured [10] |

TABLE 2.3
*Known AB power permutations $x^s$ on $\mathbf{F}_{2^m}$ with $m = 2t + 1$.*

| | Exponents $s$ | Status |
|---|---|---|
| Quadratic functions | $2^i + 1$ with $\gcd(i, m) = 1$, $1 \leq i \leq t$ | proven [13, 27] |
| Kasami's functions | $2^{2i} - 2^i + 1$ with $\gcd(i, m) = 1$ $2 \leq i \leq t$ | proven [19] |
| Welch's function | $2^t + 3$ | proven [5, 4] |
| Niho's function | $2^t + 2^{\frac{t}{2}} - 1$ if $t$ is even $2^t + 2^{\frac{3t+1}{2}} - 1$ if $t$ is odd | proven [16] |

TABLE 2.4
*Known power permutations $x^s$ with highest known nonlinearity on $\mathbf{F}_{2^m}$ with $m = 2t$.*

| Exponents $s$ | Status |
|---|---|
| $2^{2t-1} - 1$ | proven [20] |
| $2^i + 1$ with $\gcd(i, m) = 2$ $1 \leq i < t, m \equiv 2 \bmod 4$ | proven [13] |
| $2^{2i} - 2^i + 1$ with $\gcd(i, m) = 2$ $1 \leq i < t, m \equiv 2 \bmod 4$ | proven [19] |
| $\sum_{i=0}^{t} 2^{ik}$ with $t$ even, $0 < k < t$ such that $\gcd(k, m) = 1$ | proven [12] |
| $2^t + 2^{\frac{t+1}{2}} + 1$ $t$ odd | proven [9] |
| $2^t + 2^{t-1} + 1$ $t$ odd | proven [9] |
| $2^t + 2^{\frac{t}{2}} + 1$ $t \equiv 2 \bmod 4$ | proven [12] |

**2.3. Crosscorrelation of a pair of binary m-sequences.** A binary sequence $(u_i)_{i \geq 0}$ generated by a linear feedback shift register (LFSR) of length $m$ has maximal period when the feedback polynomial of the LFSR is primitive. Such a sequence is called an *m-sequence* of length $(2^m - 1)$. From now on, a binary m-sequence of length $(2^m - 1)$ is identified with the binary vector of length $(2^m - 1)$ consisting of its first $(2^m - 1)$ bits. A further property of m-sequences is that they are almost uncorrelated with their cyclic shifts. This property is important in many communication systems (such as radar communications or transmissions using spread-spectrum techniques) since it is often required that a signal be easily distinguished from any time-shifted version of itself. It is well known that for any m-sequence $u$ of length $(2^m-1)$ there exists a unique $c \in \mathbf{F}_{2^m} \setminus \{0\}$ such that

$$\forall i,\ 0 \le i \le 2^m-2,\ \ u_i = \mathrm{Tr}(c\alpha^i),$$

where $\alpha$ is a root of the feedback polynomial of the LFSR generating $u$ (i.e., $\alpha$ is a primitive element of $\mathbf{F}_{2^m}$) and Tr denotes the trace function from $\mathbf{F}_{2^m}$ to $\mathbf{F}_2$.

When a communication system uses a set of several signals (usually corresponding to different users), it is also required that each of these signals be easily distinguished from any other signal in the set and its time-shifted versions. This property is of great importance, especially in code-division multiple access systems. The distance between a sequence $u$ and all cyclic shifts of another sequence $v$ can be computed with the crosscorrelation function.

Definition 2.8. *Let $u$ and $v$ be two different binary sequences of length $N$. The crosscorrelation function between $u$ and $v$, denoted by $\theta_{u,v}$, is defined as*

$$\theta_{u,v}(\tau) = \sum_{i=0}^{N-1} (-1)^{u_i+v_{i+\tau}}.$$

*The corresponding crosscorrelation spectrum is the vector $(T_{-N},\dots,T_N)$ with*

$$T_\omega = \#\{\tau,\ 0 \le \tau \le N-1,\ \ \theta_{u,v}(\tau)=\omega\}.$$

Since $\theta_{u,v}(\tau) = N - 2wt(u+\sigma^\tau v)$, where $\sigma$ denotes the cyclic shift operator, the above mentioned applications use pairs of sequences $(u,v)$ such that $|\theta_{u,v}(\tau)|$ is small for all $\tau \in \{0,\dots,N-1\}$.

If $u$ and $v$ are two different binary m-sequences of length $(2^m-1)$, there exists an integer $s$ in $\{0,\dots,2^m-2\}$ and a pair $(c_1,c_2)$ of nonzero elements of $\mathbf{F}_{2^m}$ such that

$$\forall i,\ 0 \le i \le 2^m-2,\ \ u_i = \mathrm{Tr}(c_1\alpha^i) \text{ and } v_i = \mathrm{Tr}(c_2\alpha^{si})\ .$$

If $c_1 = c_2$, the sequence $v$ is said to be a *decimation by $s$ of $u$*. Writing $c_1 = \alpha^{j_1}$ and $c_2 = \alpha^{j_2}$, the crosscorrelation function for the pair $(u,v)$ is given by

$$\theta_{u,v}(\tau) = \sum_{i=0}^{2^m-2} (-1)^{\mathrm{Tr}(\alpha^{i+j_1}+\alpha^{si+j_2+\tau})} = \sum_{x\in\mathbf{F}_{2^m}^*} (-1)^{\mathrm{Tr}(\alpha^{\tau'}[\alpha^{j_1-\tau'}x+x^s])}\ ,$$

where $\tau' = j_2+\tau$. It follows that the corresponding crosscorrelation spectrum does not depend on the choice of $j_2$. It is then sufficient to study the pairs $(u,v)$, where $v$ is a decimation by $s$ of $u$.

We now recall the well-known link between the crosscorrelation spectrum of pairs of binary m-sequences and the weight distribution of the duals of some cyclic codes with two zeros.

Proposition 2.9. *Let $m$ and $s$ be two positive integers such that $\gcd(s,2^m-1)=1$ and $s$ is not a power of 2. Let $(T_{-2^m+1},\dots,T_{2^m-1})$ be the crosscorrelation spectrum between an m-sequence of length $(2^m-1)$ and its decimation by $s$. Let $(A_0,\dots,A_{2^m-1})$ be the weight enumerator of the dual of the binary cyclic code $\mathcal{C}_{1,s}$ of length $(2^m-1)$*

*with defining set $\{1, s\}$. Then we have that $T_\omega = 0$ for all even values of $\omega$ and for odd $\omega$:*

$$\begin{aligned} T_\omega &= A_{2^{m-1}-\frac{\omega+1}{2}} \text{ for all } \omega \notin \{-1, 2^m-1\},\\ T_{-1} &= A_{2^{m-1}} - 2(2^m-1),\\ T_{2^m-1} &= 0. \end{aligned}$$

*In particular, if $\omega_0$ is the smallest positive integer such that*

$$T_{-\omega} = T_\omega = 0 \text{for all } \omega \text{ such that } \omega > \omega_0,$$

*then we have*

$$\omega_0 = 2^m - 1 - 2w_0,$$

*where $w_0$ is the largest integer such that $0 < w_0 \le 2^{m-1}$ and*

$$A_w = A_{2^m-w} = 0 \text{ for all } w \text{ such that } 0 < w < w_0.$$

*Proof.* The code $\mathcal{C}_{1,s}^\perp$ consists of all codewords $c = (c_0, \dots, c_{2^m-2})$, where

$$c_i = \mathrm{Tr}(a\alpha^i + b\alpha^{si}) \text{ with } a, b \in \mathbf{F}_{2^m}.$$

If either $a$ or $b$ equals zero, $c$ is a codeword of the simplex code of length $(2^m-1)$. In this case, its Hamming weight is $2^{m-1}$ except for $a = b = 0$. Using that $\theta_{u,v}(\tau) = 2^m - 1 - 2wt(u + \sigma^\tau v)$, we obtain the expected result. ☐

Using Theorem 2.7 we see that

$$\begin{aligned} \omega_0 &= -1 + 2 \max_{c \in \mathcal{C}_{1,s}^\perp, c \ne 0} |2^{m-1} - wt(c)|\\ &= 2\lambda_s - 1, \end{aligned}$$

where $\lambda_s$ is the linearity of the power permutation $x \mapsto x^s$ over $\mathbf{F}_{2^m}$. In particular, when $m$ is odd the lowest possible value for $\omega_0$ is $2^{\frac{m+1}{2}} - 1$ and the values of $s$ for which this bound is reached exactly correspond to the exponents $s$ such that $x \mapsto x^s$ is an AB permutation over $\mathbf{F}_{2^m}$.

## 3. Weight polynomials of linear codes of length $2^m - 1$ and dimension $2m$.

As previously seen, both notions of nonlinearity of a function from $\mathbf{F}_{2^m}$ into $\mathbf{F}_{2^m}$ and of crosscorrelation spectrum of a pair of binary m-sequences of length $(2^m-1)$ are related to the weight polynomials of some linear binary codes of length $(2^m-1)$ and dimension $2m$. Here we give some general results on the weight distributions of linear codes having these parameters. Our method uses Pless power moment identities [30] and some ideas due to Kasami [18, Thm. 13] (see also [6, Thm. 4]).

Theorem 3.1. *Let $\mathcal{C}$ be a $[2^m-1, 2m]$-linear code which does not contain the all-one vector $\mathbf{1} = (1, \dots, 1)$. Assume that the minimum distance of the dual code $\mathcal{C}^\perp$ is at least 3. Let $A = (A_0, \dots, A_{2^m-1})$ (resp., $B = (B_0, \dots, B_{2^m-1})$) be the weight enumerator of $\mathcal{C}$ (resp., $\mathcal{C}^\perp$). Then, for any positive integer $x \le 2^{m-1}$, we have*

$$\begin{aligned} &\sum_{w=1}^{2^{m-1}-1} (w - 2^{m-1})^2 \left((w-2^{m-1})^2 - x^2\right)(A_w + A_{2^m-w})\\ &\qquad = 2^{2m-2}\left[6(B_3+B_4) + (2^m-1)(2^{m-1}-x^2)\right]. \end{aligned}$$

*Most notably this implies the following:*

(i) *If $w_0$ is such that for all $0 < w < w_0$*

$$A_w = A_{2^m - w} = 0,$$

*then*

$$6(B_3 + B_4) \le (2^m - 1)\left[(2^{m-1} - w_0)^2 - 2^{m-1}\right],$$

*where equality holds if and only if $A_w = 0$ for all $w \notin \{0, w_0, 2^{m-1}, 2^m - w_0\}$.*

(ii) *If $w_1$ is such that for all $w_1 < w < 2^{m-1}$*

$$A_w = A_{2^m - w} = 0,$$

*then*

$$6(B_3 + B_4) \ge (2^m - 1)\left[(2^{m-1} - w_1)^2 - 2^{m-1}\right],$$

*where equality holds if and only if $A_w = 0$ for all $w \notin \{0, w_1, 2^{m-1}, 2^m - w_1\}$.*

*Proof.* The main part of the proof relies on the first Pless power moment identities [30]. The first four power moment identities on the weight distribution of a code of length $2^m - 1$ and dimension $2m$ are

$$\begin{aligned}
&\sum_{w=0}^{n} wA_w = 2^{2m-1}(2^m - 1),\\
&\sum_{w=0}^{n} w^2A_w = 2^{3m-2}(2^m - 1),\\
&\sum_{w=0}^{n} w^3A_w = 2^{2m-3}\left((2^m - 1)^2(2^m + 2) - 3!B_3\right),\\
&\sum_{w=0}^{n} w^4A_w = 2^{2m-4}\left(2^m(2^m - 1)(2^{2m} + 3\cdot 2^m - 6) + 4!\,(B_4 - (2^m - 1)B_3)\right).
\end{aligned}$$

Let us consider the numbers $I_\ell = \sum_{w=1}^{2^m-1}(w - 2^{m-1})^\ell A_w$. Since for $\ell$ even

$$(w - 2^{m-1})^\ell = ((2^m - w) - 2^{m-1})^\ell,$$

we have for any even $\ell$

$$I_\ell = \sum_{w=1}^{2^m-1}(w - 2^{m-1})^\ell A_w = \sum_{w=1}^{2^{m-1}-1}(w - 2^{m-1})^\ell(A_w + A_{2^m-w}).$$

Note that the codeword of weight zero is not taken into account in the sum above. Recall that $C$ does not contain the all-one codeword.

By using the four power moments, we obtain the following values for $I_2$ and $I_4$:

$$\begin{aligned}
I_2 &= 2^{2m-2}(2^m - 1),\\
I_4 &= 2^{2m-2}\left[6(B_3 + B_4) + 2^{m-1}(2^m - 1)\right].
\end{aligned}$$

Let $x$ be a positive integer and let $\mathcal{I}(x)$ denote the value of $I_4 - x^2I_2$. We have

$$\begin{aligned}
\mathcal{I}(x) &= \sum_{w=1}^{2^{m-1}-1}(w - 2^{m-1})^2\left((w - 2^{m-1})^2 - x^2\right)(A_w + A_{2^m-w})\\
&= 2^{2m-2}\left[6(B_3 + B_4) + (2^m - 1)(2^{m-1} - x^2)\right].
\end{aligned}$$

The $w$th term in this sum satisfies

$$(w - 2^{m-1})^2 \left((w - 2^{m-1})^2 - x^2\right) \begin{array}{ll} < 0 & \text{if } 0 < |2^{m-1} - w| < x \\ = 0 & \text{if } w \in \{2^{m-1}, 2^{m-1} \pm x\} \\ > 0 & \text{if } |2^{m-1} - w| > x. \end{array}$$

This implies that, if $A_w = A_{2^m - w} = 0$ for all $w$ such that $0 < w < w_0$, all the terms in $\mathcal{I}(2^{m-1} - w_0)$ are nonpositive. Then we have

$$6(B_3 + B_4) + (2^m - 1)\left[2^{m-1} - (2^{m-1} - w_0)^2\right] \le 0$$

with equality if and only if all terms in the sum are zero. This can happen only when $A_w = 0$ for all $w \notin \{0, w_0, 2^{m-1}, 2^m - w_0\}$.

Similarly, if $A_w = A_{2^m - w} = 0$ for all $w$ such that $w_1 < w < 2^{m-1}$, all the terms in $\mathcal{I}(2^{m-1} - w_1)$ are positive. Then we have

$$6(B_3 + B_4) + (2^m - 1)\left[2^{m-1} - (2^{m-1} - w_1)^2\right] \ge 0$$

with equality if and only if all terms in the sum are zero, i.e., if $A_w = 0$ for all $w \notin \{0, w_1, 2^{m-1}, 2^m - w_1\}$. $\square$

This theorem obviously gives an upper bound on the value of $w_0$ for which all nonzero weights of $\mathcal{C}$ lie between $w_0$ and $2^m - w_0$.

Corollary 3.2. *Let $\mathcal{C}$ be a $[2^m - 1, 2m]$-linear code which does not contain the all-one vector $\mathbf{1} = (1, \ldots, 1)$. Assume that the minimum distance of the dual code $\mathcal{C}^\perp$ is at least 3. Let $w_0$ be the smallest $w$ such that $0 < w < 2^{m-1}$ and*

$$A_w + A_{2^m - w} \neq 0.$$

*Then*

$$w_0 \le 2^{m-1} - 2^{\frac{m-1}{2}}$$

*and equality holds if and only if the weight of every codeword in $\mathcal{C}$ belongs to $\{0, 2^{m-1}, 2^{m-1} \pm 2^{\frac{m-1}{2}}\}$. In this case the weight distribution of $\mathcal{C}$ is the same as the weight distribution of the dual of the 2-error-correcting Bose–Chaudhuri-Hocquenghem (BCH) code, i.e.,*

| *Weight: $w$* | *Number of words: $A_w$* |
|---|---|
| $0$ | $1$ |
| $2^{m-1} - 2^{(m-1)/2}$ | $(2^m - 1)(2^{m-2} + 2^{(m-3)/2})$ |
| $2^{m-1}$ | $(2^m - 1)(2^{m-1} + 1)$ |
| $2^{m-1} + 2^{(m-1)/2}$ | $(2^m - 1)(2^{m-2} - 2^{(m-3)/2})$ |

*Proof.* The first statement directly results from the previous theorem. Moreover, if $w_0 = 2^{m-1} - 2^{\frac{m-1}{2}}$, then $A_{w_0}$, $A_{2^{m-1}}$, and $A_{2^m - w_0}$ are the only unknown values in the weight distribution of $\mathcal{C}$. They are then completely determined by inverting the linear system formed by $A_{w_0} + A_{2^{m-1}} + A_{2^m - w_0} = 2^{2m} - 1$ and the first two Pless power moments identities. $\square$

Pless power moment identities also imply that if all nonzero codewords of a $[2^m - 1, 2m]$ code have Hamming weight in the set $\{2^{m-1} \pm W, 2^{m-1}\}$, the weight distribution of this code is unique as long as its dual has minimum distance at least 3.

Proposition 3.3. *Let $\mathcal{C}$ be a $[2^m-1,2m]$-linear code which does not contain the all-one vector $\mathbf{1}=(1,\dots,1)$. Assume that the minimum distance of the dual code $\mathcal{C}^\perp$ is at least 3. Assume that the weight of every codeword in $\mathcal{C}$ lies in $\{0,2^{m-1},2^{m-1}\pm W\}$. Then $W$ is divisible by $2^{\lceil\frac{m-1}{2}\rceil}$. Moreover, the weight distribution of $\mathcal{C}$ is completely determined:*

| *Weight: $w$* | *Number of words: $A_w$* |
|---|---|
| $0$ | $1$ |
| $2^{m-1}-W$ | $\frac{2^{m-2}(2^m-1)(2^{m-1}+W)}{W^2}$ |
| $2^{m-1}$ | $\frac{(2^m-1)((2^m+1)W^2-2^{2m-2})}{W^2}$ |
| $2^{m-1}+W$ | $\frac{2^{m-2}(2^m-1)(2^{m-1}-W)}{W^2}$ |

*In particular, the number $B_3$ (resp., $B_4$) of codewords of weight 3 (resp., 4) in $\mathcal{C}^\perp$ is given by*

$$B_3=\frac{(2^m-1)(W^2-2^{m-1})}{3\cdot 2^{m-1}},$$
$$B_4=\frac{(2^m-1)(2^{m-2}-1)(W^2-2^{m-1})}{3\cdot 2^{m-1}}.$$

*Proof.* Let $A=(A_0,\dots,A_{2^m-1})$ (resp., $B=(B_0,\dots,B_{2^m-1})$) be the weight enumerator of $\mathcal{C}$ (resp., $\mathcal{C}^\perp$). By hypothesis, $B_1=B_2=0$. Using the first two Pless power moment identities, we obtain

$$\begin{aligned}A_{2^{m-1}-W}+A_{2^{m-1}}+A_{2^{m-1}+W}&=2^{2m}-1,\\(2^{m-1}-W)A_{2^{m-1}-W}+2^{m-1}A_{2^{m-1}}+(2^{m-1}+W)A_{2^{m-1}+W}&=2^{2m-1}(2^m-1),\\(2^{m-1}-W)^2A_{2^{m-1}-W}+2^{2m-2}A_{2^{m-1}}+(2^{m-1}+W)^2A_{2^{m-1}+W}&=2^{3m-2}(2^m-1).\end{aligned}$$

Inverting this linear system gives the expected weight distribution. By writing the third and fourth Pless power moment identities, we can compute the number of codewords of weights 3 and 4 in $\mathcal{C}^\perp$:

$$B_3=\frac{(2^m-1)(W^2-2^{m-1})}{3\cdot 2^{m-1}},$$
$$B_4=\frac{(2^m-1)(2^{m-2}-1)(W^2-2^{m-1})}{3\cdot 2^{m-1}}.$$

Since $B_3$ is an integer, we have that $2^{m-1}$ divides $W^2-2^{m-1}$ and then $W$ is divisible by $2^{\lceil\frac{m-1}{2}\rceil}$. □

## 4. AB functions, 3-valued crosscorrelation functions, and weight divisibility.

**4.1. A new characterization of AB functions.** Let us now suppose that $m$ is odd, $m=2t+1$. We give a necessary and sufficient condition on $f:\mathbf{F}_{2^m}\to\mathbf{F}_{2^m}$ to achieve the highest possible nonlinearity.

Theorem 4.1. *Let $m$ be an odd integer and $\mathcal{C}$ be a $[2^m-1,2m]$-linear code which does not contain the all-one vector $\mathbf{1}=(1,\dots,1)$. Assume that the minimum distance* *of the dual code $\mathcal{C}^\perp$ is at least* 3. *Let $A = (A_0, \ldots, A_{2^m-1})$ be the weight enumerator of $\mathcal{C}$ and let $w_0$ be the smallest $w$ such that $0 < w < 2^{m-1}$ and*

$$A_w + A_{2^m-w} \neq 0.$$

*Then*

$$w_0 = 2^{m-1} - 2^{\frac{m-1}{2}}$$

*if and only if the minimum distance of the dual code $\mathcal{C}^\perp$ is* 5 *and $\mathcal{C}$ is $2^{\frac{m-1}{2}}$-divisible.*

*Proof.* This condition is sufficient because, if

$$w_0 = 2^{m-1} - 2^{\frac{m-1}{2}},$$

the code $\mathcal{C}$ has the same weight distribution as the 2-error-correcting BCH code (according to Corollary 3.2).

This condition is also necessary since, for any $w$ such that $2^{m-1} - 2^{\frac{m-1}{2}} < w < 2^{m-1}$, both integers $w$ and $2^{m-1} - w$ are not divisible by $2^{\frac{m-1}{2}}$. The condition on the divisibility of the weights of $\mathcal{C}$ then implies that

$$A_w + A_{2^m-w} = 0 \text{ for all } w \text{ such that } 2^{m-1} - 2^{\frac{m-1}{2}} < w < 2^{m-1} .$$

If $B_3 = B_4 = 0$, the lower bound given in Theorem 3.1(ii) (applied with $w_1 = 2^{m-1} - 2^{\frac{m-1}{2}}$) is reached. Then the weight of every codeword in $\mathcal{C}$ lies in $\{0, 2^{m-1}, 2^{m-1} \pm 2^{\frac{m-1}{2}}\}$. $\square$

Corollary 4.2. *Let $m$ be an odd integer and let $f$ be a function from $\mathbf{F}_2^m$ into $\mathbf{F}_2^m$ such that $\lambda_f \neq 2^{m-1}$. Then $f$ is AB if and only if $f$ is APN and the code $\mathcal{C}_f^\perp$ defined in Theorem* 2.7 *is $2^{\frac{m-1}{2}}$-divisible.*

When $f$ is a power function, $f : x \mapsto x^s$, the corresponding code $C_f$ is the binary cyclic code $\mathcal{C}_{1,s}$ of length $(2^m - 1)$ with defining set $\{1, s\}$. The weight divisibility of the corresponding dual code can therefore be obtained by applying McEliece's theorem, as expressed in Corollary 2.4. This leads to the following characterization of AB power functions.

Corollary 4.3. *Let $m = 2t+1$. Assume that the power function $f : x \mapsto x^s$ on $\mathbf{F}_{2^m}$ satisfies $\lambda_f \neq 2^{m-1}$. Then $f$ is AB on $\mathbf{F}_{2^m}$ if and only if $f$ is APN on $\mathbf{F}_{2^m}$ and*

$$\forall u, \;\; 1 \leq u \leq 2^m - 1, \;\; w_2(A(u)) \leq t + w_2(u), \tag{4.1}$$

*where $A(u) = us \bmod (2^m - 1)$.*

Note that $\lambda_f = 2^{m-1}$ means that there exists a linear combination of the Boolean components of $f$ which is an affine function.

Condition (4.1) is obviously satisfied when $w_2(u) \geq t+1$. Moreover, if $\gcd(s, 2^m - 1) = 1$ (i.e., if $x \mapsto x^s$ is a permutation), the condition also holds for all $u$ such that $w_2(u) = t$. Using that

$$A(u2^i \bmod (2^m - 1)) = 2^i A(u) \bmod (2^m - 1),$$

we deduce that condition (4.1) must be checked only for one element in each cyclotomic coset. Note that if $u$ is the smallest element in its cyclotomic coset and $w_2(u) < t$, we have $u \leq 2^{m-2} - 1$.

This result provides a fast algorithm for checking whether an APN power function is AB and then for finding all AB power functions on $\mathbf{F}_{2^m}$. There are roughly $\frac{2^{m-1}}{m}$

cyclotomic representatives $u$ such that $w_2(u) \leq t$, and each test requires one modular multiplication on $m$-bit integers and two weight computations. Condition (4.1) can then be checked with approximately $2^m$ elementary operations and at no memory cost.

**4.2. A new characterization of some pairs of m-sequences with 3-valued crosscorrelation.** We now suppose that $m$ is even, $m = 2t$, and we are interested in the values of $s$ such that the crosscorrelation function between an m-sequence of length $(2^m - 1)$ and its decimation by $s$ takes the following three values: $-1, 2W - 1$, and $-2W - 1$. From Proposition 2.9, this means that the weight of every codeword in $\mathcal{C}_{1,s}^{\perp}$ lies in $\{0, 2^{m-1}, 2^{m-1} + W, 2^{m-1} - W\}$. Proposition 3.3 then implies that $W$ is divisible by $2^t$. When $W$ is a power of 2, we have $W = 2^{t+e}$. The $[2^m - 1, 2m]$-codes whose weights lie in $\{0, 2^{m-1}, 2^{m-1} \pm 2^{t+e}\}$ are then completely characterized by their weight divisibility and by the number of codewords of weights 3 and 4 in their dual.

THEOREM 4.4. *Let $m = 2t$ be an even integer and $\mathcal{C}$ be a $[2^m - 1, 2m]$-linear code which does not contain the all-one vector $\mathbf{1} = (1, \ldots, 1)$. Assume that the minimum distance of the dual code $\mathcal{C}^{\perp}$ is at least 3. Let $A = (A_0, \ldots, A_{2^m-1})$ be the weight enumerator of $\mathcal{C}$ and let $e$ be an integer. The weight of every codeword in $\mathcal{C}$ lies in $\{0, 2^{m-1}, 2^{m-1} \pm 2^{t+e}\}$ if and only if $\mathcal{C}$ is $2^{t+e}$-divisible and the number $B_3$ (resp., $B_4$) of codewords of weight 3 (resp., 4) in $\mathcal{C}^{\perp}$ satisfies*

$$B_3 = \frac{(2^m - 1)(2^{2e+1} - 1)}{3},$$
$$B_4 = \frac{(2^m - 1)(2^{m-2} - 1)(2^{2e+1} - 1)}{3}.$$

*Proof.* Applying Proposition 3.3 shows that this condition is sufficient. It is also necessary since, for any $w$ such that $2^{m-1} - 2^{t+e} < w < 2^{m-1}$, both integers $w$ and $2^{m-1} - w$ are not divisible by $2^{t+e}$. The condition on the divisibility of the weights of $\mathcal{C}$ then implies that

$$A_w + A_{2^m - w} = 0 \text{ for all } w \text{ such that } 2^{m-1} - 2^{t+e} < w < 2^{m-1}.$$

For the given values of $B_3$ and $B_4$, we have

$$6(B_3 + B_4) = 2^{m-1}(2^m - 1)(2^{2e+1} - 1).$$

It follows that the lower bound given in Theorem 3.1(ii) (applied with $w_1 = 2^{m-1} - 2^{t+e}$) is reached. The weight of every codeword in $\mathcal{C}$ then lies in $\{0, 2^{m-1}, 2^{m-1} \pm 2^{t+e}\}$ and there is at least a codeword in $\mathcal{C}$ having any of these weights since a code with these parameters has at least three different nonzero weights [15, Thm. 4.1]. ∎

By applying McEliece's theorem (Corollary 2.4), we derive a necessary and sufficient condition to obtain a pair of m-sequences with crosscorrelation values in $\{-1, -1 \pm 2^{t+1+e}\}$. Note that if $e = 0$, $\omega_0 = \max_\tau |\theta_{u,v}(\tau)|$ equals $2^{t+1} - 1$, which is the smallest known value for $\omega_0$. In this case, the 3-valued crosscorrelation function is said to be *preferred* and the corresponding $(u, v)$ is called a *preferred pair of m-sequences.*

COROLLARY 4.5. *Let $m = 2t$ be an even integer and $s$ a positive integer such that* $\gcd(s, 2^m - 1) = 1$ *and $s$ is not a power of 2. Let $e$ be a positive integer and $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$. The crosscorrelation function between an m-sequence of length $(2^m - 1)$ and its decimation by $s$ takes exactly 3 values, $-1$, $-1 + 2^{t+1+e}$, and $-1 - 2^{t+1+e}$ if and only if the number $B_3$ (resp., $B_4$)*

*of codewords of weight* 3 *(resp.,* 4*) in* $\mathcal{C}_{1,s}$ *satisfies*

$$B_3 = \frac{(2^m-1)(2^{2e+1}-1)}{3},$$
$$B_4 = \frac{(2^m-1)(2^{m-2}-1)(2^{2e+1}-1)}{3},$$

*and*

$$\forall u,\ \ 1 \le u \le 2^m-1,\ \ w_2(A(u)) \le t-1-e+w_2(u),$$

*where* $A(u) = us \bmod (2^m-1)$.

*In particular, the crosscorrelation function is preferred if and only if*

$$B_3 = \frac{(2^m-1)}{3},\ \ B_4 = \frac{(2^m-1)(2^{m-2}-1)}{3},$$

*and*

$$\forall u,\ \ 1 \le u \le 2^m-1,\ \ w_2(A(u)) \le t-1+w_2(u).$$

**5. Some general results on the weight divisibility of $\mathcal{C}_{1,s}^{\perp}$.** We now show that the weight divisibility of the dual of the cyclic code $\mathcal{C}_{1,s}$ is conditioned by the 2-weight of $s$. Most notably this implies that $\mathcal{C}_{1,s}^{\perp}$ has a low exact weight divisibility only for particular values of $s$.

**5.1. Upper and lower bounds on the weight divisibility of $\mathcal{C}_{1,s}^{\perp}$.** We first give some general lower and upper bounds on the exact divisibility of $\mathcal{C}_{1,s}^{\perp}$ which depend on the 2-weight of $s$. The 2-weight of $s$ obviously gives an upper bound on the weight divisibility of $\mathcal{C}_{1,s}^{\perp}$ (obtained for $u=1$ in Corollary 2.4). Using this result, we immediately recover the condition on the degree of AB functions given in [6, Thm. 1] in the particular case of power functions.

COROLLARY 5.1. *Let* $\mathcal{C}_{1,s}$ *be the binary cyclic code of length* $(2^m-1)$ *with defining set* $\{1,s\}$*. If* $\mathcal{C}_{1,s}^{\perp}$ *is* $2^\ell$*-divisible, then*

$$\ell \le m - w_2(s).$$

*Most notably this implies the following:*

- *For odd* $m$*, if the power permutation* $f : x \mapsto x^s$ *is AB on* $\mathbf{F}_{2^m}$*, then*

$$w_2(s) \le \frac{m+1}{2};$$

- *for even* $m$*, if the crosscorrelation function between an m-sequence of length* $(2^m-1)$ *and its decimation by s takes the values* $-1, 2^{\frac{m}{2}+1+e}-1, -2^{\frac{m}{2}+1+e}-1$*, then*

$$w_2(s) \le \frac{m}{2} - e.$$

The 2-weight of $s$ also provides a lower bound on the divisibility of $\mathcal{C}_{1,s}^{\perp}$: the cyclic code $\mathcal{C}_{1,s}^{\perp}$ of length $(2^m-1)$ is a subcode of the punctured Reed–Muller code of length $(2^m-1)$ and of order $w_2(s)$, which is exactly $2^\ell$-divisible with $\ell = \lfloor \frac{m-1}{w_2(s)} \rfloor$. When $\gcd(2^m-1, s) = 1$, this bound can be slightly improved.

Corollary 5.2. *Let $m$ and $s$ be two positive integers such that $s \leq 2^m - 1$ and* $\gcd(2^m-1, s) = 1$. *Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$. Then $\mathcal{C}_{1,s}^{\perp}$ is $2^{\ell}$-divisible with $\ell = \lceil \frac{m-1}{w_2(s)} \rceil$.*

*Proof.* We use McEliece's theorem as expressed in Corollary 2.3. Let $u$ and $v$ be two integers in $\{0, \dots, 2^m - 1\}$ such that $us + v \equiv 0 \bmod (2^m - 1)$. Here we prove that

$$w_2(u) + w_2(v) \geq \left\lceil \frac{m-1}{w_2(s)} \right\rceil + 1. \tag{5.1}$$

Since $s \leq 2^m - 2$, we have that

$$us + v = K(2^m - 1), \text{ where } K \text{ lies in } \{1, \dots, 2^m - 1\}.$$

By writing

$$K(2^m - 1) = (K-1)2^m + [(2^m - 1) - (K - 1)],$$

we obtain that

$$w_2(K(2^m - 1)) = w_2(K-1) + m - w_2(K-1) = m$$

since $0 \leq K - 1 \leq 2^m - 1$. It follows that $w_2(us + v) = m$. We now write that

$$w_2(us + v) \leq w_2(us) + w_2(v) \leq w_2(u)w_2(s) + w_2(v)$$

and we get

$$w_2(u)w_2(s) + w_2(v) \geq m.$$

We therefore obtain

$$w_2(u) + w_2(v) \geq m - (w_2(s) - 1)w_2(u). \tag{5.2}$$

*Case* 1. $w_2(u) \leq \lceil \frac{m-1}{w_2(s)} \rceil - 1$.

Let $m - 1 = qw_2(s) + r$ with $0 \leq r < w_2(s)$. We have

$$\left\lceil \frac{m-1}{w_2(s)} \right\rceil = \begin{cases} q & \text{if } r = 0; \\ q+1 & \text{otherwise.} \end{cases}$$

When $r = 0$, inequality (5.2) gives

$$\begin{aligned} w_2(u) + w_2(v) &\geq m - (w_2(s) - 1)(q - 1) \\ &\geq m - w_2(s)q + q + w_2(s) - 1 \\ &\geq 1 + q + w_2(s) - 1 \\ &\geq q + 1 \\ &\geq \left\lceil \frac{m-1}{w_2(s)} \right\rceil + 1. \end{aligned}$$

When $r > 0$, inequality (5.2) gives

$$\begin{aligned} w_2(u) + w_2(v) &\geq m - (w_2(s) - 1)q \\ &\geq 1 + r + q \\ &\geq q + 2 \\ &\geq \left\lceil \frac{m-1}{w_2(s)} \right\rceil + 1. \end{aligned}$$

*Case* 2. $w_2(u) = \lceil \frac{m-1}{w_2(s)} \rceil$.
Since $\gcd(2^m - 1, s) = 1$, there is no integer $u < 2^m - 1$ satisfying $us \equiv 0 \bmod (2^m - 1)$. This implies that $w_2(v) \geq 1$, and therefore

$$w_2(u) + w_2(v) \geq \left\lceil \frac{m-1}{w_2(s)} \right\rceil + 1.$$

*Case* 3. $w_2(u) \geq \lceil \frac{m-1}{w_2(s)} \rceil + 1$.
In this case, inequality (5.1) obviously holds. □

**5.2. Cyclic codes $\mathcal{C}_{1,s}^{\perp}$ with a small weight divisibility.** We now focus on the values of $s$ for which the exact weight divisibility of $\mathcal{C}_{1,s}^{\perp}$ is small, i.e., for which $\mathcal{C}_{1,s}^{\perp}$ is exactly $2^{\ell}$-divisible with $\ell \in \{1, 2, 3\}$.

If $\gcd(s, 2^m - 1) = 1$, the only values of $s$ such that $\mathcal{C}_{1,s}^{\perp}$ is exactly 2-divisible satisfy $w_2(s) = m - 1$ [15, Thm. 4.7]; they all lie in the cyclotomic coset defined by $(2^{m-1} - 1)$. In this case $\mathcal{C}_{1,s}^{\perp}$ is the dual of the Melas code; its weights are all even integers $w$ such that $|w - \frac{2^m - 1}{2}| \leq 2^{\frac{m}{2}}$ [20]. The following proposition similarly exhibits all values of $s$ such that $\mathcal{C}_{1,s}^{\perp}$ is exactly 4-divisible.

PROPOSITION 5.3. *Let $m$ and $s$ be two positive integers such that* $\gcd(s, 2^m - 1) = 1$. *Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$.*
(i) *$\mathcal{C}_{1,s}^{\perp}$ is exactly 2-divisible if and only if $w_2(s) = m - 1$.*
(ii) *$\mathcal{C}_{1,s}^{\perp}$ is exactly 4-divisible if and only if either $w_2(s) = m - 2$ or $w_2(s^{-1}) = m - 2$, where $s^{-1}$ is the only integer in $\{0, \ldots, 2^m - 1\}$ such that $s^{-1}s \equiv 1 \bmod (2^m - 1)$.*

*Proof.*
(i) This was proved by Helleseth [15].
(ii) Suppose that $w_2(s) < m - 1$. In this case, $\mathcal{C}_{1,s}^{\perp}$ is at least 4-divisible. According to McEliece's theorem (Corollary 2.3) $\mathcal{C}_{1,s}^{\perp}$ is exactly 4-divisible if and only if there exists a pair of integers $(u, v)$ in $\{0, \ldots, 2^m - 1\}$ such that $us + v \equiv 0 \bmod (2^m - 1)$ and $w_2(u) + w_2(v) = 3$. Since $\gcd(s, 2^m - 1) = 1$, we have $w_2(v) \geq 1$. It follows that $(w_2(u), w_2(v)) \in \{(1, 2), (2, 1)\}$.

If $w_2(u) = 1$ and $w_2(v) = 2$, such a pair exists if and only if $w_2(s) = m - 2$; $s$ then lies in the same cyclotomic coset as $2^{m-1} - 2^i - 1$ for some $i \in [0, \ldots, 2^m - 1]$. If $w_2(u) = 2$ and $w_2(v) = 1$, such a pair exists if and only if there exists an element $s' \in Cl(s)$ and two distinct integers $i_1$ and $i_2$ in $[0, \ldots, m - 1]$ such that

$$\begin{aligned} &-(2^{i_1} + 2^{i_2})s' \equiv 1 \bmod (2^m - 1) \\ \iff &(2^m - 2^{i_1} - 2^{i_2} - 1)s' \equiv 1 \bmod (2^m - 1). \end{aligned}$$

It follows that $s'$ is the inverse modulo $(2^m - 1)$ of an element whose 2-weight equals $m - 2$. □

We now prove that if $\gcd(2^m - 1, s) = 1$ and if $\mathcal{C}_{1,s}^{\perp}$ is exactly 8-divisible, $s$ should satisfy the following necessary condition.

PROPOSITION 5.4. *Let $m$ and $s$ be two positive integers such that* $\gcd(2^m - 1, s) = 1$. *Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$. If $\mathcal{C}_{1,s}^{\perp}$ is exactly 8-divisible, then $s$ satisfies one of the following conditions:*
(i) $w_2(s^{-1}) = m - 3$, *where $s^{-1}$ is the only integer in $\{0, \ldots, 2^m - 1\}$ such that $s^{-1}s \equiv 1 \bmod (2^m - 1)$;*

(ii) $\lceil \frac{m-2}{2} \rceil \leq w_2(s) \leq m-3$.

*Proof.* Proposition 5.3 implies that $\mathcal{C}_{1,s}^{\perp}$ is 8-divisible if and only if $w_2(s) \leq m-3$ and $w_2(s^{-1}) \leq m-3$. Let us assume that this condition is satisfied. We now use the same technique as in the previous proof: $\mathcal{C}_{1,s}^{\perp}$ is exactly 8-divisible if and only if there exists a pair of integers $(u,v) \in \{0,\ldots,2^m-1\}$ such that $us+v \equiv 0 \bmod (2^m-1)$ and $w_2(u)+w_2(v) = 4$. Since $\gcd(s,2^m-1)=1$, we have that $(w_2(u),w_2(v)) \in \{(1,3),(2,2),(3,1)\}$.

- If $(w_2(u),w_2(v)) = (1,3)$, then $w_2(s) = m-3$.
- If $(w_2(u),w_2(v)) = (3,1)$, then $w_2(s^{-1}) = m-3$ (the proof here is the same as the proof of Proposition 5.3(ii)).
- If $(w_2(u),w_2(v)) = (2,2)$, we use inequality (5.2) proved in Corollary 5.2:

$$w_2(u)w_2(s)+w_2(v) \geq m.$$

It follows that

$$w_2(s) \geq \frac{m-2}{2}. \qquad \square$$

Note that Corollary 5.1 and Proposition 5.3 imply that $\mathcal{C}_{1,s}^{\perp}$ is exactly 8-divisible when $w_2(s) = m-3$ or $w_2(s^{-1}) = m-3$.

OPEN PROBLEM 5.5. *For any $\ell$, $1 \leq \ell \leq \lfloor \frac{m-1}{2} \rfloor$, does there exist an $s$ such that $w_2(s) = m-\ell$ and the cyclic code $\mathcal{C}_{1,s}^{\perp}$ of length $(2^m-1)$ is exactly $2^\ell$-divisible?*

## 6. Weight divisibility of the code $\mathcal{C}_{1,s}^{\perp}$ of length $(2^{2t+1}-1)$ with $s = 2^t+2^i-1$.

In his 1968 paper [14], Golomb mentioned a conjecture of Welch stating that for $m = 2t+1$, the crosscorrelation function between a binary m-sequence of length $(2^m-1)$ and its decimation by $s = 2^t+3$ takes on precisely the three values $-1, -1 \pm 2^{t+1}$. Niho [26] stated a similar conjecture for $s = 2^t + 2^{\frac{t}{2}} - 1$ when $t$ is even and $s = 2^t + 2^{\frac{3t+1}{2}} - 1$ when $t$ is odd. These conjectures equivalently assert that, for these values of $s$, the power function $x \mapsto x^s$ is AB over $\mathbf{F}_{2^m}$. Note that all of these exponents $s$ can be written as $2^t+2^i-1$ for some $i$. Since both Welch's and Niho's functions are APN [11, 10], Corollary 4.3 leads to a new formulation of Welch's and Niho's conjectures.

CONJECTURE 6.1. *Let $m = 2t+1$ be an odd integer. For all $u$ such that $1 \leq u \leq 2^m-1$, we have*

$$w_2((2^t+2^i-1)u \bmod (2^m-1)) \leq t + w_2(u) \tag{6.1}$$

*for the following values of $i$: $i=2$, $i = t/2$ for even $t$, and $i = (3t+1)/2$ for odd $t$.*

Previously, we proved that condition (6.1) is satisfied in the Welch case ($i = 2$) [5, 4]. More recently, Hollmann and Xiang used this formulation for proving Niho's conjecture [16]. Here we focus on all other values of $s$ which can be expressed as $s = 2^t+2^i-1$ for some $i$. We prove that for almost all of these values $\mathcal{C}_{1,s}^{\perp}$ actually contains a codeword whose weight is not divisible by $2^t$. This result is derived from both of the following lemmas which give an upper bound on the exact weight divisibility of $\mathcal{C}_{1,s}^{\perp}$.

LEMMA 6.2. *Let $m = 2t+1$ be an odd integer and $s = 2^t+2^i-1$ with $2 < i < t-1$. Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m-1)$ with defining set $\{1,s\}$. If $2^\ell$ denotes the exact divisibility of $\mathcal{C}_{1,s}^{\perp}$, we have*

- *if $t \equiv 0 \bmod i$ and $i \neq t/2$, then $\ell \leq t-1$;*
- *if $t \equiv 1 \bmod i$, then $\ell \leq t-i+2$;*

- *if* $t \equiv r \bmod i$ *with* $1 < r < i$, *then* $\ell \leq t - i + r$.

*Proof.* Let $t = iq + r$ with $r < i$ and $A(u) = (2^t + 2^i - 1)u \bmod 2^m - 1$. McEliece's theorem (Corollary 2.4) implies that $\mathcal{C}_{1,s}^{\perp}$ is at most $2^\ell$-divisible if there exists an integer $u \in \{0, \dots, 2^m - 1\}$ such that

$$w_2(A(u)) = w_2(u) + 2t - \ell.$$

Here we exhibit an integer $u$ satisfying this condition for the announced values of $\ell$.

- We first consider the case $r \neq 0$. Let $u = 2^t + 2^{r-1} \sum_{k=1}^{q} 2^{ik} + 1$. Then $w_2(u) = q + 2$ and we have

$$A(u) = 2^{2t} + 2^{t+r-1} \sum_{k=1}^{q} 2^{ik} + 2^t + 2^{t+i} - 2^t + 2^{r-1}\left(2^{(q+1)i} - 2^i\right) + 2^i - 1$$

$$= 2^{2t} + 2^{t+r-1} \sum_{k=1}^{q} 2^{ik} + 2^{t+i} + (2^{t+i-1} - 2^{i+r-1}) + (2^i - 1). \tag{6.2}$$

If $r > 1$, we have for all $k$ such that $1 \leq k \leq q$,

$$t + i < t + r - 1 + ik \leq 2t - 1.$$

We then deduce that all terms in (6.2) are distinct. It follows that

$$\begin{aligned} w_2(A(u)) &= 1 + q + 1 + (t - r) + i \\ &= w_2(u) + t - r + i. \end{aligned}$$

If $r = 1$, we obtain

$$A(u) = 2^{2t} + 2^t \sum_{k=2}^{q} 2^{ik} + 2^{t+i+1} + 2^{t+i-1} - 1.$$

In this case

$$\begin{aligned} w_2(A(u)) &= 1 + (q - 1) + 1 + (t + i - 1) \\ &= w_2(u) + t + i - 2. \end{aligned}$$

- Suppose now that $r = 0$ and $i \neq t/2$. Since $i < t$, we have $q > 2$. Let $u = 2^{t+i} + 2^{t+2} + 2^t + 2^{i+2} \sum_{k=0}^{q-2} 2^{ik} + 1$. Using that $i > 2$, we deduce that, for all $k \leq q - 2$,

$$\begin{aligned} i + 2 + ik &\leq i(q-1) + 2 \\ &\leq t - i + 2 < t. \end{aligned}$$

It follows that $w_2(u) = q + 3$. Let us now expand the corresponding $A(u)$:

$$\begin{aligned} A(u) \equiv\ & 2^{2t+i} + 2^{2t+2} + 2^{2t} + 2^{t+i+2} \sum_{k=0}^{q-2} 2^{ik} + 2^t + 2^{t+2i} - 2^{t+i} + 2^{t+i+2} \\ & -2^{t+2} + 2^{t+i} - 2^t + 2^{i+2}\left(2^{(q-1)i} - 1\right) + 2^i - 1 \\ =\ & 2^{i-1} + 2 + 2^{2t} + 2^{t+i+2} \sum_{k=1}^{q-2} 2^{ik} + 2^{t+2i} + 2^{t+i+3} + 2^{i+2}\left(2^{(q-1)i} - 1\right) \\ & +2^i - 1 \end{aligned}$$

$$= 2^{2t} + \sum_{k=0}^{q-3} 2^{t+2+(k+2)i} + 2^{t+2i} + 2^{t+i+3} - 2^{i+2} + 2^i + 2^{i-1} + 1. \tag{6.3}$$

If $i > 2$, all values of $k$ such that $0 \leq k \leq q-3$ satisfy

$$t + 2i < t + 2 + (k+2)i < 2t.$$

We then deduce that, if $q > 2$, all the terms in (6.3) are distinct except if $i = 3$. It follows that, for any $i > 3$,

$$\begin{aligned} w_2(A(u)) &= 1 + (q-2) + 1 + (t+1) + 3 \\ &= w_2(u) + t + 1. \end{aligned}$$

For $i = 3$, we have

$$A(u) = 2^{2t} + \sum_{k=0}^{q-3} 2^{t+3k+8} + 2^{t+7} - 2^5 + 2^3 + 2^2 + 1.$$

In this case

$$\begin{aligned} w_2(A(u)) &= 1 + (q-2) + (t+2) + 3 \\ &= t + q + 4 \\ &= w(u) + t + 1. \qquad \square \end{aligned}$$

LEMMA 6.3. *Let $m = 2t+1$ be an odd integer $s = 2^t + 2^i - 1$ with $t+1 < i < 2t$. Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$. If $2^\ell$ denotes the exact divisibility of $\mathcal{C}_{1,s}^{\perp}$, we have*

- *if $t+1 < i < \frac{3t+1}{2}$, then $\ell \leq m - i$;*
- *if $\frac{3t+1}{2} < i < 2t-1$, then $\ell \leq 2(m-i) - 1$;*
- *if $i = 2t-1$, then $\ell \leq 3$.*

*Proof.* Let $A(u) = (2^t + 2^i - 1)u \bmod (2^m - 1)$. Exactly as in the proof of the previous lemma, we exhibit an integer $u \in \{0, \ldots, 2^m - 1\}$ such that

$$w_2(A(u)) = w_2(u) + 2t - \ell$$

for the announced values of $\ell$. We write $i = t + j$, where $1 < j < t$.

- We first consider the case $t+1 < i < \frac{3t+1}{2}$. Let $u = 2^t + 2^{j-1} + 1$. Then $w_2(u) = 3$ and

$$\begin{aligned} A(u) &\equiv 2^{2t} + 2^{t+j-1} + 2^t + 2^{2t+j} + 2^{t+2j-1} + 2^{t+j} - 2^t - 2^{j-1} - 1 \\ &= 2^{2t} + 2^{t+2j-1} + 2^{t+j} + 2^{t+j-1} - 1. \end{aligned} \tag{6.4}$$

  Since $j < \frac{t+1}{2}$, we have that $2t > t + 2j - 1$. All the terms in (6.4) are therefore distinct. We deduce

$$\begin{aligned} w_2(A(u)) &= 3 + (t + j - 1) \\ &= w_2(u) + i - 1. \end{aligned}$$

- We now focus on the case $\frac{3t+1}{2} < i \leq 2t-1$. Let $u = 2^t + 2^j + 1$. Then $w_2(u) = 3$ and

$$\begin{aligned} A(u) &= 2^{2t} + 2^{j-1} - 2^t + 2^{t+j} + 2^{2j-t-1} - 2^j + 2^t + 2^{t+j} - 1 \\ &= 2^{2t} + 2^{t+j+1} - 2^{j-1} + 2^{2j-t-1} - 1. \end{aligned} \tag{6.5}$$

Since $\frac{t+1}{2} < j < t$, we have $0 < 2j - t - 1 < j - 1$. If $j \neq t - 1$, all the exponents in (6.5) are distinct. It follows that

$$\begin{aligned} w_2(A(u)) &= 1 + (t+2) + (2j - t - 1) \\ &= w_2(u) + 2(i - t) - 1. \end{aligned}$$

If $j = t - 1$, we have

$$A(u) = 2^{2t+1} - 2^{j-1} + 2^{2j-t-1} - 1.$$

In this case

$$\begin{aligned} w_2(A(u)) &= (2t+1) - (t - j) \\ &= w_2(u) + 2t - 3. \qquad \square \end{aligned}$$

For $i = 2t - 1$, we actually obtain the exact divisibility of $\mathcal{C}_{1,s}^{\perp}$.

PROPOSITION 6.4. *Let $m > 3$ be an odd integer. The dual of the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, 2^{m-2} + 2^{\frac{m-1}{2}} - 1\}$ is exactly $2^3$-divisible.*

*Proof.* Let $s = 2^{m-2} + 2^{\frac{m-1}{2}} - 1$. The previous lemma implies that $\mathcal{C}_{1,s}^{\perp}$ is at most $2^3$-divisible. Since

$$(2^{\frac{m+1}{2}} + 1)s \equiv 2^{\frac{m-3}{2}}(2^{\frac{m-1}{2}} - 1) \bmod (2^m - 1),$$

we obtain that $\gcd(2^m - 1, s)$ divides both $(2^m - 1)$ and $(2^{\frac{m-1}{2}} - 1)$ which are coprime. It follows that $\gcd(2^m - 1, s) = 1$.

From Proposition 5.3 we know that the divisibility of $\mathcal{C}_{1,s}^{\perp}$ is less than $2^3$ if either $w_2(s) \geq m - 2$ or $w_2(s^{-1}) \geq m - 2$. None of these conditions is satisfied here: $w_2(s) = \frac{m+1}{2}$ and $s^{-1}$ lies in the same cyclotomic coset as $2^{m-2} - 2^{\frac{m-1}{2}} - 1$ since

$$\begin{aligned} (2^{m-2} - 2^{\frac{m-1}{2}} - 1)(2^{m-2} + 2^{\frac{m-1}{2}} - 1) &\equiv (2^{m-2} - 1)^2 - 2^{m-1} \bmod (2^m - 1) \\ &\equiv 2^{m-4} \bmod (2^m - 1). \end{aligned}$$

It follows that $w_2(s^{-1}) = m - 3$ and therefore that $\mathcal{C}_{1,s}^{\perp}$ is exactly 8-divisible. $\square$

From Lemmas 6.2 and 6.3 we deduce that $\mathcal{C}_{1,s}^{\perp}$ is not $2^t$-divisible for most values of $s = 2^t + 2^i - 1$.

THEOREM 6.5. *Let $m = 2t + 1$ be an odd integer and let $s = 2^t + 2^i - 1$ with $i \in \{1, \dots, 2t\}$. Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$. The only values of $i$ such that $\mathcal{C}_{1,s}^{\perp}$ is $2^t$-divisible are $1, 2, \frac{t}{2}, t, t+1, \frac{3t+1}{2}, 2t$, and maybe $t - 1$.*

*Proof.* If $i \notin \{1, 2, \frac{t}{2}, t-1, t, t+1, \frac{3t+1}{2}, 2t\}$, $\mathcal{C}_{1,s}^{\perp}$ is not $2^t$-divisible since the upper bounds given in both previous lemmas are strictly less than $t$. Moreover, $\mathcal{C}_{1,s}^{\perp}$ is $2^t$-divisible for $i \in \{1, 2, \frac{t}{2}, t, t+1, \frac{3t+1}{2}, 2t\}$:

- $i = 1$ corresponds to a quadratic value of $s$.
- $i = 2$ corresponds to the Welch's function.
- $i = t$ corresponds to the inverse of a quadratic exponent since $(2^{t+1} - 1)(2^t + 1) \equiv 2^t \bmod 2^m - 1$.
- $i = t + 1$ corresponds to a Kasami's function since $2^t(2^{t+1} + 2^t - 1) \equiv 2^{2t} - 2^t + 1 \bmod 2^m - 1$.
- $i = 2t$ gives an $s$ which is in the same 2-cyclotomic coset as $2^{t+1} - 1$.
- $i = \frac{t}{2}$ or $i = \frac{3t+1}{2}$ corresponds to Niho's function. $\square$

The only unresolved case is then $i = t - 1$. In accordance with our simulation results we formulate the following conjecture.

CONJECTURE 6.6. *Let $m = 2t + 1$ be an odd integer, $m > 3$. The dual of the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, 2^t + 2^{t-1} - 1\}$ is exactly $2^t$-divisible.*

Note that an equivalent assertion is that the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, 2^{t+2} + 3\}$ is exactly $2^t$-divisible, since $2^{t+2} + 3$ lies in the same cyclotomic coset as the inverse of $2^t + 2^{t-1} - 1$. We checked this conjecture by computer for $m \leq 39$. Note that it is obviously satisfied for $m = 5$ and $m = 7$ since the function $x \mapsto x^s$ for $s = 2^t + 2^{t-1} - 1$ corresponds, resp., to a quadratic function and to the Welch function. On the contrary it is known that the corresponding function is not APN when 3 divides $m$ since $\mathcal{C}_{1,s}$ has minimum distance 3 in this case.

PROPOSITION 6.7 (see [8]). *Let $m = 2t + 1$ be an odd integer, $m > 3$. The binary cyclic code of length $(2^m - 1)$ with defining set $\{1, 2^t + 2^{t-1} - 1\}$ has minimum distance 3 if and only if $m \equiv 0 \bmod 3$. In this case, the number of codewords of weight 3 is*

$$B_3 = 2^m - 1.$$

*Proof.* This comes from [8, Thm. 5]. This cyclic code has minimum distance 3 if and only if

$$\gcd(m, t - 1) = 1.$$

Since $m = 2t - 1$, it follows that $\gcd(m, t - 1) = 3$ when $m \equiv 0 \bmod 3$ and that $\gcd(m, t - 1) = 1$ otherwise. $\square$

Table 6.1 gives the weight distributions of $\mathcal{C}_{1,s}^{\perp}$, where $s = 2^t + 2^{t-1} - 1$ for $m \in \{9, 11, 13\}$ and the corresponding number $B_3$ (resp., $B_4$) of codewords of weight 3 (resp., 4) in $\mathcal{C}_{1,s}$.

TABLE 6.1
*Weight distribution of some codes $\mathcal{C}_{1,s}^{\perp}$ with $s = 2^t + 2^{t-1} - 1$.*

| $m$ | $s$ | $B_3$ | $B_4$ | $w$ | $A_w$ |
|---|---|---|---|---|---|
| 9 | 23 | 511 | 9709 | 224 | 4599 |
| | | | | 240 | 55188 |
| | | | | 256 | 146657 |
| | | | | 272 | 55188 |
| | | | | 288 | 511 |
| 11 | 47 | 0 | 180136 | 960 | 45034 |
| | | | | 992 | 900680 |
| | | | | 1024 | 2368379 |
| | | | | 1056 | 835176 |
| | | | | 1088 | 45034 |
| 13 | 95 | 0 | 2981524 | 3968 | 745381 |
| | | | | 4032 | 14055756 |
| | | | | 4096 | 38030813 |
| | | | | 4160 | 13531532 |
| | | | | 4224 | 745381 |

**7. Weight divisibility of $\mathcal{C}_{1,s}^{\perp}$ when $m$ is not a prime.** We now focus on the binary cyclic codes $\mathcal{C}_{1,s}$ of length $(2^m - 1)$ when $m$ is not a prime. We show that in this case the weight divisibility of $\mathcal{C}_{1,s}^{\perp}$ is closely related to the exact weight divisibility of the cyclic code $\mathcal{C}_{1,s_0}^{\perp}$ of length $(2^g - 1)$ where $g$ is a divisor of $m$ and $s_0 = s \bmod (2^g - 1)$.

**7.1. A general result.** We first derive a lower and an upper bound on the exact weight divisibility of $\mathcal{C}_{1,s}^{\perp}$ from the exact weight divisibility of the code $\mathcal{C}_{1,s_0}^{\perp}$ of length $(2^g-1)$. These bounds allow us to give a necessary condition on the decimations which provide preferred pairs of m-sequences of length $(2^m-1)$ when $m$ is not a prime. This general condition generalizes the Sarwate–Pursley conjecture [31, 25] on the nonexistence of preferred pairs of m-sequences of length $(2^m-1)$ when 4 divides $m$.

THEOREM 7.1. *Let $g$ be a divisor of $m$. Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m-1)$ with defining set $\{1,s\}$ and $\mathcal{C}_0$ the binary cyclic code of length $(2^g-1)$ with defining set $\{1,s_0\}$, where $s_0 = s \bmod (2^g-1)$. Assume that $\mathcal{C}_0^{\perp}$ is exactly $2^{\ell}$-divisible. Then we have the following:*

(i) *$\mathcal{C}_{1,s}^{\perp}$ is $2^{\ell}$-divisible. In particular, if there exists $i$ such that $s \equiv 2^i \bmod (2^g-1)$, then $\mathcal{C}_{1,s}^{\perp}$ is $2^{g-1}$-divisible.*

(ii) *$\mathcal{C}_{1,s}^{\perp}$ is not $2^{\frac{m}{g}(\ell+1)}$-divisible.*

*Proof.* Let $s = s_0 + a(2^g-1)$. Here we use McEliece's theorem as expressed in Corollary 2.3.

(i) Let $u$ and $v$ be two integers in $\{0,\dots,2^m-1\}$ such that $us+v = k(2^m-1)$. Then we have

$$us+v = au(2^g-1) + (us_0+v) = k(2^m-1).$$

This implies that

$$us_0 + v \equiv 0 \bmod (2^g-1).$$

Let $u_0 = u \bmod (2^g-1)$ and $v_0 = v \bmod (2^g-1)$. For any integer $x$, we have $w_2(x) \geq w_2(x \bmod (2^g-1))$: by writing $x = x_2 2^g + x_1$ with $0 \leq x_1 < 2^g$, we obtain that $x \bmod (2^g-1) = x_1 + x_2$. It follows that

$$\begin{aligned} w_2(x \bmod (2^g-1)) &= w_2(x_1+x_2) \\ &\leq w_2(x_1) + w_2(x_2) = w_2(x). \end{aligned}$$

This implies that

$$w_2(u) + w_2(v) \geq w_2(u_0) + w_2(v_0) \geq \ell + 1$$

since $\mathcal{C}_0^{\perp}$ is $2^{\ell}$-divisible.

(ii) If $\mathcal{C}_0^{\perp}$ is exactly $2^{\ell}$-divisible, there exists a pair of integers $(u_0, v_0)$ with $u_0 \leq 2^g-1$ and $v_0 \leq 2^g-1$ such that

$$u_0 s_0 + v_0 \equiv 0 \bmod 2^g-1 \text{ and } w_2(u_0) + w_2(v_0) = \ell+1.$$

Let us now consider both integers $u$ and $v$ defined by

$$u = u_0 \frac{2^m-1}{2^g-1} \text{ and } v = v_0 \frac{2^m-1}{2^g-1}.$$

For $s = s_0 + a(2^g-1)$, the pair $(u,v)$ satisfies

$$\begin{aligned} us+v &= ua(2^g-1) + us_0 + v \\ &= u_0 a(2^m-1) + \frac{2^m-1}{2^g-1}(u_0 s_0 + v_0) \\ &\equiv 0 \bmod (2^m-1). \end{aligned}$$

Since $\frac{2^m-1}{2^g-1} = \sum_{i=0}^{m/g-1} 2^{ig}$ and both $u_0$ and $v_0$ are less than $2^g-1$, we have

$$w_2(u) + w_2(v) = \frac{m}{g}\left(w_2(u_0) + w_2(v_0)\right) = \frac{m}{g}(\ell+1).$$

We then deduce that $\mathcal{C}_{1,s}^{\perp}$ is not $2^{\frac{m}{g}(\ell+1)}$-divisible. ◻

COROLLARY 7.2. *Let $g$ be a divisor of $m$. Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m-1)$ with defining set $\{1, s\}$ and $\mathcal{C}_0$ the binary cyclic code of length $(2^g-1)$ with defining set $\{1, s_0\}$, where $s_0 = s \bmod (2^g-1)$. If $\mathcal{C}_{1,s}^{\perp}$ is $2^{\lfloor \frac{m}{2} \rfloor}$-divisible, then $\mathcal{C}_0^{\perp}$ is $2^{\lfloor \frac{g}{2} \rfloor}$-divisible.*

*Proof.* We denote by $2^\ell$ the exact divisibility of $\mathcal{C}_0^{\perp}$. Following Theorem 7.1(ii), we have that $\mathcal{C}_{1,s}^{\perp}$ is not $2^{\frac{m}{g}(\ell+1)}$-divisible. If $\mathcal{C}_{1,s}^{\perp}$ is $2^{\lfloor \frac{m}{2} \rfloor}$-divisible, it therefore follows that

$$\left\lfloor \frac{m}{2} \right\rfloor \leq \frac{m}{g}(\ell+1) - 1.$$

For odd $m$, this gives

$$\ell + 1 \geq \frac{g(m+1)}{2m} > \frac{g-1}{2}$$

since $(m+1)g > m(g-1)$.

For even $m$, we similarly get

$$\ell + 1 \geq \frac{g(m+2)}{2m} > \frac{g}{2} \geq \left\lfloor \frac{g}{2} \right\rfloor.$$

This implies in both cases that $\ell \geq \lfloor \frac{g}{2} \rfloor$; $\mathcal{C}_0^{\perp}$ is then $2^{\lfloor \frac{g}{2} \rfloor}$-divisible. ◻

We now deduce a necessary condition on the values of the decimations which provide preferred pairs of m-sequences (or equivalently on the exponents which correspond to AB power permutations when $m$ is odd).

PROPOSITION 7.3. *Let $m$ and $s$ be two positive integers such that $\gcd(2^m-1, s) = 1$ and $s$ is not a power of 2. The pair formed by an m-sequence of length $(2^m-1)$ and its decimation by $s$ is not preferred if there exists a divisor $g$ of $m$ with $g > 2$ satisfying one of the following conditions:*

1. *$\exists i,\ 0 \leq i < g,\ \ s \equiv 2^i \bmod (2^g-1)$;*
2. *$s_0 = s \bmod (2^g-1) \neq 2^i$ and the dual of the cyclic code of length $(2^g-1)$ with defining set $\{1, s_0\}$ is not $2^{\lfloor \frac{g}{2} \rfloor}$-divisible.*

*Proof.* Theorems 4.1 and 4.4 provide a necessary condition for obtaining a preferred crosscorrelation: $\mathcal{C}_{1,s}^{\perp}$ has to be $2^{\lfloor \frac{m}{2} \rfloor}$-divisible and the number $B_3$ of codewords of weight 3 in $\mathcal{C}_{1,s}$ should be

$$\begin{aligned} B_3 &= \frac{(2^m-1)}{3} \text{ if } m \text{ is even} \\ &= 0 \text{ if } m \text{ is odd.} \end{aligned}$$

When $s \equiv 2^i \bmod (2^g-1)$, it is known [8] that the cyclic code $\mathcal{C}_{1,s}$ has minimum distance 3 and that the number $B_3$ of codewords of weight 3 satisfies

$$B_3 \geq \frac{(2^m-1)(2^{g-1}-1)}{3}.$$

This implies that the crosscorrelation is not preferred if $g > 2$.

If $s_0 = s \bmod (2^g - 1)$ is such that the dual of the cyclic code of length $(2^g - 1)$ with defining set $\{1, s_0\}$ is not $2^{\lfloor \frac{g}{2} \rfloor}$-divisible, $\mathcal{C}_{1,s}^{\perp}$ is not $2^{\lfloor \frac{m}{2} \rfloor}$-divisible according to Corollary 7.2. ∎

This proposition is a generalization of the following result conjectured by Sarwate and Pursley [31] and recently proved in [25].

COROLLARY 7.4 (see [31, 25]). *There is no pair of preferred m-sequences of length* $(2^m - 1)$ *when* 4 *divides* $m$.

*Proof.* We consider the pair formed by an m-sequence of length $(2^m - 1)$ and its decimation by $s$. Let $s_0 = s \bmod 15$. Since 4 divides $m$, we should have $s_0 \in \{1, 2, 4, 8\} \cup \{7, 11, 13, 14\}$; otherwise $s$ and $2^m - 1$ are not coprime. The previous proposition then applies with $g = 4$ for any such $s$: condition 1 is satisfied when $s_0 \in \{1, 2, 4, 8\}$ and condition 2 holds when $s_0 \in \{7, 11, 13, 14\}$ since the dual of the cyclic code of length 15 with defining set $\{1, 7\}$ is not 4-divisible. ∎

**7.2. Exact weight divisibility of the code $\mathcal{C}_{1,s}^{\perp}$ of length $(2^{2t} - 1)$ with $s \equiv 2^i \bmod (2^t - 1)$.** When $m = 2t$ and $s \equiv 2^i \bmod (2^t - 1)$, all the weights of $\mathcal{C}_{1,s}^{\perp}$ are divisible by $2^{t-1}$. This particular case of Theorem 7.1(i) was already treated in [26] and in [12, Lemma 1]. We now prove that this bound actually corresponds to the exact divisibility of $\mathcal{C}_{1,s}^{\perp}$.

THEOREM 7.5. *Let* $m = 2t$ *be an even integer and let* $s$ *be an integer such that*

$$s \equiv 2^i \bmod (2^t - 1) \text{ with } 0 \le i < t$$

*and* $s$ *is not a power of* 2. *Let* $\mathcal{C}_{1,s}$ *be the binary cyclic code of length* $(2^m - 1)$ *with defining set* $\{1, s\}$. *Then* $\mathcal{C}_{1,s}^{\perp}$ *is exactly* $2^{t-1}$*-divisible.*

*Proof.* Since Theorem 7.1(i) implies that the weights of $\mathcal{C}_{1,s}^{\perp}$ are divisible by $2^{t-1}$, it is sufficient to find a pair $(u, v)$ such that $us + v \equiv 0 \bmod (2^m - 1)$ and $w_2(u) + w_2(v) = t$. By cyclotomic equivalence, we have only to consider the values of $s$ such that

$$s = a(2^t - 1) + 2^{t-1},$$

where $a \le 2^t$ and $a \notin \{0, 2^{t-1}\}$ (otherwise $s$ is a power of 2).

- If $a < 2^{t-1}$, we write $a = 2^j a'$ with $a'$ odd. Then $s$ lies in the same 2-cyclotomic coset as $s' = a'(2^t - 1) + 2^{t-j-1}$. Let us choose $u = 2^t + 1 - 2^{t-j-1}$. Then we have

$$\begin{aligned} us' &\equiv a'(2^{2t} - 1) + 2^{t-j-1}\left[-a'(2^t - 1) + 2^t + 1 - 2^{t-j-1}\right] \bmod (2^{2t} - 1) \\ &\equiv 2^{t-j-1}\left[2^t(1 - a') + (a' + 1 - 2^{t-j-1})\right] \bmod (2^{2t} - 1). \end{aligned}$$

Then $us' + v \equiv 0 \bmod 2^m - 1$ implies that

$$v \equiv 2^{t-j-1}\left[2^t(a' - 1) + (2^{t-j-1} - 1 - a')\right] \bmod (2^m - 1).$$

Therefore, we obtain

$$\begin{aligned} w_2(v) &\le w_2\left(2^t(a' - 1) + (2^{t-j-1} - 1 - a')\right) \\ &\le w_2(a' - 1) + w_2(2^{t-j-1} - 1 - a') \\ &\le w_2(a' - 1) + t - j - 1 - w_2(a') \\ &\le t - j - 2 \end{aligned}$$

since $a' \le 2^{t-j-1} - 1$ and $a'$ is odd. Using that $w_2(u) = j + 2$, we finally get

$$w_2(u) + w_2(v) \le t.$$

- If $2^{t-1} < a \leq 2^t$, we choose $u = 1$. By writing

$$s = (a-1)2^t + (2^t - 1) - (a - 1 - 2^{t-1}),$$

we obtain that

$$w_2(s) = w_2(a-1) + t - w_2(a - 1 - 2^{t-1}).$$

Since $2^{t-1} \leq a-1 \leq 2^t-1$, we clearly have that $w_2(a-1-2^{t-1}) = w_2(a-1)-1$. It follows that

$$w_2(s) = w_2(a-1) + t - (w_2(a-1) - 1) = t + 1.$$

For $v = 2^m - 1 - s$, we then have $w_2(u) + w_2(v) = t$. □

Most values of $s$ for which the weight distribution of $\mathcal{C}_{1,s}^{\perp}$ is completely determined satisfy $s \equiv 2^i \bmod (2^t - 1)$. Theorem 7.5 most notably covers the following well-known cases:

(i) $s = 2^t + 1$ since $s = (2^t - 1) + 2$. In this case the weights of $\mathcal{C}_{1,s}^{\perp}$ take the following values: $2^{m-1} - 2^{t-1}, 2^{m-1}, 2^{m-1} + 2^{t-1}$ [19].

(ii) $s = 2^t + 3$ since $s = (2^t - 1) + 4$. The weights of $\mathcal{C}_{1,s}^{\perp}$ take the following values: $2^{m-1} - 2^t - 2^{t-1}, 2^{m-1} - 2^t, 2^{m-1} + 2^{t-1}, 2^{m-1}, 2^{m-1} - 2^{t-1}$ [15, Thm. 4.8].

(iii) $s = \sum_{i=0}^{t} 2^{ik}$, where $0 < k < t$ and $\gcd(m, k) = 1$. Since $k$ is odd, we have

$$\begin{aligned} s = \sum_{i=0}^{t} 2^{ik} &\equiv \frac{2^{(t+1)k} - 1}{2^k - 1} \bmod (2^m - 1) \\ &\equiv \frac{2^{t+k} - 1}{2^k - 1} \bmod (2^m - 1). \end{aligned}$$

It follows that

$$s \equiv 2^k d(2^t - 1) + 1 \bmod (2^m - 1),$$

where $d$ satisfies $d(2^k - 1) \equiv 1 \bmod (2^m - 1)$. The weights of $\mathcal{C}_{1,s}^{\perp}$ take the following values: $2^{m-1} - 2^t, 2^{m-1} - 2^{t-1}, 2^{m-1}, 2^{m-1} + 2^{t-1}$ [12, Prop. 1].

(iv) $s = (2^t + 1)(2^{\frac{t}{2}} - 1) + 2$ when $t$ is even. In this case, $s = (2^{\frac{t}{2}} - 1)(2^t - 1) + 2^{\frac{t}{2}+1}$. The weights of $\mathcal{C}_{1,s}^{\perp}$ take the following values: $2^{m-1} - 2^{\frac{3t}{2}-1}, 2^{m-1} - 2^{t-1}, 2^{m-1}, 2^{m-1} + 2^{t-1}$ [26].

(v) $s = \frac{(2^m-1)}{3} + 2^i$ with $i \in \{0, 1\}$ when $t$ is odd. In this case, 3 divides $(2^t + 1)$. It follows that

$$s = \frac{(2^t + 1)}{3}(2^t - 1) + 2^i.$$

The weights of $\mathcal{C}_{1,s}^{\perp}$ take the following values [15, Thm. 4.11]:

- $2^{m-1} - \frac{2^{t-1}(2^t+1)}{3}, 2^{m-1} - \frac{2^t(2^{t-1}-1)}{3}, 2^{m-1} - 2^t, 2^{m-1} - 2^{t-1}, 2^{m-1}, 2^{m-1} + 2^{t-1}$, when $\frac{2^{m-i}(2^m-1)}{3} \equiv 0 \bmod 3$.
- $2^{m-1} - \frac{2^{t+1}(2^{t-2}+1)}{3}, 2^{m-1} - \frac{2^t(2^{t-1}-1)}{3}, 2^{m-1} - 2^t, 2^{m-1} - 2^{t-1}, 2^{m-1}, 2^{m-1} + 2^{t-1}$, when $\frac{2^{m-i}(2^m-1)}{3} \equiv 1 \bmod 3$.

Tables 7.1, 7.2, and 7.3 give the complete weight distribution of all codes $\mathcal{C}_{1,s}^{\perp}$ of length $(2^m - 1)$ for $m \in \{6, 8, 10\}$ when $s$ satisfies $s \equiv 2^i \bmod (2^{\frac{m}{2}} - 1)$.

TABLE 7.1
*Weight distribution of* $\mathcal{C}_{1,s}^{\perp}$ *with* $s \equiv 2^i \bmod (2^{\frac{m}{2}} - 1)$ *for* $m = 6$.

| | $A_w$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $s$ | 20 | 24 | 28 | 32 | 36 | $B_3$ | $B_4$ | ref. |
| 9 | | | 252 | 63 | 196 | 63 | 945 | (i) |
| 11, 23 | 126 | 252 | 756 | 1827 | 1134 | 84 | 252 | (ii),(v) |
| 15 | | 588 | 504 | 1827 | 1176 | 63 | 63 | (iii) |

TABLE 7.2
*Weight distribution of* $\mathcal{C}_{1,s}^{\perp}$ *with* $s \equiv 2^i \bmod (2^{\frac{m}{2}} - 1)$ *for* $m = 8$.

| | $A_w$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $s$ | 96 | 104 | 112 | 120 | 128 | 136 | $B_3$ | $B_4$ | ref. |
| 17 | | | | 2040 | 255 | 1800 | 595 | 37485 | (i) |
| 19, 47 | | 2040 | 2040 | 16320 | 22695 | 22440 | 595 | 3825 | (ii) |
| 31, 91 | | | 10200 | 4080 | 30855 | 20400 | 595 | 1785 | (iii) |
| 53 | 1020 | | | 24480 | 15555 | 24480 | 595 | 6885 | (iv) |
| 23, 61 | 510 | | 5100 | 14280 | 23205 | 22440 | 595 | 4335 | |

TABLE 7.3
*Weight distribution of* $\mathcal{C}_{1,s}^{\perp}$ *with* $s \equiv 2^i \bmod (2^{\frac{m}{2}} - 1)$ *for* $m = 10$.

| | $A_w$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $s$ | 320 | 336 | 352 | 448 | 464 | 480 | 496 | 512 | 528 |
| 33 | | | | | | | 16368 | 1023 | 15376 |
| 35 | | | | | 40920 | 16368 | 245520 | 377487 | 368280 |
| 63, 219 | | | | | | 169136 | 32736 | 508431 | 338272 |
| 171 | | 2046 | 1023 | | | 41261 | 225060 | 471603 | 307582 |
| 343 | 1023 | | 2046 | | | 40920 | 225060 | 472626 | 306900 |
| 39, 159 | | | | 5115 | 10230 | 77066 | 196416 | 390786 | 368962 |
| 47, 109, 125, 221 | | | | 5115 | 10230 | 87978 | 163680 | 423522 | 358050 |
| 101, 157 | | | | | 30690 | 57288 | 184140 | 418407 | 358050 |
| 187 | | | 93 | | 20460 | 82863 | 163680 | 422499 | 358980 |

| $s$ | $B_3$ | $B_4$ | Ref. |
|---|---|---|---|
| 33 | 5115 | 1304325 | (i) |
| 35 | 5456 | 81840 | (ii) |
| 63, 219 | 5115 | 35805 | (iii) |
| 171 | 20460 | 1616340 | (iv) |
| 343 | 20801 | 1677720 | (iv) |
| 39, 159 | 5115 | 71610 | |
| 47, 109, 125, 221 | 5456 | 76725 | |
| 101, 157 | 5456 | 71610 | |
| 187 | 5456 | 92070 | |

OPEN PROBLEM 7.6. *Let $m = 2t$ be an even integer and let $s$ be an integer such that*

$$s \equiv 2^i \bmod (2^t - 1) \text{ with } 0 \le i < t.$$

*Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$. Let $A = (A_0, \ldots, A_{2^m-1})$ denote the weight enumerator of $\mathcal{C}_{1,s}^{\perp}$. Find the largest integer $w_0$*

*such that*

$$A_w = A_{2^m - w} = 0 \textit{ for all } w \textit{ such that } 0 < w < w_0.$$

**7.3. $2^{\frac{m}{2}}$-divisible cyclic codes of length $(2^m - 1)$ when $m$ is a multiple of 4.** By combining Theorem 7.5 and Corollary 7.2 we are now able to exhibit a necessary condition on $s$ under which the cyclic code $\mathcal{C}_{1,s}^{\perp}$ of length $(2^m - 1)$ is $2^{\frac{m}{2}}$-divisible. This condition depends on the value of the highest power of 2 which divides $m$.

PROPOSITION 7.7. *Let $m = 2^a m'$ with $m'$ odd and $a \geq 2$. Let $s$ be a positive integer such that* $\gcd(2^m - 1, s) = 1$ *and let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$. If $\mathcal{C}_{1,s}^{\perp}$ is $2^{\frac{m}{2}}$-divisible, then*

$$s \equiv 2^i \bmod (2^{2^a} - 1) \textit{ for some } i$$

*and the number $B_3$ of codewords of weight 3 in $\mathcal{C}_{1,s}$ satisfies*

$$B_3 \geq \frac{(2^m - 1)(2^{2^a - 1} - 1)}{3}.$$

*Proof.* Assume that $\mathcal{C}_{1,s}^{\perp}$ is $2^{\frac{m}{2}}$-divisible. For $j \in \{2, \ldots, a\}$, we set $s_j = s \bmod (2^{2^j} - 1)$ and we denote by $\mathcal{C}_j$ the binary cyclic code of length $(2^{2^j} - 1)$ with defining set $\{1, s_j\}$. From Corollary 7.2 we have that $\mathcal{C}_a^{\perp}$ is $2^{2^{a-1}}$-divisible.

We now prove by induction on $j$ that if $\mathcal{C}_j^{\perp}$ is $2^{2^{j-1}}$-divisible, then $s_j$ is a power of 2.

- If $j = 2$, we have that $s_2 \notin \{3, 6, 9, 12\} \cup \{5, 10\}$ since $\gcd(s, 2^m - 1) = 1$ implies that $\gcd(s_{j-1}, 2^4 - 1) = 1$. Moreover, if $s_2 \in \{7, 11, 13, 14\}$, $\mathcal{C}_2$ is not 4-divisible. It follows that $s_2 \in \{1, 2, 4, 8\}$.
- Induction step: Let us suppose that $\mathcal{C}_j^{\perp}$ is $2^{2^{j-1}}$-divisible. According to Corollary 7.2 we have that $\mathcal{C}_{j-1}^{\perp}$ is $2^{2^{j-2}}$-divisible. By an induction hypothesis $s_{j-1}$ is a power of 2, i.e.,

  $$s_j \equiv 2^i \bmod (2^{2^{j-1}} - 1) \text{ for some } i.$$

  Now Theorem 7.5 implies that, in this case, $\mathcal{C}_j$ is exactly $2^{2^{j-1}-1}$-divisible if $s_j$ is not a power of 2.

We now deduce that if $\mathcal{C}_a^{\perp}$ is $2^{2^{a-1}}$-divisible, then

$$s \equiv 2^i \bmod (2^{2^a} - 1) \text{ for some } i. \qquad \square$$

When $m$ is a power of 2, Proposition 7.7 implies the following corollary.

COROLLARY 7.8. *Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$ ($s$ is not a power of 2). If $m$ is a power of 2 and if* $\gcd(2^m - 1, s) = 1$*, then $\mathcal{C}_{1,s}^{\perp}$ is not $2^{\frac{m}{2}}$-divisible.*

By using Corollary 4.5, we see that this result actually corresponds to the following weak version of the Helleseth conjecture [15] which was proved by Calderbank, McGuire, and Poonen [3].

COROLLARY 7.9 (see [15, 3]). *If $m$ is a power of 2, then there are no pairs of binary m-sequences of length $(2^m - 1)$ with crosscorrelation values $-1$, $-1 + 2D$, $-1 - 2D$.*

When 4 divides $m$ (and $m$ is not a power of 2) Proposition 7.7 allows us to find all values of $s$ for which $\mathcal{C}_{1,s}^{\perp}$ is $2^{\frac{m}{2}}$-divisible very fast, even for large values of $m$. We search for all such values for $m \in \{12, 10, 24\}$ and our numerical results lead us to formulate the following conjecture.

Conjecture 7.10. *Let $m = 2^a m'$ with $a \geq 2$, $m'$ odd, and $m' > 1$. Let $s$ be a positive integer such that* $\gcd(s, 2^m - 1) = 1$ *($s$ is not a power of 2) and let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$. The cyclic code $\mathcal{C}_{1,s}^{\perp}$ is $2^{\frac{m}{2}}$-divisible if and only if either $s$ or $s^{-1}$ takes one of the following values (up to cyclotomic equivalence):*

$$2^{i2^a} + 1 \quad \textit{or} \quad 2^{i2^{a+1}} - 2^{i2^a} + 1$$

*with $1 \leq i \leq \lfloor \frac{m'}{2} \rfloor$.*

It is known that for these values of $s$, $\mathcal{C}_{1,s}^{\perp}$ is exactly $2^{\ell}$-divisible with $\ell = \frac{m}{2} - 1 + 2^{a-1} \gcd(m', i)$ [13, 19]. The previous conjecture equivalently asserts that these values are the only decimations which provide a symmetric 3-valued crosscorrelation, i.e., a crosscorrelation function which takes its values in $\{-1, -1 + 2D, -1 - 2D\}$.

**7.4. Dobbertin's function.** We now obtain an upper bound on the divisibility of the weights of $\mathcal{C}_{1,s}^{\perp}$ in the following particular case.

Proposition 7.11. *Let $\mathcal{C}_{1,s}$ be the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$. Let $g$ be a divisor of $m$ such that $s$ satisfies:*

$$s \equiv -s_0 \bmod \frac{2^m - 1}{2^g - 1} \textit{ with } 0 < s_0 < \frac{2^m - 1}{2^g - 1}.$$

*Then $\mathcal{C}_{1,s}^{\perp}$ is not $2^{g(w_2(s_0)+1)}$-divisible.*

*Proof.* Here we use McEliece's theorem as formulated in Corollary 2.4. Let $u = 2^g - 1$. Then we have

$$A(u) = us \bmod 2^m - 1 = (2^m - 1) - (2^g - 1)s_0.$$

We obtain that

$$w_2(A(u)) = m - w_2((2^g - 1)s_0).$$

Since $w_2((2^g - 1)s_0) \leq g w_2(s_0)$, this implies that

$$\begin{aligned} w_2(A(u)) &\geq m - g w_2(s_0) \\ &\geq w_2(u) + m - g(w_2(s_0) + 1). \end{aligned}$$

We then deduce that $\mathcal{C}_{1,s}^{\perp}$ is not $2^{w_2(u)(w_2(s_0)+1)}$-divisible. ◻

Corollary 7.12. *Let $m$ be an odd integer. If there exists a divisor $g$ of $m$ such that $s$ satisfies*

$$s \equiv -s_0 \bmod \frac{2^m - 1}{2^g - 1} \textit{ with } 0 < s_0 < \frac{2^m - 1}{2^g - 1}$$

*and*

$$w_2(s_0) \leq \frac{1}{2}\left(\frac{m}{g} - 3\right),$$

*then the power function $x \mapsto x^s$ is not AB on $\mathbf{F}_{2^m}$.*

The third author conjectured that for $m = 5g$ the power function $x \mapsto x^s$ with $s = 2^{4g} + 2^{3g} + 2^{2g} + 2^g - 1$ is APN on $\mathbf{F}_{2^m}$ [10]. The previous corollary implies the following.

PROPOSITION 7.13. *Let $m$ be an odd integer such that $m = 5g$. The power function $x \mapsto x^s$ with $s = 2^{4g} + 2^{3g} + 2^{2g} + 2^g - 1$ is not AB on $\mathbf{F}_{2^m}$.*

*Proof.* Since $s = \frac{2^{5g}-1}{2^g-1} - 2$, we apply the previous corollary with $s_0 = 2$ and $m/g = 5$ using that

$$w_2(s_0) = 1 = \frac{1}{2}\left(\frac{m}{g} - 3\right). \qquad \square$$

Note that Proposition 7.11 implies that, for $m = 5g$ and $s = 2^{4g}+2^{3g}+2^{2g}+2^g-1$, $\mathcal{C}_{1,s}^{\perp}$ is not $2^{2g}$-divisible. This means, for example, that for $m = 15$ and $s \in Cl(3657)$, $\mathcal{C}_{1,s}^{\perp}$ is at most $2^5$-divisible; this bound actually corresponds to the exact divisibility of the code.

**7.5. Numerical results.** We now give some examples for $m = 10$ and 14. As above, $\mathcal{C}_{1,s}$ denotes the binary cyclic code of length $(2^m - 1)$ with defining set $\{1, s\}$ and $2^\ell$ denotes the exact weight divisibility of $\mathcal{C}_{1,s}^{\perp}$. Table 7.4 recalls some values of $s$ for which the value of $\ell$ is known.

Tables 7.5 and 7.6 compare the true exact weight divisibility of $\mathcal{C}_{1,s}^{\perp}$ (given in the second column of each table) with the theoretical bounds derived from the previous results.

***m* = 10.**

- We always have $\ell \le 10 - w_2(s)$ (Corollary 5.1) and if $\gcd(s, 1023) = 1$, we also get $\ell \le 10 - w_2(s^{-1})$, where $s^{-1}$ is defined by $ss^{-1} \equiv 1 \bmod 1023$. Maoreover, Proposition 5.3 implies that if $w_2(s) > 8$ and $w_2(s^{-1}) > 8$, we have $\ell \ge 3$.
- If $\gcd(s, 1023) = d > 1$, we have $\ell \le w_2(\frac{2^m-1}{d}) - 1$. This is derived from $us + v \equiv 0 \bmod 1023$ with $u = \frac{2^m-1}{d}$ and $v = 0$. Note that if $s$ and 1023 are not coprime, $\ell \le 4$ since the 2-weight of any divisor of 1023 is at most 4.
- If $s \bmod 31 \in \{1, 2, 4, 8, 16\}$, then $\ell = 4$ (Theorem 7.5).
- If $s \bmod 31 \in \{15, 23, 27, 29, 30\}$, then $\ell \le 3$. This comes from Theorem 7.1 using the fact that the code $\mathcal{C}_{1,15}^{\perp}$ of length 31 is not 4-divisible.
- Proposition 5.4 implies that $\ell \ge 4$ when $\gcd(2^m - 1, s) = 1$, $w_2(s) \le 3$, and $w_2(s^{-1}) \le 6$.

TABLE 7.4
*Exact weight divisibility of the duals of some cyclic codes of length $(2^m - 1)$ with defining set $\{1, s\}$.*

| $s$ | $\ell$ | Ref. |
|---|---|---|
| $2^i + 1$, $1 \le i \le \lfloor \frac{m}{2} \rfloor$ | $\frac{m+\gcd(m,i)}{2} - 1$ if $\frac{m}{\gcd(m,i)}$ is odd<br>$\frac{m}{2} - 1$ if $\frac{m}{\gcd(m,i)}$ is even | [13, 19] |
| $2^{2i} - 2^i + 1$, $2 \le i \le \lfloor \frac{m}{2} \rfloor$ | $\frac{m+\gcd(m,i)}{2} - 1$ if $\frac{m}{\gcd(m,i)}$ is odd<br>$\frac{m}{2} - 1$ if $\frac{m}{\gcd(m,i)}$ is even | [19] |
| $\frac{(2^m-1)}{3} + 2^i$, $m$ even | $\frac{m}{2} - 1$ | [15] |
| $2^{\frac{m}{2}} + 2^{\frac{m}{4}} + 1$, $m \equiv 0 \bmod 4$ | $\frac{m}{2} - 1$ | [12] |
| $2^{\frac{m}{2}} + 2^{\frac{m+2}{2}} + 1$, $m \equiv 2 \bmod 4$ | $\frac{m}{2} - 1$ | [9] |
| $2^{\frac{m+2}{2}} + 3$, $m \equiv 2 \bmod 4$ | $\frac{m}{2} - 1$ | [9] |

TABLE 7.5
*Exact weight divisibility of $\mathcal{C}_{1,s}^{\perp}$ for $m = 10$.*

| $s$ (representatives of cosets) | $\ell$ | | Argument |
|---|---|---|---|
| 511 | 1 | $\ell = 1$ | $w_2(s) = 9$ |
| 31 93 155 341 | 2 | $\ell = 1$ | 31 divides $\gcd(1023, s)$ |
| 173 179 255 383 447 479 495 | 2 | $\ell = 2$ | $w_2(s) = 8$ or $w_2(s^{-1}) = 8$ |
| 15 23 27 29 61 77 85 89 91 123 151 213 215 247 | 3 | $\ell = 3$ | $s \bmod 31 \in \{15, 23, 27, 29, 30\}$ |
| 7 53 59 73 127 167 191 223 235 239 251 253 351 367 375 379 439 | 3 | $\ell = 3$ | $w_2(s) = 7$ or $w_2(s^{-1}) = 7$ |
| 19 175 | 4 | $\ell = 4$ | $w_2(s) = 3$ and $w_2(s^{-1}) = 6$ |
| 33 35 39 47 63 95 101 109 125 157 159 171 187 219 221 343 | 4 | $\ell = 4$ | $s \bmod 31 \in \{1, 2, 4, 8, 16\}$ |
| 3 9 57 | 4 | $\ell = 4$ | see Table 7.4 |
| 71 111 245 | 3 | $3 \le \ell \le 4$ | $w_2(s) = 6$ or $w_2(s^{-1}) = 6$ |
| 43 115 119 183 189 207 231 237 347 363 | 4 | | |
| 45 51 55 75 99 147 165 | 3 | $3 \le \ell \le 4$ | $\gcd(1023, s) \in \{3, 11, 33\}$ |
| 11 21 69 87 105 117 121 | 4 | | |
| 37 83 | 4 | $4 \le \ell \le 5$ | $w_2(s) = 3$ and $w_2(s^{-1}) \le 6$ |
| 103 149 | 3 | $3 \le \ell \le 5$ | |
| 5 205 13 79 17 181 25 41 49 107 | 5 | $\ell = 5$ | see Table 7.4 |

- There is no 8-divisible code $\mathcal{C}_{1,s_0}^{\perp}$ of length 31 with $\gcd(s_0, 31) = 1$. It follows from Theorem 7.1(ii) that, for any $s$ such that $\gcd(s, 2^{10} - 1) = 1$, the code $\mathcal{C}_{1,s}^{\perp}$ of length $(2^{10} - 1)$ is at most $2^5$-divisible.

***m* = 14.** Here we examine only the values of $s$ such that $\gcd(2^{14} - 1, s) = 1$. Note that if $\gcd(2^{14} - 1, s) > 1$, $\mathcal{C}_{1,s}^{\perp}$ is not $2^7$-divisible since the 2-weight of any factor of $(2^{14} - 1)$ is at most 7. Table 7.6 is derived from the following theoretical results:

- Proposition 5.3 implies that $\ell \ge 3$ for any $s$ such that $w_2(s) \le 12$ and $w_2(s^{-1}) \le 12$.
- From Corollary 5.1, we have $\ell \le 14 - w_2(s)$.
- For any $s$ such that $w_2(s) = 3$ or $w_2(s^{-1}) = 3$, we have $\ell \ge 5$ according to Corollary 5.2.
- Proposition 5.4 implies that $\ell \ge 4$ when $w_2(s) \le 5$ and $w_2(s^{-1}) \le 10$.
- According to Theorem 7.1(ii) we always have that $\ell \le 7$ since the cyclic code $\mathcal{C}_{1,s_0}^{\perp}$ of length $(2^7 - 1)$ is at most $2^3$-divisible when $s_0$ is not a power of 2.
- The cyclic code $\mathcal{C}_{1,s_0}^{\perp}$ of length $(2^7 - 1)$ is exactly 2-divisible when $s_0 \in Cl(63)$ ($Cl(i)$ here denotes the 2-cyclotomic coset modulo 127). It follows that $\ell \le 3$ for any $s$ such that $s \bmod 127 \in Cl(63)$ (Theorem 7.1(ii)).
- The codes $\mathcal{C}_{1,s_0}^{\perp}$ of length $(2^7 - 1)$ are exactly 4-divisible for $s_0 \in \mathcal{E} = Cl(7) \cup Cl(19) \cup Cl(21) \cup Cl(31) \cup Cl(47) \cup Cl(55)$. Theorem 7.1(ii) then implies that $\mathcal{C}_{1,s}^{\perp}$ is at most $2^5$-divisible for any $s$ such that $s \bmod 127 \in \mathcal{E}$.
- If $s \equiv 2^i \bmod 127$ for some $i$, $\mathcal{C}_{1,s}^{\perp}$ is exactly $2^6$-divisible according to Theorem 7.5.

TABLE 7.6
*Exact weight divisibility* $\mathcal{C}_{1,s}^{\perp}$ *for* $m = 14$ *with* $\gcd(2^{14}-1, s) = 1$.

| $s$ (representatives of cosets) | $\ell$ | | Argument |
|---|---|---|---|
| 8191 | 1 | $\ell = 1$ | $w_2(s) = 13$ |
| 2741 2861 2867 6143 7679 8063 | 2 | $\ell = 2$ | $w_2(s) = 12$ or $w_2(s^{-1}) = 12$ |
| 95 119 125 253 317 349 365 373 377 379 571 619 631 761 857 881 887 1015 1111 1127 1135 1139 1141 1381 1523 1619 1643 1897 1901 1903 2381 2405 2411 2539 2651 2663 3047 3413 3421 3935 4063 | 3 | $\ell = 3$ | $s \bmod 127 \in Cl(63)$ |
| 283 511 655 703 1003 1117 1177 1259 1273 1375 1435 1679 1919 2047 2479 2527 2797 2971 3071 3583 3703 3757 3839 3967 4031 4079 4087 4091 4093 5851 5887 6079 6127 6139 7039 7103 7135 7151 7159 7663 | 3 | $\ell = 3$ | $w_2(s) = 11$ or $w_2(s^{-1}) = 11$ |
| 47 59 83 149 197 203 329 341 355 625 1109 1301 3055 3067 3551 3775 3835 4015 4075 5503 5627 5855 5983 6007 | 4 | $\ell = 4$ | $w_2(s) \leq 5$ and $w_2(s^{-1}) = 10$ |
| 175 461 691 811 967 1019 1213 1253 1277 1385 1391 1535 1615 1685 1847 1957 2015 2035 2039 2045 2347 2453 2507 2765 2771 2807 2815 3007 3455 3517 3575 3581 3823 3959 3965 4027 4055 4061 5567 5599 5615 5623 5879 5999 6011 6071 6107 7031 | 4 | $3 \leq \ell \leq 4$ | $w_2(s) = 10$ or $w_2(s^{-1}) = 10$ |
| 7 19 25 31 37 41 67 73 97 245 443 529 869 983 1123 1205 2341 3547 | 5 | $\ell = 5$ | $s \bmod 127 \in \mathcal{E}$ and $w_2(s) = 3$ |
| 115 239 463 533 617 809 1631 2005 | 4 | $4 \leq \ell \leq 5$ | $s \bmod 127 \in \mathcal{E}$ and $w_2(s) \leq 5$ and $w_2(s^{-1}) < 10$ |
| 55 61 79 91 103 107 109 121 155 169 209 211 227 275 295 313 361 395 409 419 457 475 481 539 563 569 581 587 595 601 611 649 677 709 745 781 787 793 799 803 841 871 971 1013 1171 1193 1225 1307 1343 1387 1531 1597 1621 1645 1775 1895 1967 1999 2383 2407 2779 2903 3323 3389 3419 3829 | 5 | | |
| 251 455 719 1735 2389 3925 | 3 | $3 \leq \ell \leq 5$ | $s \bmod 127 \in \mathcal{E}$ |
| 431 437 493 503 605 629 685 697 853 1237 1243 1255 1325 1331 1357 1363 1373 1481 1693 1837 1865 2455 2717 2813 2983 2995 3805 4013 | 4 | | |
| 221 347 371 491 499 505 575 599 623 757 821 823 829 859 877 883 917 931 965 973 1001 1007 1103 1133 1181 1199 1261 1267 1337 1367 1379 1439 1447 1453 1459 1471 1507 1519 1627 1639 1727 1751 1769 1871 1885 1943 1961 1979 1981 2023 2027 2029 2359 2395 2495 2525 2647 2743 2749 2767 2791 2909 2911 2935 3005 3031 3287 3293 3485 3503 3511 3541 3563 3767 3901 3931 4021 5815 | 5 | | |
| 289 | 5 | $\ell = 5$ | $w_2(s) = 3$ and $w_2(s^{-1}) = 9$ |
| 689 2687 | 4 | $4 \leq \ell \leq 5$ | $w_2(s) \leq 5$ and $w_2(s^{-1}) = 9$ |
| 77 85 157 269 305 307 323 325 391 401 451 613 647 913 1021 1093 1279 1787 1915 1951 2975 3391 3451 3535 3709 3773 3949 5471 5495 5591 | 5 | | |

TABLE 7.6
*(Continued)*

| $s$ (representatives of cosets) | $\ell$ | | Argument |
|---|---|---|---|
| 959 3059 | 3 | $3 \leq \ell \leq 5$ | $w_2(s) = 9$ or $w_2(s^{-1}) = 9$ |
| 715 895 991 1183 1223 1469 1759 1783 2419 2431 2555 2557 2677 3023 3415 3515 3791 3803 5551 5563 5819 | 4 | | |
| 235 343 427 487 743 751 767 797 847 875 935 995 1009 1231 1319 1495 1663 1747 1789 1853 1855 1939 1975 2011 2041 2357 2471 2519 2543 2551 2783 2879 2927 2939 2941 2999 3035 3037 3061 3295 3319 3325 3383 3439 3487 3559 3565 3707 3743 3797 3799 3821 3947 5611 | 5 | | |
| 131 143 191 383 389 397 413 445 509 637 667 763 893 905 953 1145 1147 1151 1175 1207 1271 1399 1405 1429 1525 1655 1715 1907 1909 1913 2429 2477 2669 2671 2675 2683 2731 2923 3431 3437 3445 | 6 | $\ell = 6$ | $s \bmod 127 \in Cl(1)$ |
| 11 1501 | 6 | $5 \leq \ell \leq 6$ | $w_2(s) = 3$ and $w_2(s^{-1}) = 8$ |
| 955 1187 1321 2719 | 4 | $4 \leq \ell \leq 6$ | $w_2(s) \leq 5$ and $w_2(s^{-1}) = 8$ |
| 89 101 163 181 185 331 541 553 1499 1511 2747 2933 2987 3317 3407 3509 | 5 | | |
| 353 535 583 1723 1883 2423 | 6 | | |
| 479 1351 | 3 | $3 \leq \ell \leq 6$ | $w_2(s) = 8$ or $w_2(s^{-1}) = 8$ |
| 407 839 943 1303 1369 1465 1661 1711 1771 2003 2549 2803 2863 2875 | 4 | | |
| 287 439 607 695 827 845 863 997 1327 1403 1463 1483 1487 1517 1529 1595 1687 1705 1757 1781 1949 1963 1973 1997 2009 2363 2399 2653 2711 2735 2989 3029 3307 3499 3701 3755 5819 | 5 | | |
| 497 683 1823 1831 1835 1879 2483 2491 5467 5483 | 6 | | |
| 5 13 17 65 113 145 193 205 241 319 979 1339 1613 2773 2893 3277 | 7 | $\ell = 7$ | see Table 7.4 |
| 35 49 137 161 265 335 433 469 919 1355 2515 | 6 | $5 \leq \ell \leq 7$ | $w_2(s) = 3$ or $w_2(s^{-1}) = 3$ |
| 167 187 199 229 247 263 299 425 707 805 911 1097 1099 1195 1445 1589 1609 1625 1717 1843 1867 1877 | 4 | $4 \leq \ell \leq 7$ | $w_2(s) \leq 5$ and $w_2(s^1) < 10$ |
| 29 71 139 151 173 217 223 233 271 281 403 421 547 551 557 565 589 593 653 659 661 665 721 739 749 785 923 947 977 1129 1165 1189 1309 1423 1607 1691 1829 2461 | 5 | | |
| 23 53 179 277 293 337 713 727 937 1163 1315 1451 1657 1739 2869 2899 | 6 | | |
| 1241 1703 2459 2645 | 3 | $3 \leq \ell \leq 7$ | |
| 415 679 701 733 755 815 907 941 949 985 1115 1211 1427 1433 1637 1721 1753 2485 | 4 | | |
| 311 359 367 467 485 671 791 851 925 1229 1235 1421 1493 1513 1709 1741 1765 1945 1993 2351 2387 2533 | 5 | | |
| 133 725 2509 | 6 | | |

## REFERENCES

[1] T. BETH AND C. DING, *On almost perfect nonlinear permutations*, in Advances in Cryptology–EUROCRYPT '93, Lecture Notes in Comput. Sci. 765, Springer-Verlag, New York, 1993, pp. 65–76.

[2] E. BIHAM AND A. SHAMIR, *Differential cryptanalysis of DES-like cryptosystems*, J. Cryptology, 4 (1991), pp. 3–72.

[3] A. Calderbank, G. McGuire, and B. Poonen, *On a conjecture of Helleseth regarding pairs of binary m-sequences*, IEEE Trans. Inform. Theory, 42 (1996), pp. 988–990.

[4] A. Canteaut, P. Charpin, and H. Dobbertin, *Binary m-sequences with three-valued crosscorrelation: A proof of Welch's conjecture*, IEEE Trans. Inform. Theory, to appear.

[5] A. Canteaut, P. Charpin, and H. Dobbertin, *Couples de suites binaires de longueur maximale ayant une corrélation croisée à trois valeurs: Conjecture de Welch*, C. R. Acad. Sci. Paris Sér. I Math., 328 (1999), pp. 173–178.

[6] C. Carlet, P. Charpin, and V. Zinoviev, *Codes, bent functions and permutations suitable for DES-like cryptosystems*, Des. Codes Cryptogr., 15 (1998), pp. 125–156.

[7] F. Chabaud and S. Vaudenay, *Links between differential and linear cryptanalysis*, in Advances in Cryptology–EUROCRYPT '94, Lecture Notes in Comput. Sci. 950, Springer-Verlag, New York, 1995, pp. 356–365.

[8] P. Charpin, A. Tietäväinen, and V. Zinoviev, *On binary cyclic codes with minimum distance* $d = 3$, Problems Inform. Transmission, 33 (1997), pp. 287–296.

[9] T. Cusick and H. Dobbertin, *Some new 3-valued crosscorrelation functions of binary m-sequences*, IEEE Trans. Inform. Theory, 42 (1996), pp. 1238–1240.

[10] H. Dobbertin, *Almost perfect nonlinear power functions on* $GF(2^n)$*: The Niho case*, Inform. and Comput., 151 (1999), pp. 57–72.

[11] H. Dobbertin, *Almost perfect nonlinear power functions on* $GF(2^n)$*: The Welch case*, IEEE Trans. Inform. Theory, 45 (1999), pp. 1271–1275.

[12] H. Dobbertin, *One-to-one highly nonlinear functions on finite field with characteristic* 2, Appl. Algebra Engrg. Comm. Comput., 9 (1998), pp. 139–152.

[13] R. Gold, *Maximal recursive sequences with* 3*-valued recursive crosscorrelation functions*, IEEE Trans. Inform. Theory, 14 (1968), pp. 154–156.

[14] S. Golomb, *Theory of transformation groups of polynomials over* $GF(2)$ *with applications to linear shift register sequences*, Inform. Sci., 1 (1968), pp. 87–109.

[15] T. Helleseth, *Some results about the cross-correlation function between two maximal linear sequences*, Discrete Math., 16 (1976), pp. 209–232.

[16] H. Hollmann and Q. Xiang, *A Proof of the Welch and Niho Conjectures on Crosscorrelations of Binary m-sequences*, preprint, 1998.

[17] H. Janwa and R. Wilson, *Hyperplane sections of Fermat varieties in* $P^3$ *in char.* 2 *and some applications to cyclic codes*, in Applied Algebra, Algebraic Algorithms and Error-Correcting Codes–Proceedings AAECC-10, Lecture Notes in Comput. Sci. 673, Springer-Verlag, Berlin, 1993, pp. 180–194.

[18] T. Kasami, *Weight distributions of Bose-Chaudhuri-Hocquenghem codes*, in Proceedings of the Conference on Combinatorial Mathematics and Its Applications, The Univ. of North Carolina Press, Chapel Hill, NC, 1969, pp. 335–357.

[19] T. Kasami, *The weight enumerators for several classes of subcodes of the second order binary Reed-Muller codes*, Inform. and Control, 18 (1971), pp. 369–394.

[20] G. Lachaud and J. Wolfmann, *The weights of the orthogonal of the extended quadratic binary Goppa codes*, IEEE Trans. Inform. Theory, 36 (1990), pp. 686–692.

[21] X. Lai, J. Massey, and S. Murphy, *Markov ciphers and differential cryptanalysis*, in Advances in Cryptology–EUROCRYPT '91, Lecture Notes in Comput. Sci. 547, Springer-Verlag, New York, 1991, pp. 17–38.

[22] M. Matsui, *Linear cryptanalysis method for DES cipher*, in Advances in Cryptology–EUROCRYPT '93, Lecture Notes in Comput. Sci. 765, Springer-Verlag, New York, 1994, pp. 386–397.

[23] M. Matsui, *The first experimental cryptanalysis of the data encryption standard*, in Advances in Cryptology–CRYPTO '94, Lecture Notes in Comput. Sci. 839, Springer-Verlag, New York, 1995, pp. 1–11.

[24] R. McEliece, *Weight congruence for p-ary cyclic codes*, Discrete Math., 3 (1972), pp. 177–192.

[25] G. McGuire and A. Calderbank, *Proof of a conjecture of Sarwate and Pursley regarding pairs of binary m-sequences*, IEEE Trans. Inform. Theory, 41 (1995), pp. 1153–1155.

[26] Y. Niho, *Multi-valued Cross-Correlation Functions Between Two Maximal Linear Recursive Sequences*, Ph.D. thesis, Univ. Southern California, Los Angeles, CA, 1972; Tech. report 409, Dept. Electrical Engineering, Univ. Southern California, Los Angeles, CA, 1972.

[27] K. Nyberg, *Differentially uniform mappings for cryptography*, in Advances in Cryptology–EUROCRYPT '93, Lecture Notes in Comput. Sci. 765, Springer-Verlag, New York, 1993, pp. 55–64.

[28] K. Nyberg, *Linear approximation of block ciphers*, in Advances in Cryptology–EUROCRYPT '94, Lecture Notes in Comput. Sci. 950, Springer-Verlag, New York, 1994, pp. 439–444.

[29] K. Nyberg and L. Knudsen, *Provable security against differential cryptanalysis*, in Advances in Cryptology–CRYPTO '92, Lecture Notes in Comput. Sci. 740, Springer-Verlag, New York, 1993, pp. 566–574.

[30] V. Pless, *Power moment identities on weight distributions in error-correcting codes*, Inform. and Control, 3 (1963), pp. 147–152.

[31] D. Sarwate and M. Pursley, *Crosscorrelation properties of pseudorandom and related sequences*, Proc. IEEE, 68 (1980), pp. 593–619.

[32] V. Sidelnikov, *On mutual correlation of sequences*, Soviet Math. Dokl., 12 (1971), pp. 197–201.

SIAM J. Discrete Math.
Vol. 13, No. 1, pp. 139–144

# ON ISOPERIMETRIC CONNECTIVITY IN VERTEX-TRANSITIVE GRAPHS[*]

Y. O. HAMIDOUNE[†], A. S. LLADÓ[‡], O. SERRA[‡], AND R. TINDELL[§]

**Abstract.** We shall define the $k$-isoperimetric connectivity $\lambda_k$ of a regular graph $\Gamma$ as the minimum number of arcs originating in a set with cardinality not exceeding half the order of the graph and containing at least $k$ vertices. Clearly $\lambda_k \leq dk - e_k$, where $d$ is the degree of $\Gamma$ and $e_k$ is the maximal number of edges induced on a set of $k$ vertices.

We shall show that Cayley graphs with a prime order and arc-transitive graphs have $\lambda_k = dk - e_k$, provided that $d \geq 3k - 3$. We describe all vertex-transitive graphs where $\lambda_2 \leq 2d - 3$.

**Key words.** isoperimetric inequalities, vertex-transitive graphs

**AMS subject classifications.** 05C20, 05C70

**PII.** S089548019630635X

## 1. Introduction.

For simplicity all graphs we consider will be assumed to be finite without loops or multiple arcs. However, the results in this paper extend easily to graphs with multiple arcs or loops and that are possibly infinite.

It is known that graphs having a transitive group of automorphisms have good connectivity. In particular Mader proved in [11] that a connected vertex-transitive undirected graph with degree $d$ remains connected after the deletion of $d - 1$ edges. This result was improved by Lovász and Plummer in [10, Lemma 5.5.26] by showing that, apart from an exceptional family, the graph is still connected after the deletion of $d$ edges nonincident to some vertex. A related result was proved independently by Tindell in [13]. The problem for directed graphs is studied in [9].

In this paper we are studying an isoperimetric generalization of connectivity. A special case of our results shows that, apart from an exceptional family, a connected vertex-transitive directed graph of outdegree $d$ is still strongly connected after the deletion of $2d - 3$ edges nonincident to some vertex.

Our $k$th isoperimetric connectivity will be defined later as the minimum number of arcs originating from a set with cardinality at least $k$ and at most half the order of the graph.

Notice that this notion of isoperimetric connectivity is related to the extraconnectivity introduced by Fàbrega and Fiol [3, 4]. Notice also that the corresponding problems for vertex-connectivity are studied in [6] and [8]. See also [2] for a related approach.

Our main tool will be the study of the $k$-atoms, which, in vertex-transitive graphs, will be shown to be blocks of imprimitivity of the automorphism group. As a corollary we shall show that for Cayley graphs with a prime order and for arc-transitive graphs,

[*]Received by the editors July 8, 1996; accepted for publication (in revised form) July 13, 1999; published electronically January 13, 2000. This work was partially supported by the Spanish Research Council CYCIT under project TIC97-0963.

http://www.siam.org/journals/sidma/13-1/30635.html

[†]E. Combinatoire, Université Pierre et Marie Curie, 4, Place Jussieu, 75005 Paris, France (yha@ccr.jussieu.fr).

[‡]Dep. Matemàtica Aplicada i Telemática, Universitat Politecnica de Catalunya, Jordi Girona, 1, 08034 Barcelona, Spain (allado@mat.upc.es, oriol@mat.upc.es).

[§]Department of Electrical Engineering and Computer Science, Stevens Institute of Technology, Hoboken, NJ 07030 (tindell@gauss.eecs.stevens-tech.edu).

the $k$th isoperimetric connectivity achieves its optimal value $dk - e_k$, where $d$ is the degree of the graph and $e_k$ is the maximal number of edges induced on a set of $k$ vertices, provided that $d \geq 3k-3$. We give also a characterization for vertex-transitive graphs where the isoperimetric connectivity is not optimal.

## 2. Isoperimetric connectivity.

The reader may find definitions not given here in [1].

By a *graph* we shall mean an irreflexive relation $\Gamma = (V, E)$, where $V$ is a finite set and $E \subset V \times V \setminus \{(x, x) : x \in V\}$.

In particular, $\Gamma(A)$ is the traditional image of $A$ defined in set theory. The elements of $V$ are called the *vertices* of $\Gamma$ and the elements of $E$ are called the *arcs* of $\Gamma$. A graph $\Gamma$ will be called symmetric (resp., antisymmetric) if the corresponding relation is so. In other words, graph $\Gamma = (V, E)$ is *symmetric* if for a pair $u, v$ of vertices of $\Gamma$, $(u, v) \in E$ implies $(v, u) \in E$. If $(u, v) \in E$ implies $(v, u) \notin E$ for every pair $u, v$ of vertices of $\Gamma$, we say that $\Gamma$ is *antisymmetric*.

Our results apply as well to undirected graphs since they may be identified with symmetric graphs in the usual way.

Let $\Gamma = (V, E)$ be a graph and let $F \subset V$. Denote by $\overline{F}$ the set $V \setminus F$. The *arc neighborhood* of $F$ is the set

$$\omega_\Gamma(F) = \{\, (u, v) \in E \;\big|\; u \in F \text{ and } v \in \overline{F} \,\}.$$

We will omit the subscript $\Gamma$ unless more than one graph is being considered. Arc neighborhoods of proper, nonempty subsets of $V$ are often called *edge cuts*. The *indegree* of vertex $v$ is $d^-(v) = |\omega(V \setminus \{v\})|$ and the *outdegree* of $v$ is $d(v) = |\omega(v)|$. A graph $\Gamma$ is said to be *regular of degree* $d$ if $d(v) = d^-(v) = d$ for every vertex $v$ of $\Gamma$. The degree of a regular graph $\Gamma$ will be denoted by $d(\Gamma)$.

A graph $\Gamma$ such that $d(u) = d^-(u)$ for every vertex $u$ of $\Gamma$ is said to be *Eulerian*. It is well known that in an Eulerian graph $\Gamma$, for every $F \subset V$,

$$|\omega(F)| = |\omega(\overline{F})|. \tag{1}$$

Recall that a finite Eulerian graph is strongly connected if and only if it is connected.

A graph $\Gamma$ is said to be *vertex-transitive* if for every pair $u, v$ of vertices of $\Gamma$ there is an automorphism $\varphi$ of $\Gamma$ with $\varphi(u) = v$.

Given a group $G$ and a subset $S \subset G \setminus \{1\}$, the Cayley graph of $G$ with respect to $S$, denoted by $\mathrm{Cay}(G, S)$, has the elements of the group as vertices and the ordered pairs $(x, xs)$, where $x \in G$ and $s \in S$, as edges. Cayley graphs are regular of degree $|S|$ and they are vertex-transitive. It is well known that $\mathrm{Cay}(G, S)$ is connected if and only if $S$ generates the group.

Let $\Gamma = (V, E)$ be a graph with $|V| \geq 2k$. Let us define the *kth-edge connectivity number* of $\Gamma$ as

$$\lambda_k(\Gamma) = \min\{\, |\omega X| \;\big|\; |X| \geq k \text{ and } |\overline{X}| \geq k \,\}.$$

When $k = 1$, $\lambda_1(\Gamma)$ is the edge-connectivity of the graph. This notion of $\lambda_k$ is related to the concept of extraconnectivity studied by Fàbrega and Fiol [3, 4] and coincides with it for symmetric graphs.

A subset $F \subset V$ is said to be a *k-fragment* if $|F| \geq k$, $|\overline{F}| \geq k$, and $|\omega(F)| = \lambda_k(\Gamma)$. A $k$-fragment of minimum cardinality will be called a *k-atom*. The cardinality of $k$-atoms of $\Gamma$ will be denoted by $\mu_k(\Gamma)$. The definitions of 1-fragment and 1-atom coincide with the notions of edge-fragment and edge-atoms; see [9, 12] and the references therein. We clearly have $\lambda_1(\Gamma) \le \lambda_2(\Gamma) \le \cdots \le \lambda_k(\Gamma)$.

Using Menger's theorem it can easily be seen that $\lambda_k(\Gamma)$ is the maximum $j$ such that two arbitrary disjoint sets of $k$ vertices of $\Gamma$ are connected by $j$ edge disjoint directed paths.

## 3. The $k$-atoms structure.

In this section we shall study the intersection properties of $k$-atoms.

Lemma 1. *Let $\Gamma$ be a Eulerian graph, let $A$ be a $k$-atom of $\Gamma$, and let $F$ be a $k$-fragment of $\Gamma$. Then either $|A\cap F| \le k-1$ or $A \subset F$. In particular, if $\mu_k(\Gamma) \ge 2k-1$, and if $B$ is a $k$-atom of $\Gamma$ distinct from $A$, then $A \cap B = \emptyset$.*

*Proof.* Suppose $|A \cap F)| \ge k$. By (1) $\overline{F}$ is a $k$-fragment of $\Gamma$ as well. Therefore, $|\overline{F}| \ge |A|$. Thus

$$|\overline{F}| = |A \setminus F| + |\overline{A \cup F}| \ge |A| = |A \setminus F| + |A \cap F|,$$

and it follows that $|\overline{A \cup F}| \ge |A \cap F| \ge k$. By the well-known submodularity relation,

$$|\omega(A \cap F)| + |\omega(\overline{A \cup F})| \le |\omega(A)| + |\omega(F)| = 2\lambda_k(\Gamma). \tag{2}$$

Since both $A \cap F$ and its complement have at least $k$ points and the same holds for $A \cup F$, we must have $|\omega(A \cap F)| \ge \lambda_k(F)$ and $|\omega(A \cup F)| \ge \lambda_k(F)$. Combined with inequality (2), we see that $A \cap F$ is a $k$-fragment. Since $A$ is a $k$-atom, it then follows that $A = A \cap F$.

We have $|A \cap \overline{B}| \ge k$, since $|A \cap B| \le k-1$. Since $\overline{B}$ is a $k$-fragment, the first part of the lemma implies $A \subset \overline{B}$. □

The first part of the above lemma can be generalized to arbitrary graphs. Since our main interest is the vertex-transitive case, we give only the Eulerian case.

We shall complete the above result with the following lemma to exclude the possibility that $k+1 \le \mu_k(\Gamma) \le 2k-2$.

Notice that $\mu_k(\Gamma) = k$ if and only if $\lambda_k(\Gamma) = dk - e_k(\Gamma)$. We shall use this obvious observation without reference throughout the paper.

The subgraph induced by a subset $F \subset V$ will be denoted by $\Gamma[F]$.

We shall denote by $e_k(\Gamma)$ the maximum number of edges in a subgraph induced by a subset of $k$ vertices.

We shall put $\epsilon(\Gamma) = 1/2$ if $\Gamma$ is antisymmetric and $\epsilon(\Gamma) = 1$, otherwise. For both $\mu$ and $\epsilon$, the reference to $\Gamma$ will be implicit.

Lemma 2. *Let $\Gamma$ be a connected regular graph with degree $d$ such that $\mu_k > k$. Then $\mu_k \ge d/\epsilon - k + 2$.*

*Proof.* Set $a = \mu_k$. Let $F$ be a subset of cardinality $k$ of a $k$-atom $A$. The number of arcs in $\Gamma[A]$ is $ad - \lambda_k$. This number is at most $a(a-1) - k(k-1) + e_k$, since these arcs induce in $F$ at most $e_k$ arcs. We have

$$dk - e_k > |\omega(A)| \ge ad - (a(a-1) - k(k-1) + e_k) = ad - (a-k)(a+k-1) - e_k.$$

Therefore, $a > d - k + 1$.

Assume now that $\Gamma$ is antisymmetric. A similar calculation now gives

$$dk-e_k > |\omega(A)| \ge ad-(1/2)(a(a-1)-k(k-1))-e_k = ad-(1/2)(a-k)(a+k-1)-e_k.$$

Hence $a > 2d - k + 1$. □

**4. Vertex-transitive graphs.** The following result is our main tool in this paper.

THEOREM 3. *Let $\Gamma$ be a vertex-transitive graph with degree $d$ such that $\mu_k > k$. Assume $d \geq \epsilon(3k-3)$. Then the set of $k$-atoms forms a complete set of imprimitivity blocks of the automorphism group of $\Gamma$. In particular, for any $k$-atom $A$, $\Gamma[A]$ is a vertex-transitive graph.*

*Moreover,*

$$\lambda_k(\Gamma) = |A|(d - d(A)) \geq (d - k + 2)(d - d(A)). \tag{3}$$

*Proof.* By Lemma 2, we have $\mu_k \geq d/\epsilon - k + 2 \geq 2k - 1$. Hence, by Lemma 1, two distinct atoms are disjoint. In particular, a $k$-atom is a block of imprimitivity of the automorphism group. The first part of the statement follows from the fact that $\Gamma$ is vertex-transitive. Inequality (3) now follows by the definitions. ◻

As a consequence of the above theorem, in the case that $\Gamma$ is a Cayley graph defined on a group $G$, the atom which contains the identity is a subgroup of $G$.

The above theorem implies that the isoperimetric connectivity achieves its optimal value in the following two important classes of graphs.

COROLLARY 4. *In a Cayley graph $\Gamma$ with prime order we have $\lambda_k(\Gamma) = dk - e_k(\Gamma)$ for $d \geq 3(k-3)/\epsilon$.*

*Proof.* By Theorem 3, $\mu_k(\Gamma) = k$. ◻

COROLLARY 5. *In a strongly connected arc-transitive graph $\Gamma$, we have $\lambda_k(\Gamma) = dk - e_k(\Gamma)$ for $d \geq 3(k-3)/\epsilon$.*

*Proof.* Notice that a strongly connected arc-transitive graph is also vertex transitive. Let $A$ be a $k$-atom of $\Gamma$. Assume $|A| \geq k+1$; then $\Gamma[A]$ must contain an interior arc $(x, y)$, since otherwise $|\omega(A)| \geq d(k+1) > \lambda_k(\Gamma)$. Also, there is $(x', y') \in \omega(A)$, since the graph is strongly connected. There is an automorphism $f$ such that $f(x) = x'$ and $f(y) = y'$. Therefore, $f(A) \neq A$ and $f(A) \cap A \neq \emptyset$, contradicting Theorem 3. ◻

The *girth* of a graph $\Gamma$ is the smallest cardinality of a directed cycle in $\Gamma$; we shall write it $g(\Gamma)$. For a symmetric graph, $g = 2$, one looks to the smallest cardinality of a cycle with size $\neq 2$ and calls it the undirected girth.

The class of graphs with large girth is widely studied in interconnection networks. The methods developed above allow us to give good lower bounds for $\lambda_k$ in this case. We mention the following simple application. In the next proof we shall use the fact that a vertex-transitive graph of girth $g$ and degree $d$ has order at least $d(g-1)+1$; see [5].

PROPOSITION 6. *Let $\Gamma$ be a connected vertex-transitive graph with girth $g \geq k+1$ and degree $d \geq \epsilon(3k-3)$. Then $\lambda_k(\Gamma) = dk - e_k(\Gamma)$.*

*Proof.* Suppose the contrary and let $A$ be a $k$-atom of $\Gamma$. We clearly have $1 \leq d(A) \leq d - 1$. By (3),

$$(d - d(A))(d(A)(g-1)+1) \leq \lambda_k < kd - e_k \leq kd - k + 1.$$

This leads to $d = d(A)$, which is a contradiction. ◻

We cannot replace $k \leq g-1$ by $k \leq g$ for $k \leq 3$ because of some obvious examples. It follows from a result in [7] that Proposition 6 holds for $k \geq 4$ when the condition $k \leq g - 1$ is replaced by $k \leq g$ for abelian Cayley graphs and there is a conjecture in [7] implying the validity of this for all vertex-transitive graphs.

Notice that the above result for symmetric graphs reduces to Mader's result mentioned in the Introduction. For antisymmetric graphs, $g \geq 3$ and hence we have the following.

Corollary 7. *Let $\Gamma$ be an antisymmetric connected vertex-transitive graph having at least four vertices. Then $\lambda_2(\Gamma) = 2d - 1$.*

The reader could formulate and prove the corresponding result for undirected graphs with undirected girth $g'$, using the well-known relation $|V| > (d-1)^{(g'-1)/2}$.

**5. The second isoperimetric connectivity.** The results obtained so far imply without difficulty that $\lambda_2$ is optimal in several cases.

Let $A$ be a 2-atom of a vertex-transitive graph $\Gamma$ with $\lambda_2$ nonoptimal. Lemma 2 gives in this case $|A| \geq d$ and, by inequality (3), $(d - d(A))d \leq (d - d(A))|A| = \lambda_2 < 2d - 1$. Therefore,

$$d(A) = d - 1 \quad \text{and} \quad \lambda_2 = |A|. \tag{4}$$

Corollary 8. *Let $\Gamma = (V, E)$ be a connected vertex-transitive graph with at least four vertices. Then $\lambda_2(\Gamma) = 2d - 2$ in the following two cases:*

(i) *$d > |V|/2$.*

(ii) *$\Gamma$ is symmetric and $|V|$ is odd.*

*Proof.* Suppose the contrary and let $A$ be a 2-atom.

Assume first that (i) holds. By (4), $d(A) = d - 1 \leq |A| - 1 \leq |V(\Gamma)|/2 - 1$, which is a contradiction.

Assume now that (2) holds. By (4), the graph obtained by deleting all edges inside the atoms is thus a symmetric graph with degree $= 1$, and hence a perfect matching, against the hypothesis that the graph has odd order. $\square$

The following result describes the local structure of Cayley graphs with nonoptimal $\lambda_2$.

Corollary 9. *Let $S$ be a set of generators for group $G$ with at least four vertices. Then $\lambda_2(Cay(G, S)) < 2|S| - 2$ if and only if there exists an element $t$ of $S$ such that the group $H$ generated by $S \setminus \{t\}$ satisfies $|S| \leq |H| \leq 2|S| - 3$.*

The above result reduces for $\lambda_2 = |S|$ to a result in [9].

We shall define the quotient graph $\Psi_k(\Gamma)$ which has the set of $k$-atoms as a vertex set. A $k$-atom $A$ is joined to a $k$-atom $B$ by $|B \cap \Gamma(A)|$ arcs in $\Psi_k(\Gamma)$. This multigraph will allow us to characterize vertex-transitive graphs with a nonoptimal second connectivity.

Proposition 10. *Let $\Gamma$ be a connected vertex-transitive graph such that $\mu_2(\Gamma) > 2$. Then $\Psi_2(\Gamma)$ is an arc-transitive multigraph, with degree $\lambda_2$.*

*Proof.* Notice that $\mu_2(\Gamma) > 2$ implies $d > 2$. Let $A$ be a 2-atom of $\Gamma$. By Theorem 3, $G[A]$ is a vertex-transitive graph.

Let us show that $\{A \cap \Gamma(B); |A \cap \Gamma(B)| \neq 0\}$ form a system of imprimitivity blocks. Let $x \in A \cap \Gamma(B)$ $y \in D \cap \Gamma(C)$ and let $f$ be an automorphism with $f(x) = y$.

By Theorem 3, $f(A) = D$. Let $x'$ be the unique element of $B \cap \Gamma^-(x)$. Let $y'$ be the unique element of $C \cap \Gamma^-(y)$. Now $f(x') \in \Gamma(f(x) \setminus D) = \{y'\}$. Hence we must have $f(x) = f(y)$. By Theorem 3, $f(B) = f(C)$. Now, $f(A \cap \Gamma(B)) = f(A) \cap \Gamma(f(B)) = D \cap \Gamma(C)$. The result is now obvious. $\square$

By Proposition 10, every vertex-transitive graph with nonoptimal $\lambda_2$ is obtained from an arc-transitive multigraph with degree $\lambda_2$ by inserting a vertex-transitive graph with order $\lambda_2$ at every vertex of the multigraph.

The above construction for $\lambda_2 = d(\Gamma)$ and $\Gamma$ symmetric reduces to the construction of a vertex-transitive graph with a nontrivial minimum edge cut described by Lovász and Plummer in [10, Lemma 5.5.26].

## REFERENCES

[1] C. Berge, *Graphes et Hypergraphes*, Monographies Universitaires de Mathématiques 37, Dunod, Paris, 1970.

[2] B. Bollobas, Ed., *Probabilistic Combinatorics and Its Applications*, Proc. Symp. Appl. Math. 44, AMS, Providence, RI, 1991.

[3] J. Fàbrega and M.A. Fiol, *On the extraconnectivity of graphs with large minimum degree and girth*, Discrete Math., 127 (1994), pp. 163–170.

[4] J. Fàbrega and M.A. Fiol, *On the extraconnectivity of graphs*, Discrete Math., 155 (1996), pp. 49–57.

[5] Y.O. Hamidoune, *An application of connectivity theory in graphs to factorization of elements in groups*, European J. Combin., 2 (1981), pp. 108–112.

[6] Y.O. Hamidoune, *An isoperimetric method in additive theory*, J. Algebra, 179 (1996), pp. 622–630.

[7] Y.O. Hamidoune, A.S. Lladó, and O. Serra, *Minimum order of loop network of given degree and girth*, Graphs Combin., 11 (1995), pp. 131–138.

[8] Y.O. Hamidoune, A.S. Lladó, and O. Serra, *An isoperimetric problem in Cayley graphs*, Theory Comput. Syst., 32 (1999), pp. 507–516.

[9] Y.O. Hamidoune and R. Tindell, *Vertex transitivity and super line connectedness*, SIAM J. Discrete Math., 3 (1990), pp. 524–530.

[10] L. Lovász and M.D. Plummer, *Matching Theory*, Ann. Discrete Math. 29, North–Holland, Amsterdam, 1986.

[11] W. Mader, *Über den Zusammenhäng symmetricher Graphen*, Arch. Math. (Basel), 21 (1970), pp. 331–336.

[12] W. Mader, *Eine Eigenschaft der Atome endlicher Graphen*, Arch. Math., 22 (1971), pp. 333–336.

[13] R. Tindell, *Edge Connectivity Properties in Symmetric Graphs*, preprint, 1983.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 2, pp. 145–153

# OPTIMAL ROUNDING OF INSTANTANEOUS FRACTIONAL FLOWS OVER TIME*

LISA FLEISCHER† AND JAMES B. ORLIN‡

**Abstract.** A transshipment problem with demands that exceed network capacity can be solved by sending flow in several waves. How can this be done in the minimum number, $T$, of waves, and at minimum cost, if costs are piecewise linear convex functions of the flow? In this paper, we show that this problem can be solved using $\min\{m, \log T, \frac{\log(m\Gamma U)}{1+\log(m\Gamma U)-\log(mU)}\}$ maximum flow computations and one minimum (convex) cost flow computation. Here $m$ is the number of arcs, $\Gamma$ is the maximum supply or demand, and $U$ is the maximum capacity. When there is only one sink, this problem can be solved in the same asymptotic time as one minimum (convex) cost flow computation. This improves upon the previous best algorithm to solve the problem without costs by a factor of $k$. Our solutions start with a stationary fractional flow and use rounding to transform this into an integral flow. The rounding procedure takes $O(n)$ time.

**Key words.** dynamic transshipment, network flows, convex costs, analysis of algorithms

**AMS subject classifications.** 68Q25, 90C08, 90C27

**PII.** S0895480198344138

## 1. Introduction.

Network flow theory deals with several fundamental problems including the shortest path problem, the maximum flow problem, the assignment problem, the minimum cost flow problem, and the multicommodity flow problem. A significant body of literature is devoted to these five problems. (See [1], for example.) In the 1960s, Ford and Fulkerson [8] introduced network flows over time to include time in the network model. Since then, network flows over time have been widely used in a variety of applications to transportation, financial planning, manufacturing, and logistics; see, for example, the surveys by Aronson [4] and Powell, Jaillet, and Odoni [18]. Despite the significance of flows over times to a wide range of problems, the class of network flows over time has not received nearly as much attention as the more classical network flow problems. Much of the work on problems of flow over time reduces the problem to a classical flow problem by using an exponentially sized, time-expanded graph [4, 18].

In this paper, we discuss a special case of flows over time: we assume all transit times are zero. This special case has been considered in [2, 5, 6, 12, 14, 15, 22], among others. Flows over time with zero transit times capture some time-related issues: they can be used to model instances when network capacities restrict the quantity of flow that can be sent at any one time. Solving these problems efficiently may help in finding a more efficient exact or approximate algorithm to solve harder problems involving flows over time with nonzero transit times or multicommodity demands.

### 1.1. The model.

A network $\mathcal{N} = (V, E, u)$ has node set $V$ of cardinality $n$ and arc set $E$ of cardinality $m$. Arc $e$ has capacity $u_e$ and $U := \max_e u_e$. Node $i$ has

*Received by the editors August 31, 1998; accepted for publication (in revised form) October 6, 1999; published electronically April 6, 2000.
http://www.siam.org/journals/sidma/13-2/34413.html

†Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027 (lisa@ieor.columbia.edu). The research of this author was supported by NSF grant EIA-9973858.

‡Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139 (jorlin@mit.edu). The research of this author was supported by ONR grant N00014-96-1-0051.

demand $\gamma_i$ with $\Gamma := \max_i |\gamma_i|$. Nodes with nonzero demands are called *terminals*. Terminals with negative demand are called *sources*; those with positive demand are called *sinks*. The sum of supplies over all terminals equals zero. We use $k$ to denote the number of terminals.

A *(static) transshipment problem* is defined on an arbitrary network with edge capacity vector $u$ and with node demand vector $\gamma$. The objective is to find a flow $f : E \to \mathbf{R}_{\geq 0}$ satisfying all demands. We define $f_{ji} = -f_{ij}$ for $ij \in E$. In other words, we seek $f$ such that $f_e \leq u_e$ for all edges $e$ and $\sum_j f_{ji} = \gamma_i$ for all nodes $i$.

A *transshipment over time* is a time-dependent flow $f(t)$ through $\mathcal{N}$ that satisfies a nonzero supply and demand vector $\gamma$ associated with $V$ and completes by a time bound $T$. A transshipment over time obeys edge capacity constraints $f(t) \leq u$ for all $t \in \{1, 2, \ldots, T\}$, flow conservation constraints $\sum_{t=1}^{r} \sum_j f_{ji}(t) \leq \gamma_i$ for all $r \in \{1, 2, \ldots, T\}$ and all $i \in V$ with $r_i \geq 0$. For $i$ with $r_i < 0$ we have $r_i \leq \sum_{t=1}^{r} \sum_j f_{ji}(t) \leq 0$. A transshipment over time also depletes all supplies and satisfies all demands by time $T$: $\sum_{t=1}^{T} \sum_{j \in V} f_{ji}(t) = \gamma_i$ for all $i \in V$. Although this is a discrete-time model for transshipment over time, the algorithm we present solves both the discrete-time and continuous-time problems, since, as noted in [5, 7], an optimal solution to the discrete-time problem can be easily transformed into an optimal solution for the continuous-time problem.

The *quickest transshipment problem* is a transshipment over time that depletes all supplies and satisfies all demands in the minimum possible time. The quickest transshipment defines the minimum time $T$ needed for the existence of a feasible transshipment. Solving a quickest transshipment problem with fixed supplies is useful for clearing a network after a communication breakdown.

To further generalize this model, we can include costs. Let $c_e$ be a cost function associated with edge $e$. In the problem discussed in this paper, $c_e$ is a function of the flow on the edge at a particular moment of time, but $c_e$ is not a function of time. The cost of a transshipment $f$ which completes by time $T$ is $\sum_{t=1}^{T} \sum_{e \in E} c_e(f_e(t))$. A *minimum cost transshipment over time* is a quickest transshipment that has minimum cost. In this paper, we present an algorithm to solve this problem that first finds the minimum time necessary for feasibility. Given this time bound, the algorithm then finds a minimum cost solution. The converse problem, finding a quickest, minimum cost transshipment, can be solved by first determining the minimum cost solution by assuming an infinite (or sufficiently large) time horizon and then searching for the minimum time bound that has a solution with this cost.

In this paper, we consider convex cost functions that are piecewise linear between successive integers. Since we want an integral solution, the assumption that the cost function is piecewise linear between integers is natural. It is useful to make this assumption, since now there always exists an integer solution that matches the cost of the cheapest fractional solution.

A flow over time is called *stationary* if it is the same in each time period. For each of the above problems, we can define a corresponding stationary version of the problem where, in addition to the above requirements, we insist the flow must be stationary. Fleischer [5] shows that every feasible transshipment over time problem has a stationary solution. The stationary problem can be solved by multiplying all arc capacities by $T$ and solving the resulting static transshipment, which is a maximum flow problem. Since all original supplies, demands, and capacities are integers, the solution is an integral flow. The stationary flow is then obtained by dividing this flow by $T$. This flow has the property that the rate of flow entering a terminal node $i$

is $\frac{\gamma_i}{T}$. A feasible stationary flow that completes in the minimum possible time can then be found using either discrete Newton's method [20] or binary search to find the minimum feasible time $T$. The following theorem is a result from [5].

THEOREM 1.1. *A quickest stationary flow can be found using at most* $\min\{\log T, m, \frac{\log(m\Gamma U)}{1+\log(m\Gamma U)-\log(mU)}\}$ *maximum flow computations.*

The maximum flow algorithm with the lowest complexity with respect to $m$ and $n$ is by Goldberg and Rao and runs in $O(m \min\{m^{1/2}, n^{2/3}\} \log \frac{n^2}{m} \log U)$ time [10].

**1.2. Our contribution.** Many of the applications of flows over time need integral solutions: when flows are of large objects, such as airplanes or train engines, the demands are often small, so fractional solutions are not very useful. By rounding carefully, we show how a stationary flow $f$ can be transformed into an integral flow over time maintaining the same flow balance constraints and such that this new flow $g$ satisfies $c(g) \leq c(f)$. The rounding technique is simple and operates in two stages on the support graph of edges with fractional flow. First, cycles in this graph are canceled, and then the edges in each tree are rounded in linear time, starting from a leaf and proceeding in a depth or breadth first manner. It is useful to think of the interval of time $[0, T)$ as a cycle on a clock. A *circular interval* is an interval modulo $T$. If the stationary flow in arc $e$ has value $f_e$ per time unit, then the integral solution we find has the property that the flow in arc $e$ has value $\lfloor f_e \rfloor$ in a circular interval and value $\lceil f_e \rceil$ for the remainder of the time (also a circular interval).

Our rounding technique takes only linear time. Since finding a quickest stationary flow takes more time, this implies that a quickest transshipment can be found in the same asymptotic time as finding the minimum feasible time $T$. This is the fastest known algorithm to solve this problem and improves upon [5] by almost a factor of $k$. If there is just one sink, then it is known [5] that the minimum feasible time $T$ can be found in strongly polynomial time using one call of the parametric maximum flow algorithm of Gallo, Grigoriadis, and Tarjan [9]. Thus, our rounding technique implies an improvement over the previous best algorithm by a factor of $k$ [5] for this special case as well. The algorithm of Gallo, Grigoriadis, and Tarjan is based on the Goldberg and Tarjan [11] maximum flow algorithm and runs in the same asymptotic time. In addition, since there always exists a minimum (convex) cost transshipment over time that is stationary (see section 2), our rounding technique yields a fast algorithm for computing an integral minimum (convex) cost transshipment over time.

The model discussed here solves the problem when the demands and the supplies are fixed at the beginning of the problem and there is no external change to these values. However, everyday usage often involves continuous streams of traffic. In section 4, we show that all the algorithms presented here allow for constant streams of external flow into or out of any node in the network.

The model as presented in this paper can also handle finite buffer capacity at the nonsource nodes of the network. Suppose storage capacity at node $i$ is $a_i$. Then we have the additional constraint (in the discrete-time model) that $\sum_{t=1}^{r} \sum_j f_{ji}(t) \leq a_i + \gamma_i$ for all $r \in \{1, \ldots, T\}$ and all $i \in V$ with $r_i \geq 0$. A stationary flow never increases the absolute value of supply or demand at any node, and our rounding technique maintains this property. Together with Theorem 2.1 this implies that our solution solves the problem for all $a \geq 0$.

**1.3. Related work.** The fastest previous algorithm for solving an integral, quickest transshipment problem with zero transit times uses $O(k \log \Gamma)$ maximum flow computations and $k$ minimum cost flow computations [5]. Previously, Hoppe and Tardos [13] described the only known polynomial time algorithm to solve the quickest transshipment problem with general transit times. However, their algorithm repeatedly calls the ellipsoid method as a subroutine, so it is theoretically slow and also impractical. Their work improves on the theoretical complexity of previous approaches, however. Traditional approaches to solving the transshipment over time consider the discrete-time model and make use of the time-expanded network—a network containing $T$ copies of the original network [4, 18]. For fine discretizations, this network is very large and not practical to work with.

A slightly different type of convex cost flow over time problem is discussed by Orlin [16]. He looks at infinite horizon convex cost flows over time and describes an algorithm that finds the minimum average convex cost circulation that repeats indefinitely in a network with general transit times.

Our paper and the above mentioned results all look at discrete-time problems. Flows over time have also been considered in the continuous-time setting [2, 3, 17, 19]. Most of the work in this area has examined networks with time-varying edge capacities, storage capacities, or costs. The focus of this research is on proving the existence of optimal solutions for classes of time-varying functions and proving the convergence of algorithms that eventually find solutions. These algorithms are not polynomial and implementations do not seem able to handle problems with more than a few nodes. For the case in which capacity functions are constant, Fleischer and Tardos [7] extend the polynomial, discrete-time transshipment over time algorithm in [13] to work in the continuous-time setting.

There are some additional continuous-time problems that have polynomial time algorithms. A *universally quickest transshipment* is a quickest transshipment that simultaneously minimizes the amount of excess left in the network at every moment of time. An optimal solution may require fractional flow sent over fractional intervals of time. There is a two-source, two-sink example for which a universally quickest transshipment does not exist [5]. Hajek and Ogier [12] describe an algorithm that solves the universally quickest transshipment problem in networks with multiple sources, a single sink, and zero transit times. Their algorithm uses $O(n)$ maximum flow computations. In [5], it is described how this problem can be solved in $O(mn\log(n^2/m))$ time—using the parametric maximum flow algorithm of Gallo, Grigoriadis, and Tarjan in conjunction with maximum flows. This algorithm extends easily to the minimum (convex) cost problem by substituting minimum (convex) cost flow computations for maximum flow computations.

Another continuous-time problem that has a polynomial time algorithm is the universally quickest flow problem with piecewise constant capacity functions with at most $l$ breakpoints. This is the universally quickest transshipment problem with just one source and sink. However, now capacities are allowed to vary over time. For this problem, Ogier [15] describes an algorithm that uses $nl$ maximum flow computations in a graph on $nl$ nodes and $(m+n)l$ arcs. In [6], it is shown how this can be improved to run in the same asymptotic time as one preflow-push maximum flow computation in a graph with $nl$ nodes and $(n+m)l$ arcs. A preflow-push maximum flow algorithm runs in $O(mn\log(n^2/m))$ time [11] on a graph with $n$ nodes and $m$ arcs.

## 2. Minimum cost flows over time.

As stated in section 1.1, any feasible transshipment has a stationary solution. In this section, we show that there is always a minimum cost solution that is stationary.

THEOREM 2.1. *There is an optimal stationary solution to any feasible minimum (convex) cost transshipment over time, even with nonzero buffer capacity.*

*Proof.* Let $f$ be any minimum convex cost transshipment over time. Let $f^T$ be defined by $f_e^T := \sum_{t=1}^{T} f_e(t)$, the total flow on edge $e$. By convexity of the cost function, the cost of this flow is at least as large as the cost of the flow $g = f^T/T$, which is a feasible, stationary transshipment over time. Note that this argument is independent of $\sum_{t=1}^{r} \sum_j f_{ji}(t)$, the amount of buffer capacity used at node $i$ at time $r$. $\square$

A stationary, minimum (convex) cost transshipment over time can be obtained by computing a minimum cost static transshipment in the network with capacities multiplied by $T$ and with the cost function $\bar{c}$ defined by $\bar{c}(f) = c(f/T) \times T$. Note that $\bar{c}$ is convex when $c$ is and that $\bar{c}$ is minimized at $f$ if and only if $c$ is minimized at $f/T$. Thus the resulting flow $f$, when divided by $T$, gives a feasible flow over time whose cost equals $\bar{c}(f)$ and is minimum among all feasible transshipments over time.

**3. Rounding a stationary flow.** In this section, we explain how to transform a stationary transshipment into an integral transshipment over time that has the same cost. We do this by considering the graph $G'$ consisting of just the arcs with fractional flow. The algorithm proceeds in two stages: it first cancels cycles in this graph. Then, one arc at a time, it determines an interval in which the flow on the arc will be rounded up to the nearest integer. For the remaining time, the flow on this arc is rounded down to the nearest integer. This is done in a way to ensure that the total flow sent along the arc in the interval $[0, T)$ remains the same. Since flow costs are linear between successive integers, this then implies that the cost of sending the flow remains the same.

**3.1. Canceling fractional flow cycles.** If the graph consisting of the edges with fractional flow contains cycles, then these cycles can be canceled by sending flow around the cycle until the flow on at least one edge in the cycle becomes integral. Iterating, we eventually obtain a stationary flow such that the graph consisting of edges with fractional flow is acyclic. Such a solution is a basic feasible solution for the stationary problem.

Let $f$ denote the optimal stationary flow. Let $\lfloor f \rfloor$ denote $f$ rounded down and let $\lceil f \rceil$ denote $f$ rounded up. Consider the stationary transshipment problem with the additional constraints $\lfloor f \rfloor \leq f \leq \lceil f \rceil$. Since we assume that costs are linear between successive integers, this modified problem is a minimum cost flow problem even if the original problem has convex costs. We can convert $f$ to a basic feasible flow for this problem by canceling fractional flow cycles. We can cancel a cycle by sending flow in either direction. Optimality conditions for minimum cost flow imply that if a cycle exists, the cost of the cycle (obtained by summing the costs on all edges of the cycle) must be 0. Thus flow can be canceled in either direction while maintaining the cost of the initial solution. It takes $O(n)$ time to find an (undirected) cycle in $G'$ using depth first search, and at most $m$ cycles are canceled since each cancellation removes at least one edge from the set of edges with fractional flow. Thus all cycles are cancelled in $O(mn)$ time.

For the minimum cost transshipment over time problem, this step is not a bottleneck, since computing a minimum cost flow takes more time. However, for the quickest transshipment, the maximum flow algorithm of Goldberg and Rao [10] may be faster on some inputs than the above cycle canceling step. Hence, we note that it is possible to cancel cycles in $O(m \log n)$ time using the dynamic tree data structure of Sleator and Tarjan [21]. The details of this procedure are described in their paper.

**3.2. Rounding fractional flow.** In this section, we assume that $G'$, the graph of fractional flows, is a forest. The rounding algorithm considers each tree in this forest independently, roots each tree at an arbitrary leaf, and rounds the flow on the edges in a tree starting from the root and spreading throughout the tree in a breadth first manner. For each arc, the rounding procedure determines an interval in which the flow on the arc will be rounded up to the nearest integer. For the remaining time, the flow on this arc is rounded down to the nearest integer. The algorithm does this while ensuring that the total flow sent along the arc in the interval $[0, T)$ remains the same. This then implies that the cost of sending the flow remains the same.

We assume a tree $F$ of fractional flows is stored in a simple data structure so that the tree has a root, which is assigned to be a node with degree one. Thus the root node has a single child. The arcs are oriented according to the direction that the fractional flow travels down the arc. Without loss of generality, we assume all arcs in $F$ are directed away from the root: each arc with fractional flow $\alpha$ directed toward the root is replaced by two arcs, one with unit flow directed towards the root and one with fractional flow $1-\alpha$ directed away from the root. Only the new arc with fractional flow is kept in the tree.

For each arc $(i, j)$ of the tree, we define the *predecessor* of $j$ to be $i$ and denote this by $\text{pred}(j) = i$. Suppose that nodes 2, 3, 4, and 5 are the children of node 1. We refer to node 2 as the *left child* of node 1 and refer to node 3 as the *right sibling* of node 2. Similarly node 4 (respectively, node 5) is the right sibling of node 3 (respectively, node 4).

Recall that we are considering all addition and intervals to be modulo $T$ so that if $a > b$, then $[a, b)$ denotes the circular interval $[a, T) \cup [0, b)$.

Let $x(u, v)$ denote the fractional part of the flow in arc $(u, v)$. Let $d(v)$ denote the fractional part of the rate of increase of supply at node $v$. That is, $d(v)$ is the fractional part of $\frac{\gamma_v}{T}$. Rounding the flows entering and leaving $v$ implicitly has the effect of rounding the rate of increase of supply at $v$. We make this explicit to provide a cleaner explanation by introducing a new node $v'$ with arc $(v, v')$ and set $x(v, v') = d(v)$. The transformed problem has fractional rates of increase of supply $d'$ defined by $d'(v) = 0$ and $d'(v') = d(v)$. Rounding up the flow on arc $(v, v')$ in the transformed problem corresponds to rounding up the rate of increase of supply at node $v$ in the original problem. With this transformation, all internal nodes $v$ of $F$ appear as nonterminal nodes. Thus for all internal nodes of $F$, the fractional part of the flow into $v$ equals the fractional part of the sum of flows out of $v$.

Let $a(u, v)$ denote the "start time" for the round up of arc $(u, v)$. It is the first period in the cyclic interval in which flow in $(u, v)$ is rounded up. Let $b(u, v)$ denote the start time for the round down of flow in arc $(u, v)$. Given $a(u, v)$, we define $b(u, v) := a(u, v) + T \times x(u, v) \mod T$. The flow on arc $(u, v)$ is rounded up in the circular interval $[a(u, v), b(u, v))$ and rounded down in the circular interval $[b(u, v), a(u, v))$.

We determine $a(u, v)$ as follows and discuss a concrete example below and in Figure 3.1. If $u$ is a node in $F$ with predecessor node $w$, and $v$ is the left child of $u$, then we set $a(u, v) = a(w, u)$. If $w$ is a node with predecessor $u$, and $v$ is the right sibling of node $w$, then we set $a(u, v) = b(u, w)$. Assuming that the stationary flow comes from a basic feasible solution to a static flow problem with integer data, then $x(u, v) \times T$ is integral, and hence the rounded intervals have integer length.

For example, consider the flow in Figure 3.1. At every unit of time, fractional flow of value 1/3 is entering node $v$ and flows of 2/3, 1/2, and 1/6 are leaving node $v$.

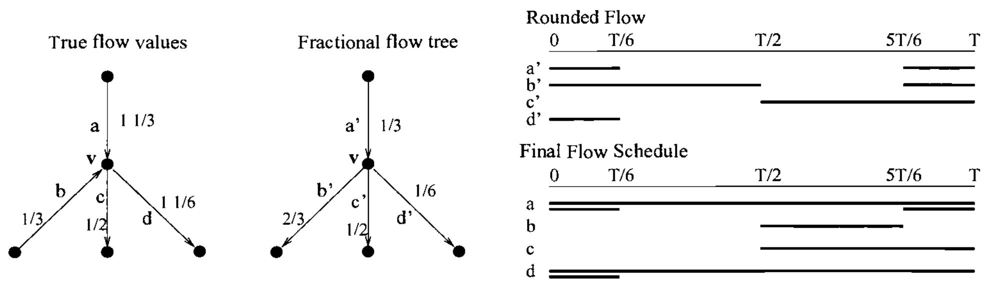


FIG. 3.1. *An example of how flow is rounded on a subtree and of the discussion in the proof of Theorem* 3.1. *This subtree has the interval* $[5T/6, T/6)$ *passed to it from higher in the tree. The rounded flow (top set of intervals) covers the whole interval* $[0, T)$ *at least once and the key interval* $[5T/6, T/6)$ *exactly one more time.*

Suppose we have previously rounded the flow in the incoming arc up to the nearest integer in the interval $[5T/6, T/6)$. Then we round the outgoing flows as follows: Fractional flow of 2/3 is rounded to 1 from $t = 5T/6$ to $t = T$ and from $t = 0$ to $t = T/2$ and is rounded down to 0 for the remaining interval $[T/2, 5T/6)$. Fractional flow of 1/2 is rounded up to 1 in the interval $[T/2, T)$ and is rounded down to 0 in the remaining interval $[0, T/2)$. Fractional flow of 1/6 is rounded up to 1 in the interval $[0, T/6)$. Since the sum of the fractional flows leaving $v$ is one more than the amount of fractional flow entering $v$, there must be an additional unit of integral flow entering $v$ at every time step. This additional unit of flow entering $v$ is used to balance the rounded flows now leaving $v$.

Each arc $(u, v)$ gets rounded up $x(u, v)$ of the time and rounded down $1 - x(u, v)$ of the time, thus leading to a net balance of flow out of node $u$. Also, the upper and lower bounds on flows are automatically satisfied. So the key issue is whether there is balance in each time unit.

THEOREM 3.1. *The rounding algorithm transforms a basic, feasible stationary transshipment into a feasible integral transshipment over time in* $O(n)$ *time.*

*Proof.* Let $v$ be a nonterminal node of $F$ with $r$ children $\{u_1, \ldots, u_r\}$ and with $\text{pred}(v) = w$. At every unit of time, the stationary flow $f$ satisfies the following: the fractional part of flow into $v$ equals the fractional part of the sum of flows leaving $v$. More precisely,

$$\sum_{i=1}^{r} x(v, u_i) \times T = x(w, v) \times T + \kappa T, \quad \kappa \in \mathbf{Z}^+.$$

This means that the amount of integral flow arriving at node $v$ before the rounding is $\kappa T$ more than the sum of integral flows leaving node $v$. Thus at any moment of time, we can round up the flow on at least $\kappa$ of the fractional edges leaving node $v$ and supply the flow on these edges with the extra integer flow arriving at $v$. In fact, for our rounding scheme to maintain flow conservation without using node storage, our solution needs to round up the flow on $\kappa$ arcs leaving $v$ at all times and on an additional arc leaving $v$ for the interval of time in which we rounded up the flow on the arc entering $v$. This is exactly what our rounding scheme accomplishes. Starting at the moment of time when we round up the flow on the arc entering $v$, we schedule rounded up flow contiguously, tracing the $[0, T)$ interval $\kappa$ times and the $[a(w, v), b(w, v))$ interval an additional time by starting at $a(w, v)$, scheduling unit flow of total value

$\sum_{i=1}^{r} T \times x(v, u_i)$, and ending at $b(w, v)$. (Note that $a(w, v) + \sum_{i=1}^{r} T \times x(v, u_i) = b(w, v) \bmod T$.)

If $v$ is a source or sink, then $v$ is a leaf (or the root) of $F$, by construction, and the only rounding we do is on one arc entering (or leaving) $v$. In this case, the rounding amounts to rounding the rate of increase of supply at $v$, and we needn't worry about balance at every time step.

The rounding procedure spends a constant amount of time considering each arc in each fractional tree. Hence, the total time to perform the rounding is bounded by the number of nodes in the graph. $\square$

**4. Incoming and outgoing traffic.** While solving a transshipment problem with fixed supplies and demands is useful for clearing a network after a communication breakdown, everyday usage more often involves continuous streams of traffic. In this section, we show that the algorithms presented in this paper, like the algorithm in [5, 12], allow for constant streams of flow into or out of any node in the network.

The algorithm described in the preceding sections first finds a fractional flow that is constant over time and then rounds the flow. To find the fractional flow, we sum up capacities per unit time over the length of our time horizon. We treat the constant rate external flows in the same manner, multiplying the value of the rate of incoming flow by $T$ and treating these as additional supplies and demands. Any solution to the resulting transshipment over time problem will have constant rates of flow from these sources and sinks of additional value equal to the rate of external flow to these nodes. Since we assume all rates of flow are integral, this contributes only integral flows to the resulting flow over time and hence is not affected by the rounding.

## REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] E. J. Anderson and P. Nash, *Linear Programming in Infinite-Dimensional Spaces*, John Wiley & Sons, New York, 1987.

[3] E. J. Anderson and A. B. Philpott, *A continuous-time network simplex algorithm*, Networks, 19 (1989), pp. 395–425.

[4] J. E. Aronson, *A survey of dynamic network flows*, Ann. Oper. Res., 20 (1989), pp. 1–66.

[5] L. Fleischer, *Faster algorithms for the quickest transshipment problem with zero transit times*, in Proceedings of the Ninth Annual ACM–SIAM Symposium on Discrete Algorithms, San Francisco, CA, January 25–27, 1998, SIAM, Philadelphia, PA, pp. 147–156; SIAM J. Optim., submitted.

[6] L. Fleischer, *Universally maximum flow with piecewise constant capacities*, in Integer Programming and Combinatorial Optimization: 7th International IPCO Conference, Graz, Austria, June 1999, G. Cornuejols, R. E. Burkard, and G. J. Woeginger, eds., Lecture Notes in Comput. Sci. 1610, Springer-Verlag, New York, pp. 151–165.

[7] L. Fleischer and É. Tardos, *Efficient continuous-time dynamic network flow algorithms*, Oper. Res. Lett., 23 (1998), pp. 71–80.

[8] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[9] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, *A fast parametric maximum flow algorithm and applications*, SIAM J. Comput., 18 (1989), pp. 30–55.

[10] A. V. Goldberg and S. Rao, *Beyond the flow decomposition barrier*, J. ACM, 45 (1998), pp. 783–797.

[11] A. V. Goldberg and R. E. Tarjan, *A new approach to the maximum flow problem*, J. ACM, 35 (1988), pp. 921–940.

[12] B. Hajek and R. G. Ogier, *Optimal dynamic routing in communication networks with continuous traffic*, Networks, 14 (1984), pp. 457–487.

[13] B. Hoppe and É. Tardos, *The quickest transshipment problem*, in Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, January 22–

24, 1995, SIAM, Philadelphia, 1995, pp. 512–521.

[14] F. H. Moss and A. Segall, *An optimal control approach to dynamic routing in networks*, IEEE Trans. Automat. Control, 27 (1982), pp. 329–339.

[15] R. G. Ogier, *Minimum-delay routing in continuous-time dynamic networks with piecewise-constant capacities*, Networks, 18 (1988), pp. 303–318.

[16] J. B. Orlin, *Minimum convex cost dynamic network flows*, Math. Oper. Res., 9 (1984), pp. 190–207.

[17] A. B. Philpott, *Continuous-time flows in networks*, Math. Oper. Res., 15 (1990), pp. 640–661.

[18] W. B. Powell, P. Jaillet, and A. Odoni, *Stochastic and dynamic networks and routing*, in Network Routing, Handbooks in Oper. Res. Management Sci. 8, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, eds., North-Holland, Amsterdam, The Netherlands, 1995, pp. 141–295.

[19] M. C. Pullan, *An algorithm for a class of continuous linear programs*, SIAM J. Control Optim., 31 (1993), pp. 1558–1577.

[20] T. Radzik, *Parametric flows, weighted means of cuts, and fractional combinatorial optimization*, in Complexity in Numerical Optimization, P. M. Pardalos, ed., World Scientific, River Edge, NJ, 1993, pp. 351–386.

[21] D. D. Sleator and R. E. Tarjan, *A data structure for dynamic trees*, J. Comput. System Sci., 26 (1983), pp. 362–391.

[22] G. I. Stassinopoulos and P. Konstantopoulos, *Optimal congestion control in single destination networks*, IEEE Trans. Comm., 33 (1985), pp. 792–800.

SIAM J. Discrete Math. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 2, pp. 154–178

# OVERLAP AND COVERING POLYNOMIALS WITH APPLICATIONS TO DESIGNS AND SELF-DUAL CODES[*]

GERALD J. JANUSZ[†]

**Abstract.** For a binary linear code $\mathcal{C}$ and a vector $u$, we define and study the overlap polynomial $F_{\mathcal{C}}^{(u)}(Z,X,D)$ which is a sum over the words in $\mathcal{C}$ of terms that takes into account the weight of the element of $\mathcal{C}$ and its overlap with the vector $u$. We obtain a generalized MacWilliams relation between $F_{\mathcal{C}}^{(u)}(Z,X,D)$ and $F_{\mathcal{C}^\perp}^{(u)}(Z,X,D)$ and a recursion equation for determining the overlap polynomials of a self-dual code in terms of overlap polynomials corresponding to vectors of lower weight. We apply the recursion to determine the weight enumerators of the cosets of the (putative) $[72,36,16]$ self-dual code of Type II, up to a small number of parameters.

The overlap polynomials are expressed in terms of covering polynomials $g_{\mathcal{C}}^{(u)}(Z,X)$ that are sums over the elements of $\mathcal{C}$ that cover $u$. The design strength $\delta(\mathcal{C})$ is closely related to the covering polynomials and we use them to get a sharpened version of the Assmus–Mattson theorem. In many cases we are able to conclude that certain codes hold $\delta(\mathcal{C})$-designs but no weight class holds a $1+\delta(\mathcal{C})$-design. We study the design strength of self-dual, doubly even extremal codes and, in the case of length divisible by 24, improve the well-known estimate $\delta(\mathcal{C}) \geq 5$ to state that either $\delta(\mathcal{C}) \geq 7$ or $\delta(\mathcal{C}) = 5$ and no weight class holds a 6-design; similar results hold in the case of other lengths.

Examples are given using a self-dual code $[32,16,8]$, certain cyclic subcodes of the punctured Reed–Muller code $\mathcal{R}(2,m)^*$, the first order RM-code, and a $[16,9,4]$ code in which all six of the weight classes hold 2-designs but one nontrivial weight class also holds a 3-design.

**Key words.** self-dual code, design, extremal code, weight enumerator

**AMS subject classifications.** 94B05, 94B60, 94B65, 05B05

**PII.** S0895480198341766

**Introduction.** We study binary, linear codes using two related enumerator polynomials which we call an overlap polynomial and a covering polynomial. The overlap polynomial $F_{\mathcal{C}}^{(u)}(Z,X,D)$ is defined for a code $\mathcal{C}$ and any vector $u$ and takes into account the elements of $\mathcal{C}$ having a given weight and given overlap with the vector $u$; the covering polynomial $g_{\mathcal{C}}^{(u)}(Z,X)$ takes into account the elements of $\mathcal{C}$ having a given weight and covering the vector $u$. See section 1 for precise definitions. We obtain a MacWilliams-type relation between $F_{\mathcal{C}}^{(u)}(Z,X,D)$ and $F_{\mathcal{C}^\perp}^{(u)}(Z,X,D)$ and exploit it to obtain similar relations for the covering polynomials.

The covering polynomials lend themselves naturally to the study of the design properties of a code because a code holds a $t$-design if and only if the correspondence $u \to g_{\mathcal{C}}^{(u)}(Z,X)$ is constant on the vectors of weight $r$ for each $r \leq t$. If $\delta(\mathcal{C})$ denotes the largest integer $t$ such that $\mathcal{C}$ holds $t$-designs in each weight class then there exist vectors $u$ and $v$ each of weight $1+\delta(\mathcal{C})$ such that the difference $g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X)$ is nonzero. We study how this polynomial factors and the investigation leads to a proof of the Assmus–Mattson theorem in a sharper form than usual (Theorem 2.6). In cases where the Assmus–Mattson bound gives the exact value of $\delta(\mathcal{C})$, we may assert that no nontrivial weight class holds a $t$-design with $t > \delta(\mathcal{C})$. The Assmus–Mattson theorem gives information about the design strength of self-dual extremal codes of Type II; namely, $\delta(\mathcal{C}) \geq 5 - 2\epsilon$. Our results sharpen this as follows.

[*]Received by the editors July 13, 1998; accepted for publication (in revised form) September 22, 1999; published electronically April 6, 2000.
http://www.siam.org/journals/sidma/13-2/34176.html

[†]Department of Mathematics, University of Illinois, 1409 W. Green Street, Urbana, IL 61801 (janusz@math.uiuc.edu).

THEOREM 2.10. *Let $\mathcal{C}$ be an extremal, self-dual code with all weights divisible by 4 and length $n = 24\mu + 8\epsilon$, $\epsilon = 0, 1,$ or 2. Then either $\delta(\mathcal{C}) \geq 7 - 2\epsilon$ or $\delta(\mathcal{C}) = 5 - 2\epsilon$ and no nontrivial weight class holds a $1 + \delta(\mathcal{C})$-design.*

In particular for an extremal code $\mathcal{C}$ with length divisible by 24, if one nontrivial weight class holds a 6-design, then every weight class holds a 7-design. There are no known examples where an extremal code holds a 6-design in any nontrivial weight class. This result makes it less likely that any exist.

In Corollary 2.8 we give another solution of Research Problem 6.1 in [M-S, p. 161] which asserts $w_1 < d_1 + s - \epsilon_{\mathcal{C}}$ and $d_1 < w_1 + s' - \epsilon_{\mathcal{C}^\perp}$ except in certain specific trivial situations. Here $\mathcal{C}$ has $s$ nonzero weights and minimum distance $d_1$ while $\mathcal{C}^\perp$ has $s'$ nonzero weights and minimum distance $w_1$ and $\epsilon_{\mathcal{C}}$, $\epsilon_{\mathcal{C}^\perp}$ are 1 or 0 depending on whether the all 1s vector is or is not in $\mathcal{C}$ or $\mathcal{C}^\perp$. According to the preface of the third edition of [M-S] this problem has been previously solved.

For applications to self-dual codes we obtain a recursion equation (3.1) that determines the covering polynomials $g_{\mathcal{C}}^{(u)}(Z, X)$ in terms of covering polynomials corresponding to vectors of lower weight by a system of linear equations. We apply this to the classic problem of the possibility of a self-dual, extremal Type II code $\mathcal{C}_{72}$ with parameters $[72, 36, 16]$. The question of the existence of such a code was raised by Neal Sloane in 1977 and remains open today. The weight enumerator of the code is uniquely determined by Gleason's theorem. Our contribution to this problem is to give the weight enumerators for all the cosets $u + \mathcal{C}_{72}$. The cosets of weight 1 through 5 and 11 and 12 are determined unambiguously with no parameters. However, the determination is made on the assumption that there is a coset of weight 11 or 12. Even this is not known for certain, as the current information states that the covering radius is between 10 and 12. The weight enumerators for cosets of weights 6, 7, 9, and 10 involve one undetermined parameter, and the weight 8 cosets require two parameters. The method for doing this computation is to use the recursion equation to determine the overlap polynomials $F_{\mathcal{C}}^{(u)}(Z, X, D)$ for $\mathrm{wt}(u) \leq 12$ and from them determine the weight enumerator of $u + \mathcal{C}_{72}$.

A number of other examples are given. An observation that seems to be new is the existence of a code where every weight class holds a 2-design and exactly 1 (of the 5) nontrivial weight classes holds a 3-design. Another is a subcode of the punctured Reed–Muller code $\mathcal{R}(2, m)^*$ whose dual, for $m = 9$, has weights $n = 511$ and every integer on the interval $[5, 506]$. These weight classes all hold 2-designs but none hold 3-designs. Another example gives a code with parameters $[32, 16, 8]$ in which the recursion equation is used to prove the design strength $\delta = 1$, whereas the Assmus–Mattson theorem gives no information.

**Notation and known results.** The field of 2 elements is denoted by $\mathbf{F}$ and the space of row vectors with $n$ coordinates is denoted by $\mathbf{F}^n$. For a vector $u = (u_1, \ldots, u_n) \in \mathbf{F}^n$, the support of $u$ is the set $\mathrm{supp}(u) = \{i : 1 \leq i \leq n,\ u_i \neq 0\}$. The *weight* of a vector $u$ is the number of nonzero coordinates and is denoted by $\mathrm{wt}(u)$. We use the term *$u$ covers $v$* to mean $\mathrm{supp}(v) \subseteq \mathrm{supp}(u)$ and we write $v \leq u$ in this case. The vector $\mathbf{1} = (1, 1, \ldots, 1)$ is the all one vector; the complement of a vector $u$ is $\bar{u} = \mathbf{1} + u$. The coordinatewise product of $u$ and $v$ is denoted by

$$u * v = (u_1, \ldots, u_n) * (v_1, \ldots, v_n) = (u_1 v_1, \ldots, u_n v_n).$$

For any subspace $\mathcal{C} \subseteq \mathbf{F}^n$, the dual space is

$$\mathcal{C}^\perp = \{x \in \mathbf{F}^n : x \cdot c = 0, \quad \text{for all } c \in \mathcal{C}\},$$

where the dot product is defined as usual by

$$x \cdot c = (x_1, \ldots, x_n) \cdot (c_1, \ldots, c_n) = \sum_{i=1}^{n} x_i c_i.$$

If $\mathcal{C} \subseteq \mathbf{F}^n$ is any linear code, the *homogeneous weight enumerator* of $\mathcal{C}$ is the polynomial

$$W_{\mathcal{C}}(X, Y) = \sum_{w \in \mathcal{C}} X^{n-\mathrm{wt}(w)} Y^{\mathrm{wt}(w)};$$

the (inhomogeneous) polynomial $W_{\mathcal{C}}(Y) = \sum_{w \in \mathcal{C}} Y^{\mathrm{wt}(w)}$ is also called the weight enumerator of $\mathcal{C}$

For convenient reference we list the following well-known facts.

Theorem M (MacWilliams formula). *For a vector $v \in \mathbf{F}^n$,*

$$\sum_{u \in \mathbf{F}^n} (-1)^{u \cdot v} X^{n-\mathrm{wt}(u)} Y^{\mathrm{wt}(u)} = (X+Y)^{n-\mathrm{wt}(v)} (X-Y)^{\mathrm{wt}(v)}.$$

For any function $f$ defined on $\mathbf{F}^n$, the Hadamard transform of $f$ is the function $\hat{f}$ defined on $\mathbf{F}^n$ by

$$\hat{f}(v) = \sum_{u \in \mathbf{F}^n} (-1)^{u \cdot v} f(u).$$

Theorem T (transform property). *For any linear code $\mathcal{D} \subseteq \mathbf{F}^n$ and any function $f$ defined on $\mathbf{F}^n$, there holds the relation*

$$\sum_{v \in \mathcal{D}} \hat{f}(v) = |\mathcal{D}| \sum_{u \in \mathcal{D}^\perp} f(u).$$

**1. Overlap and covering polynomials.** We define the *(homogeneous) overlap polynomial* associated with the linear code $\mathcal{C}$ corresponding to a vector $u \in \mathbf{F}^n$ to be

$$F_{\mathcal{C}}^{(u)}(Z, X, D) = \sum_{w \in \mathcal{C}} Z^{n-\mathrm{wt}(w)} X^{\mathrm{wt}(w)-\mathrm{wt}(w*u)} (X+D)^{\mathrm{wt}(w*u)}.$$

Note that when $D = 0$, this is independent of $u$ and equals the homogeneous weight enumerator of $\mathcal{C}$. The inhomogeneous polynomial obtained by setting $Z = 1$ is also called *the overlap polynomial* corresponding to $u$.

We list a few elementary properties of overlap polynomials to indicate how they might be useful.

Proposition 1.1. *Let $u \in \mathbf{F}^n$ and $\mathcal{C}$ be any linear subcode of $\mathbf{F}^n$. The overlap polynomial $F_{\mathcal{C}}^{(u)}(Z, X, D)$ has the following properties:*

(i) $F_{\mathcal{C}}^{(\bar{u})}(Z, X, D) = F_{\mathcal{C}}^{(u)}(Z, X+D, -D)$.

(ii) *The homogeneous weight enumerator of the coset $u + \mathcal{C}$ (a nonlinear code) is given by*

$$W_{u+\mathcal{C}}(A, B) = \left(\frac{B}{A}\right)^{\mathrm{wt}(u)} F_{\mathcal{C}}^{(u)}(A, B, (A^2 - B^2)/B).$$

(iii) *The subcode $\mathcal{C} \cap \mathcal{C} * u$ consisting of all elements of $\mathcal{C}$ covered by $u$ has weight enumerator*

$$W_{\mathcal{C}\cap\mathcal{C}*u}(A,B) = F_{\mathcal{C}}^{(u)}(A,0,B).$$

(iv) *The code $\mathcal{C} * u$ obtained by projecting $\mathcal{C}$ onto the coordinates of $u$ has weight enumerator*

$$W_{\mathcal{C}*u}(A,B) = A^{-n+\mathrm{wt}(u)}\, F_{\mathcal{C}}^{(u)}(A,A,B-A)/F_{\mathcal{C}}^{(u)}(1,1,-1).$$

*Proof.* One easily verifies that $\mathrm{wt}((\bar{u}) * v) = \mathrm{wt}(v) - \mathrm{wt}(u * v)$ for any vectors $u, v \in \mathbf{F}^n$ so that property (i) follows immediately. Property (ii) requires the equation $\mathrm{wt}(u + c) = \mathrm{wt}(u) + \mathrm{wt}(c) - 2\mathrm{wt}(u * c)$ which holds for any $u, c \in \mathbf{F}^n$.

The overlap weight enumerator is defined as a sum over all elements of $\mathcal{C}$; if that sum is restricted to those elements $c$ of $\mathcal{C}$ that are covered by $u$, we get a sum of terms

$$Z^{n-\mathrm{wt}(w)}X^{\mathrm{wt}(w)-\mathrm{wt}(w*u)}(X+D)^{\mathrm{wt}(w*u)} = Z^{n-\mathrm{wt}(w)}(X+D)^{\mathrm{wt}(w)}.$$

The function $F_{\mathcal{C}}^{(u)}(Z,0,D)$ is the sum of the terms corresponding to elements of $\mathcal{C}$ covered by $u$ so the property (iii) holds after renaming the variables.

The linear mapping $w \to w * u$ carries $\mathcal{C}$ onto $\mathcal{C} * u$ with kernel $\mathcal{C} \cap \mathcal{C} * \bar{u}$. In the sum of terms $Z^{n-\mathrm{wt}(w*u)}(Z+D)^{\mathrm{wt}(w*u)}$ for the polynomial $F_{\mathcal{C}}^{(u)}(Z,Z,D)$, each element $w * u$ is covered exactly $|\mathcal{C} \cap \mathcal{C} * \bar{u}|$ times as $w$ ranges over $\mathcal{C}$. The number of elements in this intersection is its weight enumerator evaluated at $A = B = 1$, namely, $F_{\mathcal{C}}^{(\bar{u})}(1,0,1)$ by property (iii). This number equals $F_{\mathcal{C}}^{(u)}(1,1,-1)$ by property (i) and so (iv) follows. □

**Transform of the overlap polynomials.** We regard $F_{\mathcal{C}}^{(u)}(Z,X,D)$ as a function of $u \in \mathbf{F}^n$ and compute the Hadamard transform of this function.

PROPOSITION 1.2. *The Hadamard transform of $F^{(u)}(Z,X,D)$ is given by*

$$\widehat{F}_{\mathcal{C}}^{(v)}(Z,X,D) = \sum_{\substack{w\in\mathcal{C}\\ w\geq v}} (2Z)^{n-\mathrm{wt}(w)}(2X+D)^{\mathrm{wt}(w)-\mathrm{wt}(v)}(-D)^{\mathrm{wt}(w)}.$$

*Proof.* From the definition of the transform and the overlap enumerator we have the equations

$$\begin{aligned}\widehat{F}_{\mathcal{C}}^{(v)}(Z,X,D) &= \sum_{u\in\mathbf{F}^n}(-1)^{u\cdot v}\sum_{w\in\mathcal{C}} Z^{n-\mathrm{wt}(w)}X^{\mathrm{wt}(w)-\mathrm{wt}(w*u)}(X+D)^{\mathrm{wt}(w*u)}\\ &= \sum_{w\in\mathcal{C}} Z^{n-\mathrm{wt}(w)}X^{\mathrm{wt}(w)}\sum_{a\leq w}(-1)^{a\cdot v*w}\left(\frac{X+D}{X}\right)^{\mathrm{wt}(a)}\sum_{b\leq\bar{w}}(-1)^{b\cdot v*\bar{w}},\end{aligned}$$

where for a given $w \in \mathcal{C}$ we have written $u = a + b$ with $a \leq w$ and $b \leq \bar{w}$. The inner sums are evaluated as follows:

$$\sum_{b\leq\bar{w}}(-1)^{b\cdot v*\bar{w}} = \begin{cases} 2^{\mathrm{wt}(\bar{w})} & \text{if } v \leq w,\\ 0 & \text{otherwise;}\end{cases}$$

and for fixed $w \in \mathcal{C}$ and $v \leq w$, by the MacWilliams formula

$$\sum_{a\leq w}(-1)^{a\cdot v*w}\left(\frac{X+D}{X}\right)^{\mathrm{wt}(v*w)} = \left(1+\frac{X+D}{X}\right)^{\mathrm{wt}(w)-\mathrm{wt}(v*w)}\left(1-\frac{X+D}{X}\right)^{\mathrm{wt}(v*w)}.$$

The result now follows by substitutions and simplification. □

Using the transform property and exchange of summation gives us the next corollary.

COROLLARY 1.3. *For any linear codes $\mathcal{C}$, $\mathcal{D} \subseteq \mathbf{F}^n$:*

(i)

$$|\mathcal{D}| \sum_{v\in\mathcal{D}^\perp} F_{\mathcal{C}}^{(v)}(Z,X,D) = \sum_{u\in\mathcal{D}} \sum_{\substack{w\in\mathcal{C}\\ w\geq u}} (2Z)^{n-\mathrm{wt}(w)}(2X+D)^{\mathrm{wt}(w)-\mathrm{wt}(u)}(-D)^{\mathrm{wt}(u)},$$

(ii)

$$\sum_{v\in\mathcal{D}^\perp} F_{\mathcal{C}}^{(v)}(Z,X,D) = |\mathcal{D}^\perp| \sum_{w\in\mathcal{C}} Z^{n-\mathrm{wt}(w)}2^{-\mathrm{wt}(w)} \sum_{\substack{v\in\mathcal{D}\\ v\leq w}} (2X+D)^{\mathrm{wt}(w)-\mathrm{wt}(v)}(-D)^{\mathrm{wt}(v)}.$$

A useful formula is obtained by applying this computation to the code consisting of all elements with support in a given set of coordinates.

COROLLARY 1.4. *For any vector $u \in \mathbf{F}^n$,*

$$\sum_{v\leq u} F_{\mathcal{C}}^{(v)}(Z,X,D) = 2^{\mathrm{wt}(u)} F_{\mathcal{C}}^{(u)}(Z,X,D/2).$$

*Proof.* Let $\mathcal{D} = \bar{u} * \mathbf{F}^n$ so that $\mathcal{D}^\perp = u * \mathbf{F}^n$. Apply Corollary 1.3 (ii) to get

$$\frac{1}{2^{\mathrm{wt}(u)}} \sum_{v\leq u} F_{\mathcal{C}}^{(v)}(Z,X,D) = \sum_{w\in\mathcal{C}} Z^{n-\mathrm{wt}(w)}2^{-\mathrm{wt}(w)} \sum_{v\leq w*\bar{u}} (2X+D)^{\mathrm{wt}(w)-\mathrm{wt}(v)}(-D)^{\mathrm{wt}(v)}$$
$$= \sum_{w\in\mathcal{C}} Z^{n-\mathrm{wt}(w)}2^{-\mathrm{wt}(w)}(2X+D)^{\mathrm{wt}(w)-\mathrm{wt}(w*\bar{u})} \sum_{v\leq w*\bar{u}} (2X+D)^{\mathrm{wt}(w*\bar{u})-\mathrm{wt}(v)}(-D)^{\mathrm{wt}(v)}$$
$$= \sum_{w\in\mathcal{C}} Z^{n-\mathrm{wt}(w)}2^{-\mathrm{wt}(w)}(2X+D)^{\mathrm{wt}(w)-\mathrm{wt}(w*\bar{u})}(2X+D-D)^{\mathrm{wt}(w*\bar{u})}$$
$$= \sum_{w\in\mathcal{C}} Z^{n-\mathrm{wt}(w)}X^{\mathrm{wt}(w)-\mathrm{wt}(w*u)}(X+D/2)^{\mathrm{wt}(w*u)}$$
$$= F_{\mathcal{C}}^{(u)}(Z,X,D/2).$$

The third equality holds by the binomial theorem. □

This relation will allow us to express an overlap weight enumerator corresponding to $u$ in terms of overlap weight enumerators corresponding to vectors of lower weight.

DEFINITION. *For a linear code $\mathcal{C}$ and a vector $u \in \mathbf{F}^n$, the polynomial*

$$g_{\mathcal{C}}^{(u)}(Z,X) = \sum_{\substack{w\in\mathcal{C}\\ w\geq u}} Z^{n-\mathrm{wt}(w)}X^{\mathrm{wt}(w)-\mathrm{wt}(u)}$$

*is called the (homogeneous) covering polynomial of $u$ with respect to $\mathcal{C}$.*

The next step is to express the overlap polynomials in terms of the covering polynomials.

THEOREM 1.5. *For any linear code $\mathcal{C} \subset \mathbf{F}^n$ and vector $u \in \mathbf{F}^n$ we have*

$$F_{\mathcal{C}}^{(u)}(Z,X,D) = \sum_{v\leq u} g_{\mathcal{C}}^{(v)}(Z,X)\,D^{\mathrm{wt}(v)}.$$

Our proof requires the following idea.

Let $\mathcal{P}_r$ denote the $(r+1)$-dimensional vector space of polynomials $h(D)$ of degree at most $r$. The coefficient field is not particularly relevant but may be taken as the field of rational functions in $X$ and $Z$ with rational coefficients.

Proposition 1.6. *Let $L_r$ be the linear map defined on $\mathcal{P}_r$ by $L_r(h(D)) = 2^r\, h(D/2)$. Then $L_r$ has eigenvectors $D^i$, $i = 0, 1, \ldots, r$ and corresponding eigenvalues $2^{r-i}$. Moreover, $L_r - I$ is a linear transformation mapping $\mathcal{P}_r$ onto $\mathcal{P}_{r-1}$ having the same eigenvectors as $L_r$ and corresponding eigenvalues $2^{r-i} - 1$. The null space of $L_r - I$ is the one-dimensional subspace generated by $D^r$.*

The proof is carried out by direct calculation and is left to the reader.

The reason for introducing the operator $L_r$ is the equation

$$\begin{aligned}(L_{\mathrm{wt}(u)} - I)\big(F_{\mathcal{C}}^{(u)}(Z,X,D)\big) &= 2^{\mathrm{wt}(u)} F_{\mathcal{C}}^{(u)}(Z,X,D/2) - F_{\mathcal{C}}^{(u)}(Z,X,D) \\ &= \sum_{v<u} F_{\mathcal{C}}^{(v)}(Z,X,D),\end{aligned} \tag{1.1}$$

which is just a restatement of Corollary 1.4.

We apply this to prove Theorem 1.5. The proof is by induction on $\mathrm{wt}(u)$. Suppose $u = 0$. Then

$$F_{\mathcal{C}}^{(0)}(Z,X,D) = W_{\mathcal{C}}(Z,X) = g_{\mathcal{C}}^{(0)}(Z,X)D^0,$$

so the statement holds.

Assume the result for all vectors of weight less than $\mathrm{wt}(u)$. There exist some homogeneous polynomials $f_j^{(u)}(Z,X)$ such that

$$F_{\mathcal{C}}^{(u)}(Z,X,D) = \sum_{j=0}^{\mathrm{wt}(u)} f_j^{(u)}(Z,X)D^j.$$

From (1.1) and induction we have

$$\begin{aligned}(L_{\mathrm{wt}(u)} - I)\big(F_{\mathcal{C}}^{(u)}(Z,X,D)\big) &= \sum_{j=0}^{\mathrm{wt}(u)} (2^{\mathrm{wt}(u)-j} - 1) f_j^{(u)}(Z,X)D^j \\ &= \sum_{v<u} F_{\mathcal{C}}^{(v)}(Z,X,D) \\ &= \sum_{v<u} \sum_{w\le v} g_{\mathcal{C}}^{(w)}(Z,X)D^{\mathrm{wt}(w)} \\ &= \sum_{x<u} (2^{\mathrm{wt}(u)-\mathrm{wt}(x)} - 1) g_{\mathcal{C}}^{(x)}(Z,X)D^{\mathrm{wt}(x)}.\end{aligned}$$

It follows that, for $j < \mathrm{wt}(u)$,

$$f_j^{(u)}(Z,X)D^j = \sum_{\substack{\mathrm{wt}(x)=j \\ x<u}} g_{\mathcal{C}}^{(x)}(Z,X)D^{\mathrm{wt}(x)},$$

and so the theorem is proved.

**Some average values.** Here we compute the average value of the covering and overlap polynomials. Let

$$0 = d_0 < d_1 < \cdots < d_s$$

be the weights of elements of $\mathcal{C}$, so that $d_s \leq n$, and let $A_{d_i}$ be the number of elements of weight $d_i$ in $\mathcal{C}$.

PROPOSITION 1.7. *For any linear code $\mathcal{C}$ and nonnegative integer $k$,*

(i)

$$\sum_{\mathrm{wt}(u)=k} g_{\mathcal{C}}^{(u)}(Z,X) = \sum_{j=0}^{s} \binom{d_j}{k} A_{d_j} Z^{n-d_j} X^{d_j-k},$$

(ii)

$$\sum_{\mathrm{wt}(u)=k} F_{\mathcal{C}}^{(u)}(Z,X,D) = \sum_{i=0}^{k}\sum_{j=0}^{s} \binom{n-i}{k-i}\binom{d_j}{i} A_{d_j} Z^{n-d_j} X^{d_j-i} D^i.$$

*Proof.* Starting with the definition of the covering polynomials, we obtain

$$\begin{aligned}
\sum_{\mathrm{wt}(u)=k} g_{\mathcal{C}}^{(u)}(Z,X) &= \sum_{\mathrm{wt}(u)=k} \sum_{\substack{w\in\mathcal{C}\\ w\geq u}} Z^{n-\mathrm{wt}(w)} X^{\mathrm{wt}(w)-k}\\
&= \sum_{w\in\mathcal{C}} \sum_{\substack{u\leq w\\ \mathrm{wt}(u)=k}} Z^{n-\mathrm{wt}(w)} X^{\mathrm{wt}(w)-k}\\
&= \sum_{w\in\mathcal{C}} \binom{\mathrm{wt}(w)}{k} Z^{n-\mathrm{wt}(w)} X^{\mathrm{wt}(w)-k},
\end{aligned}$$

and the statement (i) follows immediately.

For the overlap enumerators, we apply Theorem 1.5 and part (i) of this proposition to get

$$\begin{aligned}
\sum_{\mathrm{wt}(u)=k} F_{\mathcal{C}}^{(u)}(Z,X,D) &= \sum_{\mathrm{wt}(u)=k} \sum_{v\leq u} g_{\mathcal{C}}^{(v)}(Z,X)\, D^{\mathrm{wt}(v)}\\
&= \sum_{i=0}^{k} \sum_{\mathrm{wt}(v)=i} \binom{n-i}{k-i} g_{\mathcal{C}}^{(v)}(Z,X)\, D^i\\
&= \sum_{i=0}^{k}\sum_{j=0}^{s} \binom{n-i}{k-i}\binom{d_j}{i} A_{d_j} Z^{n-d_j} X^{d_j-i} D^i. \qquad \square
\end{aligned}$$

In the next section we will be concerned with designs held by the words of each weight in $\mathcal{C}$. If the weight of $u$ is at most the design strength of the code, then the covering polynomial $g_{\mathcal{C}}^{(u)}(Z,X)$ is independent of $u$. The average is then the common value. We may summarize this as follows.

COROLLARY 1.8. *Suppose all the covering polynomials $g_{\mathcal{C}}^{(u)}(Z,X)$ are the same for vectors $u$ of weight $r$. Then for any vector $u$ of weight $r$*

(i)

$$g_{\mathcal{C}}^{(u)}(Z,X) = \binom{n}{r}^{-1} \sum_{j=0}^{s} \binom{d_j}{k} A_{d_j} Z^{n-d_j} X^{d_j-k}$$

*and*

(ii)

$$F_{\mathcal{C}}^{(u)}(Z,X,D) = \binom{n}{r}^{-1} \sum_{i=0}^{k}\sum_{j=0}^{s} \binom{n-i}{k-i}\binom{d_j}{i} A_{d_j} Z^{n-d_j} X^{d_j-i} D^i.$$

**Duality.** Our next step is to give a relation between $F_{\mathcal{C}}^{(u)}(Z,X,D)$ and $F_{\mathcal{C}^\perp}^{(u)}(Z,X,D)$ similar to the MacWilliams identity and thereby obtain a functional equation for this function in the case $\mathcal{C} = \mathcal{C}^\perp$. The duality property of $F_{\mathcal{C}}^{(u)}(Z,X,D)$ is obtained by the usual method using the Hadamard transform. It can also be derived by making a change of variable in the duality theorem proved by Ozeki [Oz] for Jacobi polynomials.

Let $u \in \mathbf{F}^n$ be a fixed vector; let $h(w)$ be defined for $w \in \mathbf{F}^n$ by

$$h(w) = Z^{n-\mathrm{wt}(w)} X^{\mathrm{wt}(w)-\mathrm{wt}(w*u)} (X+D)^{\mathrm{wt}(w*u)}.$$

Proposition 1.9. *The Hadamard transform of $h(w)$ is given by*

$$\hat{h}(v) = (Z+X)^{\mathrm{wt}(u)}(Z+X+D)^{n-\mathrm{wt}(u)} \left(\frac{Z-X-D}{Z+X+D}\right)^{\mathrm{wt}(v)-\mathrm{wt}(v*u)} \left(\frac{Z-X}{Z+X}\right)^{\mathrm{wt}(w*u)}.$$

*Proof.* Starting with the definition of the transform, we obtain

$$\begin{aligned}\hat{h}(v) &= \sum_{y\in \mathbf{F}^n} (-1)^{v\cdot y} Z^n \left(\frac{X}{Z}\right)^{\mathrm{wt}(y)} \left(\frac{X+D}{X}\right)^{\mathrm{wt}(u*y)} \\ &= Z^n \sum_{a\le u}\sum_{b\le \bar{u}} (-1)^{a\cdot v*u}(-1)^{b\cdot v*\bar{u}} \left(\frac{X}{Z}\right)^{\mathrm{wt}(a)+\mathrm{wt}(b)} \left(\frac{X+D}{X}\right)^{\mathrm{wt}(a)} \\ &= Z^n \left(1+\frac{X}{Z}\right)^{\mathrm{wt}(u)-\mathrm{wt}(u*v)} \left(1-\frac{X}{Z}\right)^{\mathrm{wt}(u*v)} \left(1+\frac{X+D}{Z}\right)^{n-\mathrm{wt}(u)-\mathrm{wt}(v)+\mathrm{wt}(u*v)} \\ &\qquad \times \left(1-\frac{X+D}{Z}\right)^{\mathrm{wt}(v)-\mathrm{wt}(u*v)}.\end{aligned}$$

We wrote $y = a + b$ with $a \le u$ and $b \le \bar{u}$. The statement now follows. ☐

Next we obtain the duality formula by applying the Transform property and the homogeniety of $F^{(u)}$.

Theorem 1.10. *For any binary linear code $\mathcal{C} \subseteq \mathbf{F}^n$ and any vector $u \in \mathbf{F}^n$ the following relation holds between the homogeneous overlap enumerators of $\mathcal{C}$ and $\mathcal{C}^\perp$:*

$$F_{\mathcal{C}^\perp}^{(u)}(Z,X,D) = \frac{1}{|\mathcal{C}|}\left(\frac{Z+X+D}{Z+X}\right)^{\mathrm{wt}(u)} F_{\mathcal{C}}^{(u)}\left(Z+X, Z-X, \frac{-2ZD}{Z+X+D}\right).$$

This duality relation provides a similar duality relation for the covering polynomials.

Theorem 1.11. *For any binary linear code $\mathcal{C} \subseteq \mathbf{F}^n$ and any vector $u \in \mathbf{F}^n$ the following relation holds between the homogeneous covering polynomials of $\mathcal{C}$ and $\mathcal{C}^\perp$:*

$$|\mathcal{C}|(Z+X)^{\mathrm{wt}(u)} g_{\mathcal{C}^\perp}^{(u)}(Z,X) = \sum_{v\le u} g_{\mathcal{C}}^{(v)}(Z+X, Z-X)(-2Z)^{\mathrm{wt}(v)}.$$

*Proof.* Start with the duality equation in Theorem 1.10 and express each side in terms of the covering polynomials using Theorem 1.5 to get

$$\sum_{v\le u} g^{(u)}_{\mathcal{C}^\perp}(Z,X)D^{\mathrm{wt}(v)} = \frac{1}{|\mathcal{C}|}\Big(\frac{Z+X+D}{Z+X}\Big)^{\mathrm{wt}(u)}\sum_{v\le u} g^{(u)}_{\mathcal{C}}(Z+X,Z-X)\left(\frac{-2ZD}{Z+X+D}\right)^{\mathrm{wt}(v)}.$$

Multiply both sides by $(Z+X)^{\mathrm{wt}(u)}$ to get an equality of polynomials of degree $\mathrm{wt}(u)$ in $D$. The conclusion of the theorem is obtained by equating the coefficients of $D^{\mathrm{wt}(u)}$. □

For the record we also equate the coefficients of the other powers of $D$.

Corollary 1.12. *For each nonnegative integer $k \le \mathrm{wt}(u)$,*

$$|\mathcal{C}|(Z+X)^k \sum_{\substack{v\le u\\ \mathrm{wt}(v)=k}} g^{(v)}_{\mathcal{C}^\perp}(Z,X) = \sum_{\substack{v\le u\\ \mathrm{wt}(v)\le k}} \binom{\mathrm{wt}(u)-\mathrm{wt}(v)}{k-\mathrm{wt}(v)} g^{(v)}_{\mathcal{C}}(Z+X,Z-X)(-2Z)^{\mathrm{wt}(v)}.$$

These relations involve the MacWilliams transformation which we formalize as follows. Let $\tau$ be the function defined by

$$\tau(Z)=\frac{Z+X}{\sqrt{2}},\qquad \tau(X)=\frac{Z-X}{\sqrt{2}},$$

and on any polynomial $h(Z,X)$,

$$\tau(h(Z,X)) = h(\tau(Z),\tau(X)).$$

Note that $\tau$ is an automorphism of order 2 of the ring of real polynomials in $Z$ and $X$.

Since $g^{(u)}_{\mathcal{C}}(Z,X)$ is homogeneous of degree $n-\mathrm{wt}(u)$, we may rewrite the conclusion of Theorem 1.11 as

$$\tau\left(g^{(u)}_{\mathcal{C}^\perp}(Z,X)\right) = \frac{2^{n/2}}{|\mathcal{C}|Z^{\mathrm{wt}(u)}}\sum_{v\le u}(-1)^{\mathrm{wt}(v)}(Z+X)^{\mathrm{wt}(v)}g^{(v)}_{\mathcal{C}}(Z,X). \tag{1.2}$$

**2. Designs.** We refer to the set of all elements of a given weight in a code $\mathcal{C}$ as a *weight class* of $\mathcal{C}$. A weight class is *trivial* if its elements have weight either 0 or $n$ (the length of the code). Other weight classes are *nontrivial*. A weight class $\mathcal{W}$ of $\mathcal{C}$ holds a *t-design* if the number of elements in $\mathcal{W}$ that cover a vector $u$ of weight $t$ is independent of the particular vector of weight $t$. The *design strength* $\delta(\mathcal{C})$ of $\mathcal{C}$ is the largest integer $\delta$ such that every weight class of $\mathcal{C}$ holds a $\delta$-design.

The weights of elements of $\mathcal{C}$ are $0=d_0<d_1<\cdots<d_s$; let $\mathcal{C}^\perp$ have the weights $0=w_0<w_1<\cdots<w_{s'}$ so $w_{s'}\le n$. For a vector $u\in \mathbf{F}^n$, let $b_{\mathcal{C}}[i,u]$ be the number of elements in $\mathcal{C}$ of weight $d_i$ that cover $u$. Thus $b_{\mathcal{C}}[i,u]\ge 0$ and

$$g^{(u)}_{\mathcal{C}}(Z,X) = \sum_{i=0}^{s} b_{\mathcal{C}}[i,u]Z^{n-d_i}X^{d_i-\mathrm{wt}(u)}. \tag{2.1}$$

If $u\ne 0$, the sum begins at $i=1$ because $b_{\mathcal{C}}[0,u]=0$. If $\mathbf{1}\in\mathcal{C}$, then $d_s=n$ and $b_{\mathcal{C}}[s,u]=1$ for every vector $u$. If the elements of weight $d_i$ in $\mathcal{C}$ hold a $t$-design, then

$b_{\mathcal{C}}[i,u]$ has the same value for every element $u$ of weight $t$. Thus the coefficient of $Z^{n-d_i}X^{d_i-t}$ in

$$g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X) \tag{2.2}$$

is 0 for all vectors $u$ and $v$ of weight $t$ and conversely. We summarize as follows.

PROPOSITION 2.1. (i) *The elements of weight $d_i$ in $\mathcal{C}$ hold a $t$-design if and only if the coefficient of $Z^{n-d_i}X^{d_i-t}$ in* (2.2) *is* 0 *for all vectors $u$ and $v$ having weight $t$.*

(ii) *The design strength $\delta(\mathcal{C})$ is the largest integer $\delta$ such that the difference of covering polynomials in* (2.2) *equals* 0 *for all vectors $u, v \in \mathbf{F}^n$ with* $\mathrm{wt}(u) = \mathrm{wt}(v) \le \delta$.

*Remark.* $\delta(\mathcal{C}) = \delta(\mathcal{C}^\perp)$.

Theorems 1.5 and 1.10 can be used to quickly prove the (well-known) fact that $\delta(\mathcal{C}) = \delta(\mathcal{C}^\perp)$. Since the covering polynomial $g_{\mathcal{C}}^{(u)}(Z,X)$ depends only on the weight of vector $u$ when $\mathrm{wt}(u) \le \delta(\mathcal{C})$, the same is true of the overlap polynomials $F_{\mathcal{C}}^{(u)}(Z,X,D)$ by Theorem 1.5. Then apply Theorem 1.10 to get an expression for $F_{\mathcal{C}^\perp}^{(u)}(Z,X,D)$ which depends only on the weight of $u$. Since $g_{\mathcal{C}^\perp}^{(u)}(Z,X)$ is the coefficient of the highest power of $D$ in the overlap polynomial, it follows that the covering polynomials for $\mathcal{C}^\perp$ depend only on the weight of $u$. Hence $\delta(\mathcal{C}) \le \delta(\mathcal{C}^\perp)$. Repeat the argument starting with $\mathcal{C}^\perp$ in place of $\mathcal{C}$ to get the reverse inequality.

We will show below that the difference (2.2) has a factorization, in the case $\mathrm{wt}(u) = \mathrm{wt}(v) = 1 + \delta(\mathcal{C})$, from which conclusions can be drawn about the size of $\delta(\mathcal{C})$.

The most important theorem relating codes and designs is the following.

THEOREM (Assmus and Mattson [A-M]). *Let $\mathcal{C}$ be a binary linear code with nonzero weights $d_1 < d_2 < \cdots < d_s$ and let its dual $\mathcal{C}^\perp$ have nonzero weights $w_1 < w_2 < \cdots < w_{s'}$. Let $t$ be an integer with $0 < t < d_1$ such that the number of weights $w_i$ that are less than or equal to $n - t$ is at most $d_1 - t$. Then the code words of any given weight in $\mathcal{C}$ and in $\mathcal{C}^\perp$ form a $t$-design.*

If $t$ is a number satisfying the hypothesis of the Assmus–Mattson theorem, then the conclusion is equivalent to the inequality $t \le \delta(\mathcal{C})$. We will present a proof of the Assmus–Mattson theorem in a slightly stronger form. We begin with some properties of the difference of covering polynomials.

**Factorization of differences of covering polynomials.** Throughout the remainder of this section let $r = 1 + \delta(\mathcal{C})$.

LEMMA 2.2. *For any $u, v \in \mathbf{F}^n$ of weight $r = 1 + \delta(\mathcal{C})$, the covering polynomials satisfy*

$$g_{\mathcal{C}^\perp}^{(u)}(Z,X) - g_{\mathcal{C}^\perp}^{(v)}(Z,X) = \frac{(-2Z)^r}{|\mathcal{C}|(Z+X)^r}\left(g_{\mathcal{C}}^{(u)}(Z+X, Z-X) - g_{\mathcal{C}}^{(v)}(Z+X, Z-X)\right).$$

*Proof.* For $0 \le k < r$, all covering polynomials corresponding to vectors of weight $k$ are equal, and so it follows from Theorem 1.5 applied to both $\mathcal{C}$ and $\mathcal{C}^\perp$ that

$$F_{\mathcal{C}}^{(u)}(Z,X,D) - F_{\mathcal{C}}^{(v)}(Z,X,D) = \left(g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X)\right) D^r,$$
$$F_{\mathcal{C}^\perp}^{(u)}(Z,X,D) - F_{\mathcal{C}^\perp}^{(v)}(Z,X,D) = \left(g_{\mathcal{C}^\perp}^{(u)}(Z,X) - g_{\mathcal{C}^\perp}^{(v)}(Z,X)\right) D^r.$$

Next apply Theorem 1.10 to get

$$\begin{aligned}&F^{(u)}_{\mathcal{C}^\perp}(Z,X,D)) - F^{(v)}_{\mathcal{C}^\perp}(Z,X,D)\\ &\qquad = \frac{1}{|\mathcal{C}|}\left(\frac{Z+X+D}{Z+X}\right)^r \left(F^{(u)}_{\mathcal{C}}(Z+X,Z-X,D') - F^{(v)}_{\mathcal{C}}(Z+X,Z-X,D')\right),\end{aligned}$$

where $D' = -2ZD/(Z+X+D)$. Express each side in terms of the $g$-polynomials from the relations just above to obtain the conclusion.

The conclusion of the previous lemma may be written using the MacWilliams transformation $\tau$ as

(2.3)
$$g^{(u)}_{\mathcal{C}^\perp}(Z,X) - g^{(v)}_{\mathcal{C}^\perp}(Z,X) = \frac{1}{|\mathcal{C}|}\left(\frac{-2Z}{Z+X}\right)^r (\sqrt{2})^{n-r}\,\tau\Big(g^{(u)}_{\mathcal{C}}(Z,X) - g^{(v)}_{\mathcal{C}}(Z,X)\Big).$$

We elaborate on this idea to get some explicit factors of the difference of covering polynomials.

PROPOSITION 2.3. *Let $u$ and $v$ be vectors of weight $r = 1+\delta(\mathcal{C})$. Then*

$$\begin{aligned}g^{(u)}_{\mathcal{C}}(Z,X) - g^{(v)}_{\mathcal{C}}(Z,X) &= Z^{n-d_m}(Z+X)^{n-w_{m'}-r}X^{d_q-r}(Z-X)^{w_{q'}-r}\,P^{(u,v)}_{\mathcal{C}}(Z,X),\\ g^{(u)}_{\mathcal{C}^\perp}(Z,X) - g^{(v)}_{\mathcal{C}^\perp}(Z,X) &= Z^{n-w_{m'}}(Z+X)^{n-d_m-r}X^{w_{q'}-r}(Z-X)^{d_q-r}\,P^{(u,v)}_{\mathcal{C}^\perp}(Z,X),\end{aligned}$$

*where $P^{(u,v)}_{\mathcal{C}}(Z,X)$ and $P^{(u,v)}_{\mathcal{C}^\perp}(Z,X)$ are either both 0 or both nonzero homogeneous polynomials not divisible by $Z$, $Z+X$, $X$, or $Z-X$ and satisfying*

$$P^{(u,v)}_{\mathcal{C}^\perp}(Z,X) = \lambda\,\tau\Big(P^{(u,v)}_{\mathcal{C}}(Z,X)\Big), \qquad \lambda = \frac{(-1)^r}{|\mathcal{C}|}\left(\sqrt{2}\right)^{n+d_m-d_q-w_{m'}+w_{q'}}.$$

*When $g^{(u)}_{\mathcal{C}}(Z,X) - g^{(v)}_{\mathcal{C}}(Z,X) \neq 0$, the exponents are determined as follows:*

(i) *$q$ is the smallest index such that $b_{\mathcal{C}}[q,u] - b_{\mathcal{C}}[q,v] \neq 0$ and $q'$ is the smallest index such that $b_{\mathcal{C}^\perp}[q',u] - b_{\mathcal{C}^\perp}[q',v] \neq 0$;*

(ii) *$m$ is the largest index such that $b_{\mathcal{C}}[m,u] - b_{\mathcal{C}}[m,v] \neq 0$ and $m'$ is the largest index such that $b_{\mathcal{C}^\perp}[m',u] - b_{\mathcal{C}^\perp}[m',v] \neq 0$;*

(iii) *For any vectors $u$ and $v$ of weight $r$, the indices satisfy*
  (a) *$1 \le q \le m \le s$ with $m < s$ if the all ones vector is in $\mathcal{C}$.*
  (b) *$1 \le q' \le m' \le s'$ with $m' < s'$ if the all ones vector is in $\mathcal{C}^\perp$.*

*Proof.* Begin by expressing the difference as

$$A = g^{(u)}_{\mathcal{C}}(Z,X) - g^{(v)}_{\mathcal{C}}(Z,X) = Z^a(Z+X)^b X^c (Z-X)^e\,P^{(u,v)}_{\mathcal{C}}(Z,X)$$

for some nonnegative integers $a$, $b$, $c$, $e$, and some homogeneous polynomial $P$ not divisible by any of the factors $Z, X, Z+X, Z-X$. In the same manner write

$$B = g^{(u)}_{\mathcal{C}^\perp}(Z,X) - g^{(v)}_{\mathcal{C}\perp}(Z,X) = Z^{a'}(Z+X)^{b'}X^{c'}(Z-X)^{e'}\,P^{(u,v)}_{\mathcal{C}\perp}(Z,X).$$

Starting from (2.3), we obtain

$$\begin{aligned}B &= \frac{1}{|\mathcal{C}|}\left(\frac{-2Z}{Z+X}\right)^r(\sqrt{2})^{n-r}\tau(A)\\ &= \frac{(-1)^r(\sqrt{2})^{n+r}}{|\mathcal{C}|}Z^{b+r}(Z+X)^{a-r}X^e(Z-X)^c\,\tau\Big(P^{(u,v)}_{\mathcal{C}}(Z,X)\Big).\end{aligned}$$

The polynomials $P_{\mathcal{C}}$ and $P_{\mathcal{C}^\perp}$ are not divisible by any of the linear factors $Z$, $X$, $Z+X$, $Z-X$, and $\tau$ permutes these factors (up to constant multiples); thus $\tau\,(P_{\mathcal{C}^\perp})$ is not divisible by any of these linear factors. We may draw two conclusions: First,

$$\lambda\,\tau\Big(P_{\mathcal{C}}^{(u,v)}(Z,X)\Big) = P_{\mathcal{C}^\perp}^{(u,v)}(Z,X)$$

for some constant $\lambda$. Using the fact that $P$ is homogeneous of degree $n-r$ the constant is found to be as stated in the proposition.

Second, the exponents of the linear factors must be the same on each side of this equation. Thus

$$a' = b + r, \quad b' = a - r, \quad c' = e, \quad \text{and} \quad e' = c. \tag{2.4}$$

Next appeal to the explicit form of the covering polynomials given in (2.1) to see

$$g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X) = \sum_{i=1}^{s}(b_{\mathcal{C}}[i,u] - b_{\mathcal{C}}[i,v])Z^{n-d_i}X^{d_i-r}.$$

Thus the exact exponent of $Z$ dividing this difference is $a = n - d_m$ and the exact exponent of $X$ is $c = d_q - r$ where $m$ and $q$ as defined in the statements (i) and (ii). In the same way, examine the difference of the covering polynomials for $\mathcal{C}^\perp$ to conclude $a' = n - w_{m'}$ and $c' = w_{q'} - r$. Now combine this information with the relations (2.4) to obtain the exponents as stated in the conclusion of the proposition. Note that when the all ones vector is in $\mathcal{C}$, then $d_s = n$ and $b_{\mathcal{C}}[s,u] = b_{\mathcal{C}}[s,v] = 1$ and so $m < s$. Similarly $m' < s'$ if $\mathbf{1} \in \mathcal{C}^\perp$.

In the presence of some additional hypothesis, the exponents appearing in the factorization may be described more precisely.

COROLLARY 2.4. *Keep the notation of Proposition* 2.3.

(i) *If all weights in $\mathcal{C}$ are divisible by an even integer $k$, then $n - w_{m'} = w_{q'}$, $P_{\mathcal{C}}^{(u,v)}(Z,X)$ is a polynomial in $Z^k$ and $X^k$, and*

$$g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X) = Z^{n-d_m}X^{d_q-r}(Z^k - X^k)^{w_{q'}-r}h(Z,X)$$

*for some polynomial $h(Z,X)$.*

(ii) *If all weights in $\mathcal{C}^\perp$ are divisible by an even integer $k$, then $n - d_m = d_q$, $P_{\mathcal{C}^\perp}^{(u,v)}(Z,X)$ is a polynomial in $Z^k$ and $X^k$, and*

$$g_{\mathcal{C}^\perp}^{(u)}(Z,X) - g_{\mathcal{C}^\perp}^{(v)}(Z,X) = Z^{n-w_{m'}}X^{w_{q'}-r}(Z^k - X^k)^{d_q-r}h^\perp(Z,X)$$

*for some polynomial $h^\perp(Z,X)$.*

*Proof.* If $n = kt - j$ for integers $t$ and $j$ and all weights in $\mathcal{C}$ are divisible by $k$, then $Z^jX^r\,g_{\mathcal{C}}^{(u)}(Z,X)$ is a polynomial in $Z^k$ and $X^k$ (see (2.1)) and so

$$Z^jX^r\left(g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X)\right) = M(Z,X) \tag{2.5}$$

is a homogeneous polynomial that satisfies $M(Z,X) = M(Z,\omega X)$ for any complex $k$th root of unity $\omega$. If $(Z-X)^a$ is the exact power of $Z - X$ dividing $M(Z,X)$, then the substitution of $\omega X$ for $X$ shows that the exact power of $Z - \omega X$ dividing $M(Z,X)$ is also the $a$th power. Hence, $(Z^k - X^k)^a$ is the product of all these factors of $M(Z,X)$ and $a = n - w_{m'} - r = w_{q'} - r$. The result follows. $\square$

**Bounds on $\delta(\mathcal{C})$.** The factorizations in Proposition 2.3 contain enough information to prove the Assmus–Mattson theorem and even a bit more. We first require an elementary lemma.

LEMMA 2.5. *If $f(Z,X) = c_1 Z^{k-i_1}X^{i_1} + \cdots + c_m Z^{k-i_m}X^{i_m}$ is a polynomial equal to a sum of exactly $m$ nonzero terms, $c_j Z^{k-i_j}X^{i_j}$, and if $f(Z,X)$ is divisible by $(Z-\alpha X)^k$ for $\alpha \neq 0$ then $k+1 \leq m$.*

*Proof.* The statement is obvious for $k = 1$ since a single term cannot be divisible by $Z - \alpha X$. For $k > 1$, factor out a suitable power of $X$ to achieve $f(Z,0) \neq 0$, and then take the derivative with respect to $X$. The number of nonzero terms is reduced by one and the power of $Z - \alpha X$ is reduced by one. The inequality follows by induction.

DEFINITION. *For a linear code $\mathcal{C}$ let $\epsilon_{\mathcal{C}} = 1$ if the all one vector $\mathbf{1} \in \mathcal{C}$ and 0 otherwise.*

THEOREM 2.6. (i) *The design strength $\delta(\mathcal{C})$ satisfies the inequalities*

$$w_1 - s + \epsilon_{\mathcal{C}} \leq \delta(\mathcal{C}) \qquad \text{and} \qquad d_1 - s' + \epsilon_{\mathcal{C}^\perp} \leq \delta(\mathcal{C}).$$

(ii) *If $w_1 - s + \epsilon_{\mathcal{C}} = \delta(\mathcal{C})$, then no nontrivial weight class of $\mathcal{C}$ holds a $1+\delta(\mathcal{C})$-design.*

(iii) *If $d_1 - s' + \epsilon_{\mathcal{C}^\perp} = \delta(\mathcal{C})$, then no nontrivial weight class of $\mathcal{C}^\perp$ holds a $1+\delta(\mathcal{C})$-design.*

*Proof.* Since $\delta = \delta(\mathcal{C})$ is the largest integer such that the difference (2.2) of covering polynomials is 0 for all pairs $u$, $v$ of vectors of equal weight $\leq \delta$, there must be some pair of vectors of weight $r = 1 + \delta$ such that

$$\Delta g_{\mathcal{C}} \;\; = g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X) \neq 0.$$

Let $p$ denote the exact number of nonzero terms (of the form $c_j Z^{n-d_j}X^{d_j-r}$) in $\Delta g_{\mathcal{C}}$. With $m$ and $q$ as defined in Proposition 2.3 we see there are at most $m-q+1$ nonzero terms in $\Delta g_{\mathcal{C}}$, so

$$p \leq m - q + 1 \leq s - \epsilon_{\mathcal{C}}.$$

Lemma 2.5 and the factorization in Proposition 2.3 imply the inequality $w_{q'} - r + 1 \leq p$. Since $r = 1 + \delta(\mathcal{C})$ and $w_1 \leq w_{q'}$, we obtain the inequalities $w_{q'} - p \leq \delta(\mathcal{C})$ and

$$w_1 - s + \epsilon_{\mathcal{C}} \leq w_{q'} - s + \epsilon_{\mathcal{C}} \leq w_{q'} - m + q - 1 \leq w_{q'} - p \leq \delta(\mathcal{C}). \tag{2.6}$$

In the analogous manner we obtain

$$d_1 - s' + \epsilon_{\mathcal{C}^\perp} \leq d_q - s + \epsilon_{\mathcal{C}^\perp} \leq d_q - m' + q' - 1 \leq d_q - p' \leq \delta(\mathcal{C}), \tag{2.7}$$

where $p'$ is the number of nonzero terms in the difference of the covering polynomials for $u$ and $v$ relative to $\mathcal{C}^\perp$. We have proved

$$\delta(\mathcal{C}) \geq \max\{w_1 - s + \epsilon_{\mathcal{C}}, d_1 - s' + \epsilon_{\mathcal{C}^\perp}\}.$$

which we refer to as the Assmus–Mattson bounds on $\delta(\mathcal{C})$.

Now suppose equality holds in one case, say, $\delta(\mathcal{C}) = w_1 - s + \epsilon_{\mathcal{C}}$. Then equality holds at each point in (2.6); we must have $q' = 1$ and $p = s - \epsilon_{\mathcal{C}}$; that is, the number of nonzero terms in $\Delta g_{\mathcal{C}}$ is as large as possible, one for every weight $d_j$ excluding $d_s$ if $\mathbf{1} \in \mathcal{C}$. For every index $j$, the coefficient $b_{\mathcal{C}}[j,u] - b_{\mathcal{C}}[j,v]$ of $Z^{n-d_j}X^{d_j-r}$ is nonzero, except $d_s$ when $d_s = n$. By Proposition 2.1(i), no nontrivial weight class in $\mathcal{C}$ can hold a $1+\delta(\mathcal{C})$-design.

Similarly if $d_1 - s' + \epsilon_{\mathcal{C}^\perp} = \delta(\mathcal{C})$ then no nontrivial weight class of $\mathcal{C}^\perp$ holds a $1+\delta(\mathcal{C})$-design.

*Remark.* There is just a bit more available information provided by the proof. For example, if $u$ and $v$ are vectors of weight $1+\delta(\mathcal{C})$ such that the difference (2.1) is nonzero, and if the equality $\delta(\mathcal{C}) = w_1 - s + \epsilon_{\mathcal{C}}$ holds, then $q' = 1$ and $p = m - q + 1 = s - \epsilon_{\mathcal{C}}$. It follows that $q = 1$ and $m = s - \epsilon_{\mathcal{C}}$. We also see from the definitions in Proposition 2.3 that whenever the difference (2.1) is nonzero, then in fact for every nontrivial weight $d_j$, the number of elements of weight $d_j$ in $\mathcal{C}$ that cover $u$ is different from the number of elements of weight $d_j$ that covers $v$. In other words, the same pair $u, v$ can be used to disprove the $1+\delta(\mathcal{C})$-design property in each weight class simultaneously.

We give an example to show that a code may have a weight class holding a $t$-design with $t > \delta(\mathcal{C})$. In view of Theorem 2.6, the Assmus–Mattson bound cannot be an equality.

*Example* 2.7. Here is a code $\mathcal{C}$ with parameters [16, 9, 4] and its dual $\mathcal{C}^\perp$ with parameters [16, 7, 6] with the property that $\delta(\mathcal{C}) = 2$ and the words of weight 8 in $\mathcal{C}$ hold a 3-design; the words of weight 8 in $\mathcal{C}^\perp$ also hold a 3-design. $\mathcal{C}$ is the code constructed from the incidence matrix (taken mod 2) of the 2-(16, 4, 1) design given in [CRC, p. 11]. One generating matrix for the code is

$$G = [I_9 \,|\, A]\,,$$

where $I_9$ is the $9 \times 9$ identity matrix and $A$ is the $9 \times 7$ matrix whose rows are the binary expansions (padded with zeros on the left) of 7, 41, 91, 117, 81, 61, 107, 103, 31. The weight enumerators of $\mathcal{C}$ and $\mathcal{C}^\perp$ are

$$W_{\mathcal{C}}(1,X) = 1 + 20\,X^4 + 160\,X^6 + 150\,X^8 + 160\,X^{10} + 20\,X^{12} + X^{16},$$
$$W_{\mathcal{C}^\perp}(1,X) = 1 + 48\,X^6 + 30\,X^8 + 48\,X^{10} + X^{16}.$$

The Assmus–Mattson bounds give $6 - 6 + 1 = 1 \le \delta(\mathcal{C})$. The code is small enough so that all the covering polynomials $g_{\mathcal{C}}^{(u)}(X) = g_{\mathcal{C}}^{(u)}(1,X)$ for $\mathrm{wt}(u) = 2$ can be computed; they are all equal to

$$X^2 + 20\,X^4 + 35\,X^6 + 60\,X^8 + 11\,X^{10} + X^{14}.$$

Hence the design strength is $\delta(\mathcal{C}) \ge 2$. Next compute the covering polynomials $g_{\mathcal{C}^\perp}^{(u)}(X) = g_{\mathcal{C}^\perp}^{(u)}(1,X)$ for the smaller dual code and discover that there are two such polynomials as $u$ ranges over all elements of weight 3:

$$g_1 = 3X^5 + 12X^7 + X^{13}, \qquad g_2 = 2X^3 + 3X^5 + 10X^7 + X^{13}.$$

Since they are not all equal, the code does not hold 3-designs in all weight classes. However, in all these covering polynomials for elements of weight 3, the coefficient of $X^5$ is 3; thus any weight 3 vector is covered exactly by 3 elements of weight 8 $(= 5 + \mathrm{wt}(u))$ in $\mathcal{C}^\perp$. Thus the elements of weight 8 in $\mathcal{C}^\perp$ hold a 3-design. A similar direct check verifies that the same is true of the elements of weight 8 in $\mathcal{C}$.

There is another way to see that the elements of weight 8 in $\mathcal{C}^\perp$ hold a 3-design. The 30 elements of weight 8 span a 5-dimensional code $D = [16, 5, 8]$ with weight enumerator $W_D(1,X) = 1 + 30X^8 + X^{16}$. Its dual has weight enumerator

$$W_{D^\perp}(1,X) = 1 + 140\,X^4 + 448\,X^6 + 870\,X^8 + 448\,X^{10} + 140\,X^{12} + X^{16}.$$

Thus the dual of $D$ has $s' = 6$ so the Assmus–Mattson bound yields $8 - 6 + 1 = 3 \leq \delta(D)$. Hence the elements of weight 8 in $D$ hold a 3-design and these are the elements of weight 8 in $\mathcal{C}^\perp$.

Theorem 2.6 implies restrictions on the minimum weights of $\mathcal{C}$ and $\mathcal{C}^\perp$ in terms of the number of weights. We use it to obtain a solution of the Research Problem (6.1) in [M-S, p. 166] which has previously been solved by I. I. Dumer and also by C. Roos, according to [M-S, Preface to 3rd ed., p. xii].

COROLLARY 2.8. *Let $\mathcal{C}$ be binary linear code such that neither $\mathcal{C}$ nor $\mathcal{C}^\perp$ contain the code consisting of all even-weight vectors in $\mathbf{F}^n$. Then*

$$w_1 < d_1 + s - \epsilon_{\mathcal{C}}, \quad \textit{and} \quad d_1 < w_1 + s' - \epsilon_{\mathcal{C}^\perp}.$$

*Proof.* First we argue that the design strength $\delta(\mathcal{C})$ cannot be as large as the minimum weight $d_1$ of a nonzero word in $\mathcal{C}$ so that $\delta(\mathcal{C}) < d_1$. Suppose $\delta(\mathcal{C}) \geq d_1$. Then, since there is a vector of weight $d_1$ in $\mathcal{C}$, every vector $u \in \mathbf{F}^n$ of weight $d_1$ is covered by the same number of elements in $\mathcal{C}$ having weight $d_1$, namely one. It follows that every vector of weight $d_1$ is in $\mathcal{C}$. If $d_1 = n$, then $\mathcal{C} = \{0, \mathbf{1}\}$ and $\mathcal{C}^\perp$ is the code consisting of all even-weight vectors in $\mathbf{F}^n$. If $1 \leq d_1 < n$ then $\mathcal{C}$ contains the set of all even-weight vectors because the set of all vectors of any given weight (excluding 0 and $n$) generates a code containing even-weight subcode. Thus $\mathcal{C}$ or $\mathcal{C}^\perp$ is either $\mathbf{F}^n$ or the even-weight subcode, and these cases have been excluded.

Now since $\delta < d_1$, we apply Theorem 2.6 to conclude

$$w_1 - s + \epsilon_{\mathcal{C}} \leq \delta(\mathcal{C}) < d_1.$$

Using $\mathcal{C}^\perp$ in place of $\mathcal{C}$ yields $d_1 - s' + \epsilon_{\mathcal{C}^\perp} \leq \delta(\mathcal{C}) < w_1$. □

Now we consider the factorization from Proposition 2.3 in the case of self-dual codes.

THEOREM 2.9. *Assume $\mathcal{C} = \mathcal{C}^\perp$ is a self-dual code and $k$ is the greatest common divisor of the weights of $\mathcal{C}$ (so then $k = 2$ or 4). Define $q$ as in Proposition 2.3.*

(i) *If $u$ and $v$ are vectors of weight $r = 1 + \delta(\mathcal{C})$*

$$g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X) = Z^{d_q} X^{d_q - r}(Z^k - X^k)^{d_q - r} H(Z,X)$$

*for a homogeneous polynomial $H(Z,X)$ relatively prime to $ZX(Z^k - X^k)$.*

(ii) *Moreover $H(Z,X)$ is a polynomial in $Z^k$ and $X^k$ and*

$$\tau\left(H(Z,X)\right) = (-1)^r H(Z,X).$$

(iii) *If $k = 2$, then*

$$H(Z,X) = \gamma(Z^2 + X^2)^a (Z^4 - 6Z^2X^2 + X^4)^b \prod_{\alpha} h_\alpha(Z,X)$$

*where either $h_\alpha(Z,X)$ is a constant or*

$$h_\alpha(Z,X) = (\alpha^2(\alpha^2-1)^2(X^8+Z^8) - (\alpha^8 + 14\alpha^4 + 1)(Z^6X^2 + Z^2X^6)$$
$$+ 2(\alpha^8 + 7\alpha^6 + 7\alpha^2 + 1)Z^4X^4,$$

$$\alpha \quad \textit{subject to} \quad \alpha^2(\alpha^2-1)(\alpha^4 + 6\alpha^2 + 1) \neq 0.$$

*Here $\gamma$ is a constant and $a$ and $b$ are nonnegative integers with $b \equiv 1 + \delta(\mathcal{C})$ (mod 2).*

(iv) *If $k = 4$, then*

$$H(Z,X) = \gamma(Z^8 + 14Z^4X^4 + X^8)^a(Z^{12} - 33Z^8X^4 - 33Z^4X^8 + X^{12})^b \prod_\alpha H_\alpha,$$

*where $H_\alpha$ is a $\tau$-invariant, homogeneous polynomial, either constant or of degree* 24, *whose roots are complex numbers not equal to* $0, \pm 1$, *or any root of either polynomial* $z^8 + 14z^4 + 1$, *or* $z^{12} - 33z^8 - 33z^4 + 1$. *Here $\gamma$ is a constant and $a$, $b$ are nonnegative integers with* $b \equiv 1 + \delta(\mathcal{C}) \pmod 2$.

*Proof.* Since $\mathcal{C}$ is self-dual, all elements have even weight and thus $k$ is an even integer. A theorem of Gleason, Pierce, and Turyn [M-S, p. 597] implies $k = 2$ or 4. Since $n$ and every $d_j$ is a multiple of $k$, it follows from (2.1) that

$$X^r \left(g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X)\right) = M(Z,X) \tag{2.8}$$

for a polynomial $M(Z,X)$ in $Z^k$ and $X^k$. Since $M(Z,X) = M(Z,-X)$, it follows that the power of $(Z+X)$ dividing $M(Z,X)$ equals the power of $(Z-X)$ dividing $M(Z,X)$. The factorization in Proposition 2.3 implies this common power is $d_q - r$ and so $n - d_m = d_q$. In case $k = 2$, then $H(Z,X) = P_{\mathcal{C}}^{(u,v)}(Z,X)$ as in Proposition 2.3 and the conclusion $\tau(H(Z,X)) = (-1)^r H(Z,X)$ holds because the constant $\lambda$ in Proposition 2.3 reduces to $(-1)^r$ for a self-dual code ($d_i = w_i$ and $|\mathcal{C}| = 2^{n/2}$). Moreover $H(Z,X)$ is a polynomial in $Z^2$ and $X^2$.

If $k = 4$, then

$$(Z^2 + X^2)^{d_q - r} H(Z,X) = P_{\mathcal{C}}^{(u,v)}(Z,X).$$

One checks that $\tau(Z^2 + X^2) = Z^2 + X^2$ so $\tau(H(Z,X)) = (-1)^r\, H(Z,X)$ follows from the corresponding property of $P_{\mathcal{C}}^{(u,v)}(Z,X)$ given in Proposition 2.3. Furthermore

$$Z^{d_q} X^{d_q} (Z^4 - X^4) H(Z,X)$$

is a polynomial in $Z^4$ and $X^4$ (because $d_q$ is divisible by 4) and so $H(Z,X)$ is also.

Now we turn to the description of the possible $H$. This is accomplished by examining the factors as determined by the set of roots. If $Z - \alpha X$ is a factor of $H(Z,X)$ we say $\alpha$ is a root of $H$. Let $\mathcal{R}$ be the set of roots of $H$.

Assume $k = 2$. Because $H$ is a polynomial in $Z^2$, whenever $\alpha$ is a root, then so is $-\alpha$ so $\mathcal{R}$ is closed under multiplication by $-1$. In other words, if $M$ is the transformation of complex numbers $M(\alpha) = -\alpha$, then $\mathcal{R}$ is closed under the action of $M$.

Let $T$ be the transformation of complex numbers $\neq -1$ defined by

$$T(\alpha) = \frac{1 - \alpha}{1 + \alpha}.$$

Using the identity

$$\tau(X - \alpha Z) = -\frac{1 + \alpha}{\sqrt{2}}\,(X - T(\alpha)Z)$$

and the equation $\tau(H) = \pm H$, we conclude $\mathcal{R}$ is also invariant under transformation by $T$. The group of linear fractional transformations $\mathcal{G}$ generated by $T$ and $M$ has order 8. For a complex number $\alpha$, let $\mathcal{G}(\alpha) = \{B(\alpha) : B \in \mathcal{G}\}$ be the orbit of $\alpha$ under $\mathcal{G}$. Each orbit $\mathcal{G}(\alpha)$ corresponds to a polynomial

$$p_\alpha(X, Z) = \prod_{\beta \in \mathcal{G}(\alpha)} (X - \beta Z).$$

This polynomial is a relative invariant for $\tau$ and is always a polynomial in $X^2$ and $Z^2$. There are only a finite number of $\alpha$ for which $\mathcal{G}(\alpha)$ has fewer than eight elements. These are found by solving the seven quadratic equations

$$B(x) = \frac{a + bx}{c + dx} = x, \qquad B \in \mathcal{G} - \text{identity}.$$

One then forms the orbit of a solution and determines the corresponding polynomial. The orbits of length less than 8 are represented by 1, $i$, and $1 + \sqrt{2}$ giving rise to the polynomials

$$p_1 = X^2 - Z^2, \qquad p_i = X^2 + Z^2, \qquad p_{1+\sqrt{2}} = X^4 - 6X^2Z^2 + Z^4.$$

The first two are invariant under $\tau$; the last is changed to its negative by $\tau$. If $\alpha$ has an orbit of 8 elements, the polynomial $p_\alpha$ has degree 8 and has the form described above. Statement (iii) now follows.

For the case $k = 4$, let $M(\alpha) = i\alpha$ and $T$ as above. In this case the group $\mathcal{G}$ generated by $M$ and $T$ has order 24 and leaves $\mathcal{R}$ invariant; the orbits of complex numbers have length 24 except for a finite number of cases. The orbits with length less than 24 are represented by the elements

$$1, \quad \frac{1+i}{\sqrt{2}}, \quad \left(\frac{1+i}{2}\right)(1 + \sqrt{3}), \quad 1 + \sqrt{2}.$$

The corresponding polynomial factors are

$$X^4 - Z^4, \quad X^4 + Z^4, \quad X^8 + 14X^4Z^4 + Z^8, \quad X^{12} - 33X^8Z^4 - 33X^4Z^8 + Z^{12}$$

with each being $\tau$-invariant except the last which is transformed into its negative by $\tau$. An orbit of length 24 gives rise to a factor of $H$ having degree 24. Statement (iv) is a consequence of these facts. $\square$

**2.1. Extremal codes.** The computations of the previous subsection can be used to refine some known results about the design strength of *extremal, doubly even self-dual* codes.

Recall that a self-dual code $\mathcal{C}$ with parameters $[n, n/2, d]$ is doubly even if all weights are multiples of 4. By the result of MacWilliams and Sloane [M-S, p. 624], such a code satisfies

$$d \leq 4\left[\frac{n}{24}\right] + 4.$$

$\mathcal{C}$ is called *extremal* when equality holds. It is known that a doubly even self-dual code must have length divisible by 8 [M-S, Chapter 19]. The Assmus–Mattson bounds imply $\delta(\mathcal{C}) \geq 5 - 2\epsilon$. We give more precise information in the following.

THEOREM 2.10. *Let $\mathcal{C}$ be a doubly even, extremal self-dual code with parameters $[n, n/2, d]$ and length $n = 24\mu + 8\epsilon$, $\epsilon = 0$, 1, or 2. Then either*

(1) *$\delta(\mathcal{C}) \geq 7 - 2\epsilon$,*
*or*
(2) (i) *$\delta(\mathcal{C}) = 5 - 2\epsilon$, and* (ii) *no nontrivial weight class of $\mathcal{C}$ holds a $1 + \delta(\mathcal{C})$-design.*

*Proof.* The parameters of an extremal code require $d = d_1 = 4\mu + 4$ and so

$$s \leq \frac{n - 2d_1}{4} + 2 = 4\mu + 2\epsilon \tag{2.9}$$

since the right side counts the multiples of 4 on the interval $[d_1, n - d_1]$ plus 1 for weight $n$.

Select two vectors $u$ and $v$ of weight $r = 1 + \delta(\mathcal{C})$ such that $\Delta g_{\mathcal{C}} = g_{\mathcal{C}}^{(u)}(Z, X) - g_{\mathcal{C}}^{(v)}(Z, X) \neq 0$. Then $\Delta g_{\mathcal{C}}$ is homogeneous of degree $n - r$ by the definition. From the factorization given in Theorem 2.9(i) (with $k = 4$) we obtain

$$\deg(\Delta g_{\mathcal{C}}) = n - r = 6d_q - 5r + \deg(H(Z, X)). \tag{2.10}$$

Since all the weights of elements of $\mathcal{C}$ are multiples of 4, it follows that $d_q \geq d_1 + 4(q-1)$ with equality when $q = 1$. From (2.10) and $r = 1 + \delta(\mathcal{C})$ follows

$$4\delta(\mathcal{C}) \geq 6d_q - n - 4 + \deg(H) \geq 6d_1 + 24(q-1) - n - 4 + \deg(H)$$

with equality if $q = 1$. Now substitute $n = 24\mu + 8\epsilon$ and $d_1 = 4\mu + 4$ to conclude

$$\delta(\mathcal{C}) \geq 6q - 1 - 2\epsilon + \left(\frac{1}{4}\right)\deg(H).$$

Thus if there is any $u$ and $v$ such that $\Delta g_{\mathcal{C}} \neq 0$ and $q = q(u, v) > 1$, then $\delta(\mathcal{C}) \geq 11 - 2\epsilon$, and statement (1) holds. Thus we may assume that $q = 1$. Then we have the equality

$$\delta(\mathcal{C}) = 5 - 2\epsilon + \left(\frac{1}{4}\right)\deg(H).$$

The description of $H$ in Theorem 2.9(iv) implies either $H$ is a constant or $\deg(H) \geq 8$. If $H$ is a constant, then $\delta(\mathcal{C}) = 5 - 2\epsilon$ and the Assmus–Mattson bound gives the exact design strength. By Theorem 2.6(ii), no nontrivial weight class of $\mathcal{C}$ can hold a $1 + \delta(\mathcal{C})$-design; thus statement (2) is true. If $H$ is not constant, then $\deg(H) \geq 8$ and so $\delta(\mathcal{C}) \geq 7 - 2\epsilon$ and statement (1) holds. ∎

*Remarks.* There are no examples known where statement 2 is false. In fact there are no examples known (apart from the trivial cases $\mathcal{C}$ or $\mathcal{C}^{\perp}$ is contained in $\{0, \mathbf{1}\}$) where an extremal code holds a 6-design. The search for a 6-design in an extremal, doubly even code of length divisible by 24 seems unlikely to be successful; for if such a code has one nontrivial weight class holding a 6-design, then every weight class must hold a 7-design.

We give an example to illustrate the general results.

*Example* 2.11. We consider a class of cyclic subcodes of the punctured second order Reed–Muller code $\mathcal{R}(2, m)^*$. A number of these, each with slightly different parameters, are described in MacWilliams and Sloane [M-S, Chapter 8, section 7 and Chapter 15, section 4]. We focus on the code $\mathcal{C}$ described in [M-S, p. 451] having parameters $[n = 2^m - 1, 2m, d_1]$ with $m$ any odd integer at least 5 and three nonzero weights given by

$$d_1 = 2^{m-1} - 2^{(m-1)/2}, \qquad d_2 = 2^{m-1}, \qquad d_3 = 2^{m-1} + 2^{(m-1)/2}.$$

The weight enumerator is determined in [M-S, p. 451]. The dual code $\mathcal{C}^\perp$ has $w_1 = 5$ and so $\delta(\mathcal{C}) \geq 2$. If $\delta(\mathcal{C}) \geq 3$ then the words of weight 5 in $\mathcal{C}^\perp$ would hold a 3-design; the usual divisibility test shows this is impossible. Thus $\delta(\mathcal{C}) = 2$.

By Corollary 2.4 and using the particular constants for this code, the difference polynomial, for elements $u$ and $v$ of weight $r = 3$, is

$$\Delta g = g_{\mathcal{C}}^{(u)}(Z,X) - g_{\mathcal{C}}^{(v)}(Z,X) = \alpha Z^{d_1-1}X^{d_1-3}(Z^k - X^k)^2 \qquad k = 2^{(m-1)/2},$$

for some integer $\alpha$. Note that the undetermined polynomial $h(Z,X)$ in Corollary 2.4 is seen to be a constant here by degree considerations. We may apply (2.3) to conclude that

$$g_{\mathcal{C}^\perp}^{(u)}(Z,X) - g_{\mathcal{C}^\perp}^{(v)}(Z,X) = \lambda Z^3(Z^2 - X^2)^{d_1-4}(Z-X)[(Z+X)^k - (Z-X)^k]^2$$

where $\lambda$ is some constant. We have not been able to deal with this product for general $m$ but for the particular cases $m = 5, 7, 9, 11$, this polynomial has nonzero coefficients of $Z^{n-i}X^{i-3}$ for every $i$ on the interval $5 \leq i \leq n - i$. In these cases it follows that every integer on the interval $[5, n-5]$ is a weight of $\mathcal{C}^\perp$ (and together with $n$ are all the weights) and the corresponding weight classes of $\mathcal{C}^\perp$ hold 2-designs but none hold a 3-design (Proposition 2.1).

These examples illustrate the great disparity in the two test conditions giving the Assmus–Mattson bounds. While $w_1 - s = 5 - 3 = \delta(\mathcal{C})$ gives equality, the condition for the dual code reads $d_1 - s' + 1 \leq \delta(\mathcal{C})$. For $m = 5, 7, 9, 11$, these inequalities are, respectively, $-10$, $-62$, $-262$, $-1047$ $\leq \delta(\mathcal{C})$. Even though these are far from equality, the conclusion that no nontrivial weight class holds a $1 + \delta(\mathcal{C})$-design is still valid for $\mathcal{C}^\perp$.

*Example* 2.12. Let $\mathcal{C}$ be first order Reed–Muller code with parameters $[n = 2^{k-1}, k, 2^{k-2}]$ for $k \geq 3$. It weights are $d_1 = 2^{k-2}$ and $d_2 = n$. It is known that every weight class of $\mathcal{C}$ and $\mathcal{C}^\perp$ holds a 3-design; the minimum weight in $\mathcal{C}^\perp$ is 4 and Assmus–Mattson gives $\delta \geq 3$. Since $\delta(\mathcal{C}) = \delta(\mathcal{C}^\perp)$ and $\mathcal{C}^\perp$ contains an element of weight 4, we cannot have $\delta(\mathcal{C}) \geq 4$. Thus $\delta(\mathcal{C}) = 3$ and we have the case of equality $w_1 - s + \epsilon_{\mathcal{C}} = \delta(\mathcal{C})$ in Theorem 2.6. Thus no nontrivial weight class of $\mathcal{C}$ can hold a 4-design. By an analysis similar to that in the preceding example, one shows no nontrivial weight class of $\mathcal{C}^\perp$ holds a 4-design. The key idea is that for two vectors $u$ and $v$ of weight 4 we have

$$g_{\mathcal{C}^\perp}^{(u)}(X) - g_{\mathcal{C}^\perp}^{(v)}(X) = \lambda Z^4(Z+X)^{n-d_1-4}(Z-X)^{n-d_1-4} = \lambda Z^4(Z^2 - X^2)^{n/2-4},$$

with $\lambda$ some constant. There exist $u$ and $v$ with $\lambda \neq 0$ and then this difference has exactly $n/2 - 4 + 1 = 2^{k-2} - 3$ nonzero terms of the shape $c_t\, Z^{n-w_t}X^{w_t-4}$. We know $\mathbf{1} \in \mathcal{C}^\perp$ and, excluding 0 and $n$, the weights of $\mathcal{C}^\perp$ are even integers on the interval $[4, n-4]$. There are $(n-6)/2 = 2^{k-2} - 3$ even integers on the interval, so it follows that every nontrivial weight class corresponds to a nonzero term in the difference of the covering polynomials. We conclude no nontrivial weight class of $\mathcal{C}^\perp$ holds a 4-design (by Proposition 2.1).

## 3. Application to self-dual codes.

**3. Application to self-dual codes.** Suppose $\mathcal{C}$ is a self-dual code with parameters $[n, n/2, d]$. We can take advantage of the duality principle (Theorem 1.10) and the equation in Theorem 1.5 to express conditions on the coefficients of the overlap polynomials for $\mathcal{C}$ and determine them in some cases. The idea is to compute $F_{\mathcal{C}}^{(u)}$ from a knowledge of the $F_{\mathcal{C}}^{(v)}$ for $\mathrm{wt}(v) < \mathrm{wt}(u)$ and some other data. Of course it is sufficient to determine the covering polynomials in view of Theorem 1.5. We focus our attention on determining the $g_{\mathcal{C}}^{(u)}(Z,X)$. We first make some initial observations.

The overlap polynomial corresponding to the zero vector $\mathbf{0}$ is just the weight enumerator; that is,

$$F_{\mathcal{C}}^{(\mathbf{0})}(Z,X,D) = g_{\mathcal{C}}^{(\mathbf{0})}(Z,X) = \sum_{w\in\mathcal{C}} Z^{\mathrm{wt}(w)-0}X^0D^0 = W_{\mathcal{C}}(1,Z).$$

The process of determining the overlap and covering polynomials begins with the weight enumerator.

For a vector $u$ of weight 1, we have from (1.2)

$$Z\,\tau\Big(g_{\mathcal{C}}^{(u)}(Z,X)\Big) = -g_{\mathcal{C}}^{(u)}(Z,X)(Z+X) + W_{\mathcal{C}}(1,Z).$$

Assuming that the weight enumerator is explicitly known, this equation provides linear constraints on the unknown coefficients of $g_{\mathcal{C}}^{(u)}$.

We go to the general case, rather than deal with this special case any further.

We will use the form of $g_{\mathcal{C}}^{(u)}(Z,X)$ in (2.1) given in terms of constants $b_{\mathcal{C}}[i,u]$ which we now write as $b[i,u]$ since $\mathcal{C}=\mathcal{C}^{\perp}$.

Begin by expanding the term involving $\tau\left(g^{(u)}(Z,X)\right)$ as an explicit polynomial. The Krawtchouk notation is convenient. Define the integers $P_q(t;m)$ by the generating function

$$(Z-X)^t(Z+X)^{m-t} = \sum_{q=0}^{m} P_q(t;m)Z^{m-q}X^q.$$

The explicit form is

$$P_q(t;m) = \sum_{j=0}^{q}(-1)^j\binom{t}{j}\binom{m-t}{q-j}, \qquad q=0,1,\ldots,m.$$

Of course we follow the usual convention that $\binom{t}{j}=0$ if $t<j$. This representation can be used in the description of the terms in (2.1) above.

Recursion Equation 3.1. *Let $\mathcal{C}$ be a self-dual binary code of length $n$ and let $u$ be a vector in $\mathbf{F}^n$ of weight $r$. Let the covering polynomial $g^{(u)}(Z,X)$ be given with coefficients $b[i,u]$ as in (2.1). For $u\neq 0$, $b[0,u]=0$. For any $u$, $b[s,u]=1$. The remaining $b[i,u]$, $1\le i<s$, satisfy linear equations obtained by equating to zero the coefficients of powers of $X$ in the following polynomial:*

$$\sum_{q=0}^{n-r}\left\{\sum_{\substack{0\le t\le s\\ d_t\ge r}} b[t,z]P_q(d_t-r;n-r)\right\}X^q$$

$$+(-1)^{r-1}2^{(n/2)-r}(1+X)^r\sum_{\substack{0\le t\le s\\ d_t\ge r}} b[t,z]X^{d_t-r}$$

$$-2^{(n/2)-r}\sum_{k=0}^{r-1}(-1)^k(1+X)^k\sum_{\substack{w\le u\\ \mathrm{wt}(w)=k}} g^{(w)}(1,X).$$

The polynomial is obtained by expressing $\tau(g^{(u)}(Z,X))$ using the Krawtchouk notation, applying (1.2) and rearranging some of the terms, and then setting $Z=1$.

We generally have $n-r+1$ equations but the number of unknowns is difficult to predict, in general. For $\mathrm{wt}(u)=1$ there are $s-1$ unknowns. When $\mathrm{wt}(u)\geq 2$ there may be more unknowns. At least for small $r$, the system is overdetermined but must have a solution for any given code. Condition (i) of Proposition 1.1 indicates that the recursion need only be applied up to the mid-point of the weights and by symmetry all others will be obtained. This system will have a unique solution if $\mathrm{wt}(u)\leq\delta(\mathcal{C})$. Of course in that case we know the covering polynomial in this case from the computation of the average given in Corollary 1.8(i).

We illustrate the process for very small weights. Let $\mathrm{wt}(u)=1$. The recursion equation involves the unknowns $b[t,u]$ and the sum of the $g^{(w)}(1,X)$ for $\mathrm{wt}(w)<\mathrm{wt}(u)=1$; only $g^{(0)}(1,X)$ appears in this term. The system of linear equations for the unknowns $b[i,u]$ is obtained by setting the coefficients of distinct powers of $X$ equal to 0: recall that $b[0,u]=0$ and $b[s,u]=1$:

$$\begin{array}{lll} d_j-1, \quad 1\leq j\leq s-1, & 0=|\mathcal{C}|b[j,u]+2\sum_{t=1}^{s}P_{d_j-1}(d_t-1;n-1)b[t,u], \\ d_j, \quad 0\leq j\leq s, & 0=|\mathcal{C}|b[j,u]-|\mathcal{C}|\,A_{d_j}+2\sum_{t=1}^{s}P_{d_j}(d_t-1;n-1)b[t,u], \\ \text{other exponents, } q, & 0=\sum_{t=1}^{s}b[t,u]\,P_q(d_t-1;n-1). \end{array}$$

In some situations, this system has a unique solution. In general, the system has some rank $\rho\leq s-1$ and $\rho$ of the variables can be expressed as linear forms in the remaining $s-1-\rho$ variables. We illustrate the next step under the assumption that the first $\rho$ variables $b[1,u],\dots,b[\rho,u]$ can be expressed in terms of $b[\rho+1,u],\dots,b[s-1,u]$. (I know of no example where this is not the case.) Another level of subscripting would be necessary to illustrate the most general possibility. The system produces

$$b[i,u]=\alpha_i+\sum_{t=\rho+1}^{s-1}\beta_{i,t}b[t,u], \qquad 1\leq i\leq\rho,$$

where the rational numbers $\alpha_i$ and $\beta_{i,t}$ depend only on the parameters of the code, $n$, $d_j$, and $A_{d_j}$; that is, these depend only on the weight enumerator $W_{\mathcal{C}}(1,X)$.

Now substitute these values of $b[i,u]$ into the definition of $g^{(u)}(1,X)$ to get

$$g^{(u)}(X)=\sum_{t=1}^{s}b[t,u]\,X^{d_t-1}=G_1(X)+\sum_{t=\rho+1}^{s-1}b[t,u]p_t(X),$$

where

$$G_1(X)=X^{n-1}+\sum_{i=1}^{\rho}\alpha_iX^{d_i-1},$$
$$p_t(X)=X^{d_t-1}+\sum_{i=1}^{\rho}\beta_{i,t}X^{d_i-1}, \quad \rho+1\leq t\leq s-1.$$

We emphasize that the polynomials $G_1(X)$ and $p_t(X)$ are independent of the particular vector $u$ so long as $\mathrm{wt}(u)=1$.

Now move on to compute $g^{(u)}(X)$ for $\mathrm{wt}(u)=2$. The recursion equation requires the sum of the terms $g^{(w)}(X)$ for $\mathrm{wt}(w)=1$ and $w\in S(u)$. This sum is

$$\sum_{\substack{\mathrm{wt}(w)=1\\ w\leq u}}g^{w}(X)=\sum_{\substack{\mathrm{wt}(w)=1\\ w\leq u}}\left\{G_1(X)+\sum_{t=\rho+1}^{s-1}b[t,w]p_t(X)\right\}$$

$$= 2\,G_1(X) + \sum_{t=\rho+1}^{s-1} B[t,u]\,p_t(X),$$

where

$$B[t,u] = \sum_{\substack{w \leq u \\ \mathrm{wt}(w)=1}} b[t,w].$$

Regard $B[\rho+1,u],\dots,B[s-1,u]$ as parameters involved in the evaluation of $g^{(u)}(X)$. By substituting this value for $\sum g^{(w)}(X)$ into the recursion equation, we obtain a polynomial, involving polynomials $G_1(X)$ and $p_t(X)$ independent of $u$, the unknowns $b[1,u],\dots,b[s-1,u]$, and parameters $B[\rho+1,u],\dots,B[s-1,u]$, all of whose coefficients equal 0. This is a linear system subject to the same considerations just made. It is a bit cumbersome to write this system explicitly but it lends itself to machine computation.

*Example* 3.1. Consider a self-dual code with parameters $[32,16,8]$ and weight enumerator

$$W(X) = 1 + 364X^8 + 2048X^{10} + 6720X^{12} + 14336X^{14} + 18598X^{16} + \cdots + X^{32}.$$

Such a code exists; see [C-S, p. 1325]. Thus $s = 10$ and $d_1 - s + 1 = -1$ so the Assmus–Mattson theorem does not give any information about $\delta(\mathcal{C})$. However, the recursion equation for the covering polynomials corresponding to elements of weight 1 has the unique solution

$$\begin{aligned} g_{\mathcal{C}}^{(u)}(1,X) = 91X^7 + 640X^9 + 2520X^{11} + 6272X^{13} + 9299X^{15} + 8062X^{17} \\ +4200X^{19} + 1408X^{21} + 273X^{23} + X^{31}. \end{aligned}$$

Thus $\delta(\mathcal{C}) \geq 1$. For vectors of weight 2, the solution of the equations for $g^{(u)}(X)$ are not unique and one easily checks numerical conditions to verify that the code does not hold 2-designs. Thus $\delta(\mathcal{C}) = 1$.

**4. The $[72,36,16]$ problem.** Consider the (putative) extremal, doubly even, self-dual code $\mathcal{C}_{72}$ with parameters $[72,36,16]$. The question of the existence of such a code is a long standing problem that has been considered by many researchers. (See [M-S, p. 626] and [A-P] and some references therein.) We present some facts here that are a consequence of the recursion equation with the hope that they may be of some value in the construction of the code or proving its nonexistence.

We use slightly inaccurate terminology by referring to "the code" as if it exists.

One may carry out many computations based on the recurrence equation, but one has no idea of the value of some of the information so obtained. Our choice here is to present the weight enumerators of the cosets $u + \mathcal{C}_{72}$. It is known that the covering radius of $\mathcal{C}_{72}$ lies between 10 and 12 [A-P] so one need only consider cosets of weight up to 12. (The weight of a coset is the least weight of an element in the coset.) Assmus and Pless [A-P] also show that the weight enumerator of a coset of weight 11 or 12 (if either exist) is uniquely determined. We determine these enumerators under the assumption that they exist. For the cosets of weight between 6 and 10, some parameters remain undetermined: two parameters in the case of the weight 8 coset and one parameter in the four other cases.

The method for computing the weight enumerators of the cosets is to first determine the overlap polynomial $F_{\mathcal{C}_{72}}^{(u)}(Z,X,D)$ for vectors of weight at most 12 using the recursion equation in section 3. From that and Proposition 1.1(ii), we then obtain an expression for the weight enumerator of the coset. Generally there will be several constants appearing in $F^{(u)}_{\mathcal{C}_{72}}(Z, X, D)$ that can be eliminated from the coset weight enumerator by assuming $u$ is the element of minimum weight in the coset. This assumption will force certain coefficients to equal 0.

In the listing below, $W(X)$ is the weight enumerator of the code; the cosets of weight at most 5 have uniquely determined weight enumerators because the code has $\delta(\mathcal{C}) = 5$. The formula for these cosets is given in Corollary 1.8A. The weight enumerators of the cosets of weight $i$ are denoted as *coset*[$i$]. Since the terms $X^{n-j}Y^j$ and $X^jY^{n-j}$ have equal coefficients, only the terms with $j \leq 36$ are displayed and we set $Y = 1$. The undetermined parameters $\epsilon(i, j)$ are nonnegative integers with the notation selected so that $\epsilon(i, j)$ is the number of elements of weight $i$ in a coset of weight $j$. It is possible that two different cosets of the same weight require different values of the parameters; that is $\epsilon(i, j)$ really depends on the particular coset, and not just $i$ and $j$. The bound on the $\epsilon(i, j)$ indicated after its appearance is obtained from the fact that the coefficients in the enumerator are nonnegative. In most cases these bounds can be improved by making separate arguments in each case.

**Weight enumerators of the cosets $u + \mathcal{C}_{72}$.**

$$\begin{aligned} W(X) =\ & 1 + 249849X^{16} + 18106704X^{20} + 462962955X^{24} \\ & + 4397342400X^{28} + 16602715899X^{32} + 25756721120X^{36} \\ & + 16602715899X^{40} + 4397342400X^{44} + 462962955X^{48} \\ & + 18106704X^{52} + 249849X^{56} + X^{72}. \end{aligned}$$

For $1 \leq k \leq 5$,

$$\textit{coset}[k] = \binom{72}{k}^{-1} \sum_{i=0}^{k} \sum_{j=0}^{12} \binom{72-d_j}{i} \binom{d_j}{k-i} A_{d_j} X^{72+k-2i-d_j} Y^{d_j-k+2i}.$$

$$\begin{aligned} \textit{coset}[6] =\ & X^6 + X^{10}\,\epsilon(10,6) + X^{12}\ (468 - 6\,\epsilon(10,6)) + X^{14}\ (8910 + 5\,\epsilon(10,6)) \\ & + X^{16}\ (117744 + 40\,\epsilon(10,6)) + X^{18}\ (1215005 - 90\,\epsilon(10,6)) \\ & + X^{20}\ (9063912 - 76\,\epsilon(10,6)) + X^{22}\ (52128219 + 406\,\epsilon(10,6)) \\ & + X^{24}\ (231465520 - 120\,\epsilon(10,6)) + X^{26}\ (802952286 - 925\,\epsilon(10,6)) \\ & + X^{28}\ (2198865900 + 870\,\epsilon(10,6)) + X^{30}\ (4782282868 + 1143\,\epsilon(10,6)) \\ & + X^{32}\ (8300716512 - 1968\,\epsilon(10,6)) + X^{34}\ (11541281895 - 540\,\epsilon(10,6)) \\ & + X^{36}\ (12879278256 + 2520\,\epsilon(10,6)) + \cdots, \\ & 0 \leq \epsilon(10,6) \leq 78. \end{aligned}$$

$$\begin{aligned} \textit{coset}[7] =\ & X^7 + X^9\,\epsilon(9,7) + X^{11}\ (78 - \epsilon(9,7)) \\ & + X^{13}\ (2150 - 29\,\epsilon(9,7)) + X^{15}\ (33150 + 93\,\epsilon(9,7)) \\ & + X^{17}\ (396229 + 62\,\epsilon(9,7)) + X^{19}\ (3425500 - 638\,\epsilon(9,7)) \\ & + X^{21}\ (22482876 + 498\,\epsilon(9,7)) + X^{23}\ (113347715 + 1742\,\epsilon(9,7)) \\ & + X^{25}\ (444247726 - 2981\,\epsilon(9,7)) + X^{27}\ (1368169130 - 1819\,\epsilon(9,7)) \\ & + X^{29}\ (3336322210 + 7257\,\epsilon(9,7)) + X^{31}\ (6478822580 - 1497\,\epsilon(9,7)) \\ & + X^{33}\ (10061693655 - 9708\,\epsilon(9,7)) + X^{35}\ (12530795368 + 7020\,\epsilon(9,7)) + \cdots, \\ & 0 \leq \epsilon(9,7) \leq 78. \end{aligned}$$

$$\begin{aligned} coset[8] = {} & X^8\epsilon(8,8) + X^{10}\epsilon(10,8) + X^{12}\,(546 - 14\epsilon(8,8) - 6\epsilon(10,8)) \\ & + X^{14}\,(8640 - 64\epsilon(8,8) + 5\epsilon(10,8)) \\ & + X^{16}\,(119016 + 475\epsilon(8,8) + 40\epsilon(10,8)) \\ & + X^{18}\,(1207360 - 448\epsilon(8,8) - 90\epsilon(10,8)) \\ & + X^{20}\,(9083844 - 2156\epsilon(8,8) - 76\epsilon(10,8)) \\ & + X^{22}\,(52123968 + 4928\epsilon(8,8) + 406\epsilon(10,8)) \\ & + X^{24}\,(231383880 + 1897\epsilon(8,8) - 120\epsilon(10,8)) \\ & + X^{26}\,(803097792 - 15680\epsilon(8,8) - 925\epsilon(10,8)) \\ & + X^{28}\,(2198885310 + 9262\epsilon(8,8) + 870\epsilon(10,8)) \\ & + X^{30}\,(4781952384 + 22912\epsilon(8,8) + 1143\epsilon(10,8)) \\ & + X^{32}\,(8301020112 - 31045\epsilon(8,8) - 1968\epsilon(10,8)) \\ & + X^{34}\,(11541479040 - 11648\epsilon(8,8) - 540\epsilon(10,8)) \\ & + X^{36}\,(12878752952 + 43160\epsilon(8,8) + 2520\epsilon(10,8)) + \cdots, \\ & 1 \le \epsilon(8,8) \le 9, \quad 7\epsilon(8,8) + 3\epsilon(10,8) \le 273. \end{aligned}$$

$$\begin{aligned} coset[9] = {} & X^9\,\epsilon(9,9) + X^{11}\;(91 - \epsilon(9,9)) \\ & + X^{13}\;(2135 - 29\,\epsilon(9,9)) + X^{15}\;(33408 + 93\,\epsilon(9,9)) \\ & + X^{17}\;(394408 + 62\,\epsilon(9,9)) + X^{19}\;(3428810 - 638\,\epsilon(9,9)) \\ & + X^{21}\;(22487322 + 498\,\epsilon(9,9)) + X^{23}\;(113325160 + 1742\,\epsilon(9,9)) \\ & + X^{25}\;(444263456 - 2981\,\epsilon(9,9)) + X^{27}\;(1368212321 - 1819\,\epsilon(9,9)) \\ & + X^{29}\;(3336243405 + 7257\,\epsilon(9,9)) + X^{31}\;(6478814496 - 1497\,\epsilon(9,9)) \\ & + X^{33}\;(10061820720 - 9708\,\epsilon(9,9)) + X^{35}\;(12530712636 + 7020\,\epsilon(9,9)) + \cdots, \\ & 1 \le \epsilon(9,9) \le 73. \end{aligned}$$

$$\begin{aligned} coset[10] = {} & X^{10}\epsilon(10,10) + X^{12}\,(546 - 6\epsilon(10,10)) \\ & + X^{14}\,(8640 + 5\epsilon(10,10)) + X^{16}\,(119016 + 40\epsilon(10,10)) \\ & + X^{18}\,(1207360 - 90\epsilon(10,10)) + X^{20}\,(9083844 - 76\epsilon(10,10)) \\ & + X^{22}\,(52123968 + 406\epsilon(10,10)) + X^{24}\,(231383880 - 120\epsilon(10,10)) \\ & + X^{26}\,(803097792 - 925\epsilon(10,10)) + X^{28}\,(2198885310 + 870\epsilon(10,10)) \\ & + X^{30}\,(4781952384 + 1143\epsilon(10,10)) + X^{32}\,(8301020112 - 1968\epsilon(10,10)) \\ & + X^{34}\,(11541479040 - 540\epsilon(10,10)) + X^{36}\,(12878752952 + 2520\epsilon(10,10)) + \cdots, \\ & 1 \le \epsilon(10,10) \le 91. \end{aligned}$$

$$\begin{aligned} coset[11] = {} & 91\,X^{11} + 2135\,X^{13} + 33408\,X^{15} + 394408\,X^{17} + 3428810\,X^{19} \\ & + 22487322\,X^{21} + 113325160\,X^{23} + 444263456\,X^{25} \\ & + 1368212321\,X^{27} + 3336243405\,X^{29} + 6478814496\,X^{31} \\ & + 10061820720\,X^{33} + 12530712636\,X^{35} + \cdots. \end{aligned}$$

$$\begin{aligned} coset[12] = {} & 546\,X^{12} + 8640\,X^{14} + 119016\,X^{16} + 1207360\,X^{18} \\ & + 9083844\,X^{20} + 52123968\,X^{22} + 231383880\,X^{24} \\ & + 803097792\,X^{26} + 2198885310\,X^{28} + 4781952384\,X^{30} \\ & + 8301020112\,X^{32} + 11541479040\,X^{34} + 12878752952\,X^{36} + \cdots . \end{aligned}$$

**Acknowledgments.** The author thanks an anonymous referee for several improvements to the early version, in particular for the suggestion to use homogeneous polynomials. The author also expresses his thanks to Gary McGuire for his suggestion to consider the $[16, 9, 4]$ code (see Example 2.7).

## REFERENCES

[A-M] E. F. Assmus Jr. and H. F. Mattson Jr., *New 5-designs*, J. Combin. Theory, 6 (1969), pp. 122–151.

[A-P] E. F. Assmus Jr. and V. Pless, *On the covering radius of extremal self-dual codes*, IEEE Trans. Inform. Theory, IT-29, (1983), pp. 359–363.

[C-S] J. H. Conway and N. J. A. Sloane, *A new upper bound on the minimal distance of self-dual codes*, IEEE Trans. Inform. Theory, 36 (1990), pp. 1319–1333.

[CRC] C. J. Colbourn and J. H. Dinitz, eds., *The CRC Handbook of Combinatorial Designs,* CRC Press, Boca Raton, FL, 1996.

[M-S] F. J. MacWilliams and N. J. A. Sloane, *Theory of Error Correcting Codes*, North–Holland, Amsterdam, 1977.

[Oz] M. Ozeki, *On the notion of Jacobi polynomials for codes*, Math. Proc. Cambridge Philos. Soc., 121 (1997), pp. 15–30.

SIAM J. Discrete Math. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 2, pp. 179–193

# CONVEX-ROUND AND CONCAVE-ROUND GRAPHS*

JØRGEN BANG-JENSEN[†], JING HUANG[‡], AND ANDERS YEO[§]

**Abstract.** We introduce two new classes of graphs which we call convex-round, respectively concave-round graphs. Convex-round (concave-round) graphs are those graphs whose vertices can be circularly enumerated so that the (closed) neighborhood of each vertex forms an interval in the enumeration. Hence the two classes transform into each other by taking complements. We show that both classes of graphs have nice structural properties. We observe that the class of concave-round graphs properly contains the class of proper circular arc graphs and, by a result of Tucker [*Pacific J. Math.*, 39 (1971), pp. 535–545] is properly contained in the class of general circular arc graphs. We point out that convex-round and concave-round graphs can be recognized in $O(n + m)$ time (here $n$ denotes the number of vertices and $m$ the number of edges of the graph in question). We show that the chromatic number of a graph which is convex-round (concave-round) can be found in time $O(n + m)$ ($O(n^2)$). We describe optimal $O(n + m)$ time algorithms for finding a maximum clique, a maximum matching, and a Hamiltonian cycle (if one exists) for the class of convex-round graphs. Finally, we pose a number of open problems and conjectures concerning the structure and algorithmic properties of the two new classes and a related third class of graphs.

**Key words.** proper circular arc graphs, round graphs, round enumeration, Hamiltonian cycle, coloring, maximum matching, maximum clique, linear algorithms, recognition

**AMS subject classifications.** 05C85, 05C38, 05C70, 68R10

**PII.** S0895480197322154

## 1. Introduction.

An *interval graph* is the intersection graph of a family of linear intervals; a *proper interval graph* is the intersection graph of an inclusion-free family of linear intervals. A *circular arc graph* is the intersection graph of a family of circular arcs; as above, it is called *proper* if the family can be chosen to be inclusion-free.

Algorithmic aspects of interval graphs and circular arc graphs have been intensively studied; cf. [13]. Interval graphs can be recognized in linear time (i.e., $O(n+m)$ where $n$ and $m$ are, respectively, the numbers of vertices and of edges of the input graph) [6, 19]. There exists an $O(n^2)$ algorithm to recognize circular arc graphs [10]. Many optimization problems such as finding a maximum independent set, a maximum clique, a minimum dominating set, and so on, can all be solved efficiently for both interval graphs and circular arc graphs [1, 4, 15]. It is worth noticing that the minimum coloring problem is solvable in polynomial time for proper circular arc graphs but NP-complete for general circular arc graphs [3, 11, 26].

An important feature of proper circular arc graphs is that the vertices can be circularly enumerated so that the neighbors of each vertex $v$, together with $v$ itself,

*Received by the editors May 30, 1997; accepted for publication (in revised form) December 16, 1999; published electronically April 6, 2000.
http://www.siam.org/journals/sidma/13-2/32215.html

[†]Department of Mathematics and Computer Science, Odense University, Odense, DK-5230, Denmark (jbj@imada.sdu.dk). The research of this author was supported by the Curtin University of Technology, Perth, Australia, and NSERC grant 203191-98.

[‡]Department of Mathematics and Statistics, University of Victoria, Victoria, BC, V8W 3P4, Canada (jing@math.uvic.ca). The research of this author was supported by the Danish Research Council under grant 11-9743-1, the Australia Research Council under grant A49532661, and NSERC under grant 203191-98.

[§]Brics, Department of Computer Science, University of Aarhus, Ny Munkegade, Building 540, DK-8000 Aarhus C, Denmark (yeo@daimi.au.dk). The research of this author was supported by the Danish Research Council under grant 9800435.

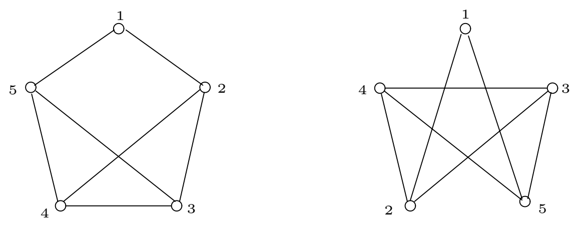


FIG. 1. *A graph which is concave-round and convex-round.*

appear consecutively in the enumeration (see [13]). More precisely, if $G$ is a proper circular arc graph, then the vertices of $G$ can be circularly enumerated, $v_1, v_2, \ldots, v_n$, so that the closed neighborhood of each vertex $v_i$ forms a set $\{v_{i-l}, v_{i-l+1}, \ldots, v_{i+r}\}$ for some $l, r \geq 0$ (which depend on $i$), where the addition and subtraction are modulo $n$. However, not all circular arc graphs enjoy this property. Because of this, many problems can be solved much more efficiently for proper circular arc graphs than for general circular arc graphs. For instance, the recognition problem and the maximum clique problem for proper circular arc graphs can be solved in linear time (see [3, 9]), but no linear time algorithms are known for general circular arc graphs.

In order to better understand the above feature of proper circular arc graphs, we introduce two new classes of graphs as follows: A graph $G$ is *concave-round* if the vertices of $G$ can be circularly enumerated, $v_1, v_2, \ldots, v_n$, so that the closed neighborhood of each vertex $v_i$ forms a set $\{v_{i-l}, v_{i-l+1}, \ldots, v_{i+r}\}$ for some $l, r \geq 0$. Here $l, r$ depend on $i$ and the addition is modulo $n$. We shall refer to the circular enumeration $v_1, v_2, \ldots, v_n$ as a *concave-round enumeration* and often denote it by $\mathcal{L}$. A graph is *convex-round* if it is the complement of a concave-round graph, that is, if the neighborhood of each vertex forms an interval in the enumeration. Within the class of bipartite graphs, convex-round graphs form a proper subclass of so-called *convex bipartite graphs*. Convex bipartite graphs have nice algorithmic properties and practical applications; see [7, 8, 12, 20, 22, 24, 29, 32]. In [20], a superclass of convex bipartite graphs was introduced under the name *circular convex bipartite graphs*. Even though this name seems to indicate that a circular convex bipartite graph must also be convex-round, this is not the case, as one can see from the definition in [20].

Let $\mathcal{L} = v_1, v_2, \ldots, v_n$ be a circular enumeration of the vertices of a graph $G$. A vertex $v_i$ is *concave* with respect to $\mathcal{L}$ if the closed neighborhood of $v_i$ forms an interval in $\mathcal{L}$. A vertex $v_i$ is *convex* with respect to $\mathcal{L}$ if its neighborhood forms an interval in $\mathcal{L}$. A circular enumeration $\mathcal{L}$ of $V(G)$ is a *concave-round* (*convex-round*) enumeration of $V(G)$ if all vertices of $G$ are concave (convex) with respect to $\mathcal{L}$. Figure 1 shows a concave-round and a convex-round enumeration of the same graph.

It is easy to see that every induced subgraph of a concave-round (convex-round) graph is concave-round (convex-round). It follows from the definition of a concave-round graph and an earlier remark on a feature of proper circular arc graphs that all proper circular arc graphs are concave-round graphs. But the converse is not true, that is, there exist concave-round graphs which are not proper circular arc graphs. So the class of concave-round graphs strictly contains the class of proper circular arc graphs. See Figure 2 for examples of graphs showing relationships between the above classes.

In this paper we study the structure of concave-round and convex-round graphs. We prove that the chromatic number problem can be solved in time $O(n^2)$ for both

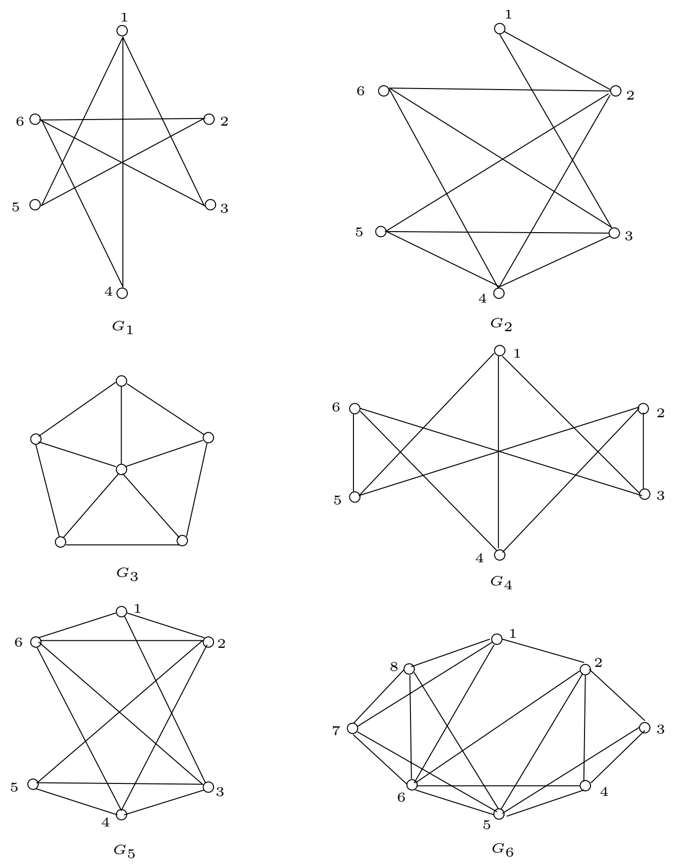


FIG. 2. *Graphs showing the relations between concave-round, convex-round graphs, and circular arc graphs: $G_1, G_4$ are convex-round but not circular arc graphs; $G_3$ is a circular arc graph but is neither concave-round nor convex-round; $G_5$ is a proper circular arc graph and concave-round; $G_6$ is concave-round but not a proper circular arc graph; $G_2$ is neither concave-round nor convex-round, but with respect to the labelling shown, every vertex is either concave or convex.*

of these classes. (The reason why our algorithms are $O(n^2)$ is that the operation of taking complementary graphs is used.) We describe optimal linear time algorithms for finding a maximum clique, a maximum matching, and a Hamiltonian cycle (if one exists) for convex-round graphs. Since every concave-round graph is a circular arc graph (see Theorem 3.1) and since the above three problems can be solved efficiently for circular arc graphs (see [4, 23, 27]), these three problems are efficiently solvable for concave-round graphs. Due to this and space considerations, we shall concentrate more on the algorithmic aspects of convex-round graphs. However, we wish to point out that none of the algorithms in [4, 23, 27] achieves optimal running time and we believe this is possible for each of the above problems when restricted to concave-round graphs.

## 2. Terminology and notation.

**2. Terminology and notation.** We assume all graphs are finite and simple, i.e., they contain no loops or multiple edges. For standard terminology, we refer to [5].

Let $G$ be a graph. We shall use $V(G)$ (resp., $E(G)$) to denote the set of vertices (resp., the set of edges) in $G$. We shall always use $n$ (resp., $m$) to denote the number of vertices (resp., edges) of $G$. Two vertices $x, y \in V(G)$ are *adjacent* and $y$ is a *neighbor* of $x$ if $xy \in E(G)$. If $xy \notin E(G)$, then $y$ is a *nonneighbor* of $x$. The *neighborhood* of $x$, denoted by $N(x)$, is the set of all neighbors of $x$. The *closed neighborhood* of $x$, denoted by $N[x]$, is defined as $N(x) \cup \{x\}$.

A graph is *bipartite* if the vertex set $V(G)$ can be partitioned into two sets $A$ and $B$ such that every edge of $G$ has one endvertex in $A$ and the other in $B$. We shall often use the terminology $(A, B)$ for a bipartite graph with vertex set $A \cup B$ and we only specify the set of edges $E$ when this is necessary.

A graph $G'$ is a *subgraph* of $G$ if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. If, in addition, $E(G') = \{xy \in E|\ x, y \in V(G')\}$, then $G'$ is an *induced subgraph* of $G$. For each $S \subseteq V(G)$, the subgraph of $G$ *induced by* $S$, denoted by $G\langle S\rangle$ or simply $S$, is the unique induced subgraph of $G$ with vertex set $S$. A *clique* of $G$ is a subgraph $G'$ of $G$ for which every vertex in $V(G')$ is adjacent to every other vertex in $V(G')$. The size of a largest clique in a graph $G$ is denoted by $\omega(G)$.

Suppose that $G$ is a graph and $S \subseteq V(G)$ is a set of vertices of $G$. We shall use $G - S$ to denote the subgraph induced by $V(G) - S$. We shall write $G - x$ instead of $G - \{x\}$. If $S$ contains no adjacent vertices, then $S$ is called an *independent set* of $G$. The size of a largest independent set is denoted by $\alpha(G)$.

Let $G$ be a graph. The *complement* of $G$, denoted by $\overline{G}$, is a graph such that the vertex set of $\overline{G}$ is $V(G)$ and two vertices are adjacent in $\overline{G}$ if and only if they are not adjacent in $G$.

A *path* $P$ of length $k$ is a graph with the vertex set $\{x_1, x_2, \ldots, x_{k+1}\}$ and the edge set $\{x_1x_2, x_2x_3, \ldots, x_kx_{k+1}\}$, such that all the vertices shown are distinct. We shall call such a path an $(x_1, x_{k+1})$-path and denote it by $x_1x_2x_3 \ldots x_kx_{k+1}$. A *cycle* $C$ of length $k$ is a graph with the vertex set $\{v_1, v_2, \ldots, v_k\}$ and the edge set $\{x_1x_2, x_2x_3, \ldots,\ x_{k-1}x_k, x_kx_1\}$. A *Hamiltonian* path (cycle) in a graph $G$ is a path (cycle) with vertex set $V(G)$.

Suppose that $x_1x_2x_3 \ldots x_k$ is a path (resp., a cycle). Then vertices $v_i$ and $v_{i+1}$ are called *consecutive vertices*. (The subscript addition is modulo $k$ in the case of cycle.) A path or a cycle is *chordless* (in a graph $G$) if nonconsecutive vertices are not adjacent (in $G$).

A graph $G$ is an *interval graph* if there is a one-to-one correspondence between $V(G)$ and a family $\mathcal{I}$ of intervals on the real line so that two vertices in $G$ are adjacent if and only if the two corresponding intervals intersect. If the family $\mathcal{I}$ can be chosen so that no interval is contained in another, then $G$ is called a *proper interval graph*. A graph $G$ is a *circular arc graph* if there is a one-to-one correspondence between $V(G)$ and a family $\mathcal{F}$ of circular arcs on a circle so that two vertices are adjacent in $G$ if and only if the corresponding two circular arcs intersect. As above, if the family $\mathcal{F}$ can be chosen so that no circular arc is contained in another, then $G$ is called a *proper circular arc graph.*

A matrix whose entries belong to $\{0, 1\}$ has the *circular* 1*'s property* for rows (columns) if its columns (rows) can be permuted in such a way that the 1's in each row occur in a circular consecutive order. The *augmented adjacency matrix* of a graph $G$ is obtained from the adjacency matrix of $G$ by adding 1's along the main diagonal.

It is helpful to visualize a concave-round enumeration of the vertices of $G$ by putting the vertices $v_1, v_2, \ldots, v_n$ clockwise around a circle. If $a = v_i$ and $b = v_j$ are two vertices, then we define the interval $[a, b]$ as the set $\{v_i, v_{i+1}, \ldots, v_j\}$, with the subscription calculated modulo $n$. Thus in our visualized representation, the interval extends from $a$ to $b$ clockwise. Similarly, we define $(a, b) = [a, b] - \{a, b\}$,

$[a, b) = [a, b] - \{b\}$, and $(a, b] = [a, b] - \{a\}$. It is easy to see from the remark above that a graph is concave-round if and only if its augmented adjacency matrix has the circular 1's property for columns (and rows). Similarly, a graph is convex-round if and only if its adjacency matrix has the circular 1's property for columns (and rows).

Let $G$ be a convex-round graph with a convex-round enumeration $\mathcal{L} = v_1, v_2, \ldots, v_n$. Suppose that $G$ is bipartite. Then $G$ is said to be *bipartite with respect to* $\mathcal{L}$ if there exist $i$ and $j$ with $i \neq j$ such that every edge of $G$ has an endvertex in $A = [v_i, v_j)$ and an endvertex in $B = [v_j, v_i)$. We shall refer to $(A, B)$ as a *bipartition of* $G$ *with respect to* $\mathcal{L}$.

## 3. Structure of concave-round graphs and convex-round graphs.

In this section we shall describe several structural results on graphs which are concave-round or convex-round. These results will be used in the later sections. As we observed earlier, concave-rounds graphs form a superclass of the class of proper circular arc graphs. We will now show that they form a subclass of the class of circular arc graphs.

THEOREM 3.1. *If a graph is concave-round, then it is a circular arc graph.*

*Proof.* By a theorem of Tucker, see [13, Theorem 8.16, p. 190], a graph is a circular arc graph if its augmented adjacency matrix has the circular 1's property for columns. As we remarked earlier the augmented adjacency matrix of any concave-round graph has this property. □

THEOREM 3.2. *There is a linear time algorithm for recognizing concave-round (convex-round) graphs and finding (if it exists) a concave-round (convex-round) enumeration.*

*Proof.* Checking whether a graph is convex-round can be done using PQ-trees [6]: we are checking whether the adjacency matrix has the circular 1's property for columns. Instead of working with the adjacency matrix (whose size may not be $O(n + m)$ if $m$ is not large) we can perform the checking by just giving the adjacency lists of $G$ (i.e., entries which correspond to a one). For details on PQ-trees, see [6, 13]. The same argument shows that we can decide whether a graph is concave-round in time $O(n + m)$. A convex-round (concave-round) enumeration can be found from the resulting circular 1's ordering. □

A *tournament* is an orientation of a complete graph, i.e., an oriented graph in which every pair of distinct vertices are joined by an arc. A tournament is *transitive* if and only if it has no directed cycles. An oriented graph $D = (V, A)$ is a *local tournament* (*local transitive tournament*) if the set of out-neighbors as well as the set of in-neighbors of each vertex $v \in V$ forms a tournament (a transitive tournament) [2, 14, 16].

The following lemma can be found in [16]; see also [28].

LEMMA 3.3. *Let $G$ be an undirected graph; then each of the following are equivalent:*

1. *$G$ is a proper circular arc graph.*
2. *$G$ can be oriented as a local tournament digraph.*
3. *$G$ can be oriented as a local transitive tournament digraph.*

LEMMA 3.4. *Let $G$ be a concave-round graph with a concave-round enumeration $\mathcal{L} = v_1, v_2, \ldots, v_n$. Then $G$ is a proper circular arc graph if $G$ satisfies the following three properties:*

1. *Every vertex of $G$ has a nonneighbor.*
2. *$G$ has no pair of vertices $v_i, v_j$ such that $(v_i, v_j] \subseteq N(v_i)$ and $(v_j, v_i] \subseteq N(v_j)$.*
3. *$G$ has no pair of vertices $v_i, v_j$ such that $[v_j, v_i) \subseteq N(v_i)$ and $[v_i, v_j) \subseteq N(v_j)$.*

*Proof.* By the assumption above, for each vertex $v_i$ we can distinguish those neighbors of $v_i$ that are after $v_i$ and those that are before $v_i$, namely, $v_j$ is after (before) $v_i$ if and only if $v_i$ is adjacent to every vertex in $[v_{i+1}, v_j]$ ($[v_j, v_{i-1}]$). Now we orient $G$ by orienting an edge $v_iv_j$ from $v_i$ to $v_j$ if $v_j$ is after $v_i$ and from $v_j$ to $v_i$ otherwise. It follows from the assumption in the lemma that this orientation is well defined. Furthermore, if $v_i \rightarrow v_j$ is an arc then $v_i$ is adjacent to every vertex in $[v_{i+1}, v_j]$ and since $v_j$ has a nonneighbor in $[v_{j+1}, v_i]$ (by the assumption of the lemma), we also have that $v_j$ is adjacent to every vertex in $[v_i, v_{j-1}]$. This shows that the orientation is a local transitive tournament orientation of $G$. Hence, by Lemma 3.3, $G$ is a proper circular arc graph. □

In [16], the second author characterized all possible orientations of a proper circular arc graph as local tournaments. The following problem may be seen as an attempt to generalize that result.

PROBLEM 3.5. *Given a graph $G$ which is concave-round, what are the possible concave-round enumerations of $G$?*

We now turn our attention to convex-round graphs.

LEMMA 3.6. *Let $G = (A, B)$ be a connected bipartite graph. Suppose that $G$ is convex-round with a convex-round enumeration $\mathcal{L}$. Then $G$ is bipartite with respect to $\mathcal{L}$.*

*Proof.* Let $[x, y] \subseteq A$ be an interval (with respect to $\mathcal{L}$) which is maximal (in terms of cardinality of $[x, y]$). If $A \subseteq [x, y]$, then we are done; otherwise let $w$ be a vertex in $A - [x, y]$ and let $P = u_1u_2 \ldots u_l$ be a shortest path from $[x, y]$ to $w$. Then $u_2 \in B$, and we must have $N(u_2) \subseteq [x, y]$ since $u_2$ is adjacent to neither $x^-$ nor $y^+$ (here $x^-$ is the vertex "next" to $x$ in counterclockwise direction and $y^+$ is the vertex "next" to $y$ in clockwise direction). Therefore $u_3 \in [x, y]$, contradicting the minimality of $P$. □

LEMMA 3.7. *Let $G$ be a connected convex-round graph with a convex-round enumeration $\mathcal{L} = v_1, v_2, \ldots, v_n$. If $v_iv_{i+k} \notin E(G)$ for some $i$, then either $G$ is bipartite, or at least one of $[v_i, v_{i+k}]$, $[v_{i+k}, v_i]$ is independent. Moreover, it can be decided in time $O(n + m)$ which of the above properties holds for $G$.*

*Proof.* Since $v_iv_{i+k} \notin E(G)$ and $G$ is convex-round, we have $N(v_i) \subseteq (v_i, v_{i+k})$ or $N(v_i) \subseteq (v_{i+k}, v_i)$ and, similarly, $N(v_{i+k}) \subseteq (v_i, v_{i+k})$ or $N(v_{i+k}) \subseteq (v_{i+k}, v_i)$.

Suppose first that $N(v_i) \subseteq (v_i, v_{i+k})$ and $N(v_{i+k}) \subseteq (v_i, v_{i+k})$. We shall show that $[v_{i+k}, v_i]$ is independent. Assume that $[v_{i+k}, v_i]$ is not independent, i.e., there exists an edge between two vertices in $[v_{i+k}, v_i]$, say one of them is $x_1$. Clearly, $x_1$ can be chosen such that $x_1 \in (v_{i+k}, v_i)$. Let $P = x_1x_2 \ldots x_l$ be a shortest path from $x_1$ to $[v_i, v_{i+k}]$. Then $x_{l-1}$ is adjacent to $x_l \in (v_i, v_{i+k})$ and a vertex in $(v_{i+k}, v_i)$. Since $x_{l-1}$ is convex, we must either have $x_{l-1}v_i \in E(G)$ or $x_{l-1}v_{i+k} \in E(G)$. However, this contradicts the assumption that $N(v_i) \subseteq (v_i, v_{i+k})$ and $N(v_{i+k}) \subseteq (v_i, v_{i+k})$. Therefore $[v_{i+k}, v_i]$ is independent. Analogously, if $N(v_i) \subseteq (v_{i+k}, v_i)$ and $N(v_{i+k}) \subseteq (v_{i+k}, v_i)$, then we obtain that $[v_i, v_{i+k}]$ is independent.

Suppose now that $N(v_i) \subseteq (v_i, v_{i+k})$ and $N(v_{i+k}) \subseteq (v_{i+k}, v_i)$ (the case when $N(v_i) \subseteq (v_{i+k}, v_i)$ and $N(v_{i+k}) \subseteq (v_i, v_{i+k})$ can be handled analogously). Let $x \in [v_{i+k}, v_i)$ be arbitrary. Observe that $x$ cannot have edges to vertices in both $(x, v_i)$ and in $[v_i, x)$, as this would imply that $x$ is adjacent to $v_i$, a contradiction. Thus we can partition $[v_{i+k}, v_i)$ into three sets $A$, $L$, and $R$ as follows: $A$ consists of vertices $x$ which have a neighbor in $[v_i, v_{i+k})$, $L$ consists of vertices $x$ with $N(x) \subseteq (x, v_i)$, and $R$ consists of vertices $x$ with $N(x) \subseteq [v_{i+k}, x)$. Note that $L \neq \emptyset$ as it contains $v_{i+k}$. Since $G$ is connected, we have that $A \neq \emptyset$.

Let $a \in A$ be chosen so that $[v_{i+k}, a) \cap A = \emptyset$. We claim that $[a, v_i)$ must be independent. Assume that this is not the case and let $x_1 \in [a, v_i)$ be chosen such that $x_1$ has an edge to a vertex in $(x_1, v_i)$. Let $P = x_1 x_2 \ldots x_l$ be a shortest path from $x_1$ to $[v_i, a)$. Clearly, $N(x_{l-1}) \cap (x_{l-1}, v_i) = \emptyset$, as otherwise $x_{l-1}$ would be adjacent to $v_i$, a contradiction. This implies that $l \geq 3$ and $x_{l-1} \notin L$. If $x_{l-2} \in L$, then $x_{l-1}$ must be adjacent to $a$, since $x_{l-1}$ is convex, $x_{l-2} \in [a, x_{l-1})$, and $x_l \in [v_i, a)$. However, this implies that $a$ is adjacent to $v_i$, which is a contradiction. Hence $x_{l-2} \notin L$, which implies that $x_{l-2} \in R$ (the minimality of $l$ ensures $x_{l-2} \notin A$). However, this implies that $N(x_{l-1}) \cap (x_{l-1}, v_i) \neq \emptyset$, a contradiction. Therefore $[a, v_i)$ is independent.

Now suppose that $[v_{i+k}, a) \cap R \neq \emptyset$. Let $r \in [v_{i+k}, a) \cap R$ be chosen such that $[v_{i+k}, r) \subseteq L$. We will show that $[r, a) \subseteq R$. If this is not true, then there is a vertex $l \in [r, a) \cap L$ such that $[r, l) \subseteq R$. Let $P = x_1 x_2 \ldots x_m$ be a shortest path from $r = x_1$ to $[l, v_{i+k})$. As $[v_{i+k}, a) \cap A = \emptyset$ we must have $x_m \in [l, v_i)$, $x_{m-1} \in L$, and $x_{m-2} \in R$. However this implies that $x_{m-1}$ and $l$ is adjacent, a contradiction against $l \in L$. Therefore $[r, a) \subseteq R$.

Let $w \in L \cap [v_{i+k}, a)$ be chosen such that $(w, a) \cap L = \emptyset$. The choice of $w$, together with the fact that $[a, v_i)$ is independent, implies that $(w, v_i)$ is independent. Using the above arguments, we see that $[v_{i+k}, w]$ is also independent and, furthermore, the vertices in $[v_{i+k}, w]$ can have edges only into $(w, v_i)$.

Repeating all the above arguments for $[v_i, v_{i+k})$, we can analogously obtain a vertex $w'$ (in $[v_i, v_{i+k})$), such that both $(w', v_{i+k})$ $[v_i, w']$ are independent and vertices in $[v_i, w']$ can have edges only into $(w', v_{i+k})$. This implies that both $(w, w']$ and $(w', w]$ are independent, which proves $G$ is bipartite. ∎

By inspecting the proof above, we see that the following holds.

Corollary 3.8. *Let $G$ be a connected convex-round graph with a convex-round enumeration $\mathcal{L} = v_1, v_2, \ldots, v_n$. Suppose that $v_i v_{i+k} \notin E(G)$. Then the following statements hold:*

1. *If $N(v_i) \subseteq (v_i, v_{i+k})$ and $N(v_{i+k}) \subseteq (v_{i+k}, v_i)$, then $G$ is bipartite.*
2. *If $N(v_i) \subseteq (v_{i+k}, v_i)$ and $N(v_{i+k}) \subseteq (v_i, v_{i+k})$, then $G$ is bipartite.*
3. *If $N(v_i) \subseteq (v_i, v_{i+k})$ and $N(v_{i+k}) \subseteq (v_i, v_{i+k})$, then $[v_{i+k}, v_i]$ is independent.*
4. *If $N(v_i) \subseteq (v_{i+k}, v_i)$ and $N(v_{i+k}) \subseteq (v_{i+k}, v_i)$, then $[v_i, v_{i+k}]$ is independent.*

The following lemma is easy to prove; we leave the details to the reader.

Lemma 3.9. *Let $C = v_1 v_2 \ldots v_n v_1$ be a chordless cycle with $n \geq 5$.*

- *If $n = 2k$ with $k \geq 3$, then (up to cyclic permutations and full reversal) there is a unique concave-round enumeration $\mathcal{L}$ of $C$, namely, $\mathcal{L} = v_1, v_2, \ldots, v_{2k}$. Furthermore, there is no convex-round enumeration of $C$.*
- *If $n = 2k + 1$ with $k \geq 1$, then (up to cyclic permutations and full reversal) there is precisely one concave-round enumeration of $C$, namely, $\mathcal{L}_1 = v_1, v_2, \ldots, v_{2k+1}$. Furthermore, there is precisely one convex-round enumeration of $C$, namely, $\mathcal{L}_2 = v_1, v_3, \ldots, v_{2k+1}, v_2, v_4, \ldots, v_{2k}$.*

Corollary 3.10. *If a convex-round digraph is not bipartite, then it is connected.*

*Proof.* Suppose that $G$ is convex-round with a convex-round enumeration $\mathcal{L}$. Assume that $G$ contains an odd cycle $C = v_1 v_2 \ldots v_{2k+1}$ ($k \geq 1$). By Lemma 3.9, the vertices of $C$ must occur in the order $v_1, v_3, \ldots, v_{2k+1}, v_2, v_4, \ldots, v_{2k}$ in $\mathcal{L}$. Let $v$ be a vertex not on $C$. Then $v$ must be between $v_i$ and $v_{i+2}$ for some $i \leq 2k + 1$ (the addition is modulo $2k + 1$). Since $\mathcal{L}$ is a convex-round enumeration, we see that $v$ is adjacent to $v_{i+1}$. Hence every vertex of $G$ is adjacent to at least one vertex of $C$ and therefore $G$ is connected. ∎

LEMMA 3.11. *If $G$ is concave-round and $\overline{G}$ is not bipartite, then $G$ is a proper circular arc graph.*

*Proof.* Let $\mathcal{L} = v_1, v_2, \ldots, v_n$ be a concave-round enumeration of $G$ and let $C$ be an arbitrary odd cycle in $\overline{G}$. Since $C$ is odd it is easy to check that for every edge $v_iv_j$ in $G$, there is some edge $v_kv_r$ in $C$ such that $v_k, v_r$ are either both in $(v_i, v_j)$, or both in $(v_j, v_i)$. Suppose, without loss of generality, that $v_k, v_r$ are both in $(v_j, v_i)$ and that, according to $\mathcal{L}$, the vertices come in the order $v_j, v_k, v_r, v_i$. Now apply Lemma 3.7 to $\overline{G}$ and the edge $v_iv_j$. It follows from the lemma that $[v_i, v_j]$ is an independent set in $\overline{G}$ and hence a clique in $G$.

By Corollary 3.10, every vertex of $G$ has a nonneighbor. Above we observed that for every edge $v_iv_j$ in $G$, either $[v_i, v_j]$, or $[v_j, v_i]$ is a clique. Combining these two facts we see that for every edge $v_iv_j$ in $G$, either $v_i, v_j$ both have a nonneighbor in $[v_i, v_j]$, or they both have a nonneighbor in $[v_j, v_i]$. It follows that $G$ satisfies 1–3 in Lemma 3.4 and hence $G$ is a proper circular arc graph. ∎

## 4. Hamiltonian cycles in convex-round graphs.

LEMMA 4.1. *The Hamiltonian cycle problem is solvable in linear time for bipartite graphs which are convex-round.*

*Proof.* Since this follows from a more general result for circular convex bipartite graphs (using a reduction to circular arc graphs) [20], we give here only the main idea which is to reduce the problem to the same problem for interval graphs.

Let $G$ be a connected bipartite graph which is convex-round with a convex-round enumeration $\mathcal{L} = v_1, v_2, \ldots, v_n$. By Lemma 3.6, there is a bipartition $(A, B)$ of $G$ with respect to $\mathcal{L}$. We have $A = [v_i, v_j)$ and $B = [v_j, v_i)$ for some $i$ and $j$. We assume that $|A| = |B|$ as otherwise $G$ has no Hamiltonian cycles. Let $G'$ be a graph obtained from $G$ by adding edges $xy$ if $x, y \in A$ and $N(x) \cap N(y) \neq \emptyset$. The new graph $G'$ is an interval graph: the interval family which represents $G'$ consists of one interval containing just one point for each $v_k \in B = [v_j, v_i)$, and one interval which represents the neighborhood of $v_s$ for each $v_s \in A = [v_i, v_j)$. It is easy to see that $G$ is Hamiltonian if and only if $G'$ is. Note that, given $G$ and a convex-round enumeration $\mathcal{L}$ of $G$, we can construct in time $O(n + m)$ the interval representation of $G'$ (here $n$ and $m$ are, respectively, the numbers of vertices and edges of $G$). Thus the claim follows from the existence of an $O(n)$ algorithm for finding a Hamiltonian cycle (if one exists) in an interval graph with a given interval representation [18]. ∎

THEOREM 4.2. *The Hamiltonian cycle problem is solvable in time $O(n + m)$ for convex-round graphs.*

*Proof.* Let $\mathcal{L} = v_1, v_2, \ldots, v_n$ be a convex-round enumeration of $G$. By Corollary 3.2, such an enumeration can be found in time $O(n + m)$. If $G$ is not connected, then $G$ is not Hamiltonian. So assume that $G$ is connected.

If $n$ is odd, then do the following. Let $k = \frac{1}{2}(n - 1)$. If $v_iv_{i+k} \in E(G)$ for all $v_i \in V(G)$, then the subgraph containing precisely the edge set $\{v_iv_{i+k} : i = 1, 2, \ldots, 2k + 1\}$ is a Hamiltonian cycle. If there is some $v_i \in V(G)$ such that $v_iv_{i+k} \notin E(G)$, then Lemma 3.7 implies that either $\alpha(G) \geq k + 1 = \frac{1}{2}(n + 1)$ or $G$ is bipartite, which in both cases implies that $G$ is not Hamiltonian.

If $n$ is even, then do the following. Let $k = \frac{1}{2}n$. If there is some $v_i \in V(G)$ such that $v_iv_{i+k-1} \notin E(G)$, then Lemma 3.7 implies that either $[v_i, v_{i+k-1}]$ is independent, or $[v_{i+k-1}, v_i]$ is independent, or $G$ is bipartite with respect to $\mathcal{L}$. If $[v_{i+k-1}, v_i]$ is independent, then $G$ is not Hamiltonian, because $\alpha(G) > \frac{1}{2}n$. If $[v_i, v_{i+k-1}]$ is independent, then by deleting all edges in $G$ with both endpoints in $(v_{i+k-1}, v_i)$ we obtain a new convex-round graph, $G'$. Now $G'$ is bipartite with respect to $\mathcal{L}$ and, furthermore, $G'$ is Hamiltonian if and only if $G$ is Hamiltonian. By Lemma 4.1, we can decide whether $G'$ is Hamiltonian in time $O(n+m)$, and thereby also whether $G$ is Hamiltonian in time $O(n+m)$. If $G$ is bipartite with respect to $\mathcal{L}$, then by Lemma 4.1 we can decide whether it is Hamiltonian in time $O(n+m)$. Therefore we may assume that $v_i v_{i+k-1} \in E(G)$ for all $v_i \in V(G)$. As $G$ is convex-round, we must have $\{v_{i+k-1}, v_{i+k}, v_{i+k+1}\} \subseteq N(v_i)$. Now we see that $G$ is Hamiltonian: If $k$ is even, then $G$ contains the following Hamiltonian cycle:

$$v_1 v_{k+2} v_3 v_{k+4} \dots v_{2k} v_k v_{2k-1} v_{k-2} v_{2k-3} \dots v_{k+1} v_1.$$

If $k$ is odd, then $G$ contains the following Hamiltonian cycle:

$$v_1 v_{k+2} v_3 v_{k+4} v_5 v_{k+6} \dots v_k v_{2k} v_{k-1} v_{2k-2} v_{k-3} v_{2k-4} \dots v_2 v_{k+1} v_1. \qquad \square$$

Note that there exist highly connected convex-round graphs which contain a factor (a spanning collection of disjoint cycles) but not a Hamiltonian cycle. Such a graph can be obtained from the graph $G = (\{a,b,c,d\}, \{ab, bc, bd, ca\})$ by substituting for each vertex an independent set of size $k > 1$.

**5. Maximum matchings in convex-round graphs.** The following lemma is due to Glover [12]. For the sake of completeness and since the paper [12] may not be easily accessible, we give a short outline of Glover's (greedy) algorithm.

LEMMA 5.1. *The maximum matching problem is solvable in time $O(n+m)$ for bipartite graphs which are convex-round.* $\square$

*Description of Glover's algorithm.* Let $G = (A, B, E)$ be a bipartite graph. We may assume that $G$ is connected, as otherwise we consider the components of $G$ separately. According to Lemma 3.6, $G$ has a convex-round enumeration $\mathcal{L} = a_1, \dots a_s, b_1, \dots, b_t$ where $A$ consists of the first $s$ vertices and $B$ consists of the next $t$ vertices. Such an enumeration can be found in time $O(n+m)$ ensured by Theorem 3.2. Now sort the vertices of $A$ according to their highest numbered neighbor in $B$. The sorting is done in increasing order. Let $a'_1, \dots, a'_s$ denote this order. This takes time $O(n+m)$ since we are sorting numbers in the range $1, 2, \dots, t$ (see [25, pp. 127–128]).

Perform the following greedy algorithm $\mathcal{A}$:

1. Let $M = \emptyset$ and let $i = 1$.
2. If $a'_i$ has a neighbor that is not yet matched by $M$, then let $b_j$ be the smallest numbered nonmatched neighbor of $a'_i$ and let $M := M + a'_i b_j$.
3. If $i < s$, then let $i := i + 1$ and goto 2.

Glover proved that the matching $M$, produced by $\mathcal{A}$, is a maximum matching of $G$. The algorithm $\mathcal{A}$ can be performed in time $O(n+m)$ and hence the total time to find $M$ is $O(n+m)$ as claimed.

Let $G$ be a convex-round graph with convex-round enumeration $\mathcal{L} = v_1, v_2, \dots, v_n$. An interval $[v_i, v_j]$ is called *potentially independent* if either $N(v_i) \cap [v_i, v_j] = \emptyset$ or $N(v_j) \cap [v_i, v_j] = \emptyset$. In general, a largest potentially independent interval can be found as follows: Treat the convex-round enumeration $\mathcal{L}$ as a circular order. For each vertex $x$, find the furthest vertex $y$ in clockwise order with $N(x) \cap [x, y] = \emptyset$ and the furthest vertex $z$ in counterclockwise order with $N(x) \cap [z, x] = \emptyset$. Then both $[x, y]$ and $[z, x]$ are potentially independent intervals. Compare the size of all these intervals to obtain a largest potentially independent interval. It is clear that these procedures can be conducted in time $O(n+m)$.

THEOREM 5.2. *The maximum matching problem is solvable in time $O(n+m)$ for convex-round graphs.*

*Proof.* Let $G$ be a convex-round graph with a convex-round enumeration $\mathcal{L} = v_1, v_2, \ldots, v_n$. We may assume that $G$ is connected, as otherwise we consider the components of $G$ separately. We claim that the following algorithm will find a maximum matching in $G$ in time $O(n+m)$.

Algorithm $\mathcal{B}$:

1. If $v_i v_{i+\lfloor n/2 \rfloor} \in E(G)$ for all $i = 1, 2, \ldots, \lfloor n/2 \rfloor$, then these edges form a matching of size $\lfloor n/2 \rfloor$ which clearly is maximum. Otherwise find a largest potentially independent interval in $\mathcal{L}$. Without loss of generality, let the interval be $[v_1, v_l]$ for some $l \geq 1$. (Since $v_i v_{i+\lfloor n/2 \rfloor} \notin E(G)$ for some $i$, either $N(v_i) \cap [v_i, v_{i+k}] = \emptyset$ or $N(v_i) \cap [v_{i+k}, v_i] = \emptyset$. This means that either $[v_i, v_{i+\lfloor n/2 \rfloor}]$ or $[v_{i+\lfloor n/2 \rfloor}, v_i]$ is a potentially independent interval. Hence a largest potentially independent interval must contain at least $\lfloor n/2 \rfloor$ vertices and, in particular, $l \geq \lfloor n/2 \rfloor$.)
2. If $G$ is bipartite, then use Lemma 5.1 to find a maximum matching in $G$ and return this.
3. If $G$ is not bipartite, then let $G'$ be the bipartite graph obtained from $G$ by deleting all edges in $[v_{l+1}, v_n]$. (We will show this is bipartite below.) Now find a maximum matching in $G'$ using Lemma 5.1 and return this matching.

The complexity of the algorithm is $O(n+m)$ by Lemma 5.1 and the remarks in step 1 of $\mathcal{B}$. To prove it is correct, it is enough to show that, when $G$ is not bipartite, the graph $G'$ defined in step 3 is in fact bipartite and convex-round and, furthermore, the size of a maximum matching in $G$ equals the size of a maximum matching in $G'$. So we assume below that $G$ is not bipartite.

It is easy to see that $G'$ is convex-round. Since $G$ is connected, either $v_1$ or $v_l$ has an edge into $[v_l, v_1]$ and thus $[v_l, v_1]$ is not independent. However, since $G$ is not bipartite, Lemma 3.7 now ensures that $[v_1, v_l]$ must be independent, which implies that $G'$ (in step 3) is bipartite.

It remains to show that the size of a maximum matching in $G$ equals the size of a maximum matching in $G'$. To do this, we shall construct a maximum matching in $G$, which is also a matching in $G'$. Define

$$\begin{aligned} j = &\quad \max\{j \ : \ j \leq l \text{ and } v_{l+i} v_i \in E(G) \text{ for all } i = 1, 2, \ldots, j-1\}, \\ k = &\quad \max\{k \ : \ k \leq l \text{ and } v_{n-i} v_{l-i} \in E(G) \text{ for all } i = 0, 1, \ldots, k-1\}. \end{aligned}$$

If $l + j \geq n - k + 1$ (i.e., $k - 1 \geq n - l - j$), then we obtain the following matching of size $n - l$, which is in both $G$ and $G'$:

$$\{v_1 v_{l+1}, v_2 v_{l+2}, \ldots, v_{j-1} v_{l+(j-1)}, v_l v_n, v_{l-1} v_{n-1}, v_{l-2} v_{n-2}, \ldots, v_{l-(n-l-j)} v_{n-(n-l-j)}\}.$$

Since $[v_1, v_l]$ is independent, a matching of $G$ contains at most $n - l$ edges and thus the above matching is maximum in $G$. So assume that $l + j \leq n - k$. We will now make some useful observations.

(i) For all $v_r \in [v_{l+1}, v_n]$, we have $N(v_r) \cap [v_1, v_l] \neq \emptyset$: Assume $N(v_r) \cap [v_1, v_l] = \emptyset$ for some $v_r \in [v_{l+1}, v_n]$. If $N(v_r) \subseteq [v_r, v_n]$, then $[v_1, v_r]$ is a potentially independent interval of size greater than that of $[v_1, v_l]$, a contradiction. If $N(v_r) \subseteq [v_{l+1}, v_r]$, then $[v_r, v_l]$ is a potentially independent interval of size greater than that of $[v_1, v_l]$, a contradiction.

(ii) $j \geq 2$ and $k \geq 1$: As $G$ is not bipartite, there exists an edge $v_a v_b$ in $G$ with $l + 1 \leq a < b \leq n$. By (i), we have that $v_a v_n \in E(G)$ and $v_b v_{l+1} \in E(G)$. Using (i) again, we get that $v_{l+1} v_1 \in E(G)$ and $v_n v_l \in E(G)$. Therefore $j \geq 2$ and $k \geq 1$.

(iii) $N(v_{l+j}) \cap [v_1, v_j] = \emptyset$ and $N(v_{n-k}) \cap [v_{l-k}, v_l] = \emptyset$: The definition of $j$ implies that $v_j v_{l+j} \notin E(G)$. If $N(v_{l+j}) \cap [v_1, v_j] \neq \emptyset$, then $N(v_{l+j}) \cap [v_j, v_{l+j}] = \emptyset$ (as $G$ is convex-round). However, this implies that $[v_j, v_{l+j}]$ is a potentially independent interval of size $l+1$, a contradiction. An analogous argument shows that $N(v_{n-k}) \cap [v_{l-k}, v_l] = \emptyset$.

(iv) $N(v_q) \cap ([v_1, v_{j-1}] \cup [v_{l-k+1}, v_l]) = \emptyset$ for all $v_q \in [v_{l+j}, v_{n-k}]$: Assume that $N(v_q) \cap [v_1, v_{j-1}] \neq \emptyset$ for some $v_q \in [v_{l+j}, v_{n-k}]$ (the case when $N(v_q) \cap [v_{l-k+1}, v_l] \neq \emptyset$ can be handled analogously). If $v_q v_{j-1} \in E(G)$, then $v_{j-1} v_{l+j} \in E(G)$ (as $v_{j-1} v_{l+j-1} \in E(G)$) which contradicts (iii). Therefore $v_q v_{j-1} \notin E(G)$, which implies that $[v_{j-1}, v_q]$ is a potentially independent interval of size $q - (j-1) + 1 \geq (l+j) - j + 2 = l + 2$, a contradiction.

Combining (i), (ii), and (iv), we have that $N(v_q) \subseteq [v_j, v_{l-k}]$ for all $v_q \in [v_{l+j}, v_{n-k}]$ (in particular, $[v_{l+j}, v_{n-k}]$ is independent). Now let $M'$ be a maximum matching in the bipartite graph $G'' = G\langle [v_{l+j}, v_{n-k}] \cup [v_j, v_{l-k}] \rangle$. It is clear that any matching in $G$ can have at most $|M'|$ edges with an endpoint in $[v_{l+j}, v_{n-k}]$ and at most $(j-1)+k$ edges with one endpoint in $[v_{l+1}, v_{l+j-1}] \cup [v_{n-k+1}, v_n]$. Since $[v_1, v_l]$ is independent, the following edges form a maximum matching in $G$ (and $G'$):

$$M' \cup \{v_1 v_{l+1}, v_2 v_{l+2}, \ldots, v_{j-1} v_{l+(j-1)}, v_l v_n, v_{l-1} v_{n-1}, \ldots, v_{l-k+1} v_{n-k+1}\}. \qquad \square$$

**6. Maximum cliques in convex-round graphs.** In this section, we shall present a linear time algorithm for solving the maximum clique problem for convex-round graphs. We observe that an $O(n^2)$ algorithm may be obtained by adapting the algorithm in [15], for solving the maximum independent set problem for circular arc graphs. (Although the algorithm in [15] is linear, we obtain only an $O(n^2)$ algorithm, as we have to take the complement of the input graph.) The algorithm below is more efficient and in fact it applies for a larger class of graphs, which is of independent interest.

We say that a graph $G$ is *interval enumerable* if there exists a linear enumeration $\mathcal{I} = v_1, v_2, \ldots, v_n$ of $V(G)$ such that, for each vertex $v_i$, there exists numbers $1 \leq \ell_i,\ h_i \leq i-1$ such that $N(v_i) \cap\ [v_1, v_{i-1}] = [v_{\ell_i}, v_{h_i}]$, where $\ell_i > h_i$ is used to indicate that $v_i$ has no neighbor $v_i$ with a label less than $i$. That is, we require that for each $v_i$ the neighbors with labels less than $i$ form an interval. (Note that there is no restriction on the neighbors of $v_i$ with index higher than $i$.) We shall refer to the enumeration $\mathcal{I}$ as an *interval enumeration* of $G$.

THEOREM 6.1. *The maximum clique problem is solvable in time $O(n+m)$ for interval enumerable graphs, provided that we are given an interval enumeration of the input graph.*

*Proof.* Given an interval enumeration $\mathcal{I} = v_1, \ldots, v_n$ of $G$, we can find all the numbers $\ell_1, \ldots, \ell_n, h_1, \ldots, h_n$ in time $O(n+m)$ just by scanning the neighbors of each vertex once. So assume below that all the numbers $\ell_1, \ldots, \ell_n, h_1, \ldots, h_n$ have been computed.

Now execute the following algorithm $\mathcal{C}$:

```
1. For $i = 1, \ldots, n$ do begin
2.         $C(i) := \{v_i\}$;
3.         $s(i) := 1$;
4.         $p(i) := i$;
5. end;
6. For $i := 1$ to $n$ do
7.         For $j := \ell_i$ to $h_i$ do
8.                 If $p(j) \leq h_i$ then begin
```

```
 9.                 $C(j) := C(j) \cup \{v_i\}$;
10.                 $p(j) := i$;
11.                 $s(j) := s(j) + 1$
12.             end;
13. $t := 0$; (* $t$ denotes the size of a maximum clique found so far *)
14. For $i := 1$ to $n$ do
15.         If $s(i) > t$ then begin
16.             $t := s(i)$;
17.             $q := i$
18.         end;
```

It should be clear from the description of the algorithm above that $C(i)$ is the set of vertices of a clique whose lowest (highest) labeled vertex is $v_i$ ($v_{p(i)}$) and whose size is $s(i)$.

We claim that the algorithm $\mathcal{C}$ returns a maximum clique, namely, the one starting at the vertex $v_q$ and whose vertices are stored in the set $C(q)$. In fact, we claim that the clique represented by $C(q)$ is the lexicographically smallest maximum clique[1] according to the ordering of $V(G)$. Suppose this is not the case and let $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$, $1 \leq i_1 < \cdots < i_k \leq n$, denote the vertices of the lexicographically smallest maximum clique $\Omega$ of $G$. Then $C(q) \neq \Omega$. It is clear from lines 14–18 that the clique $C(q)$ returned by $\mathcal{C}$ has the lowest numbered first vertex $v_q$ among all the $C(i)$ of the same size as $C(q)$. Consider $C(i_1)$ which is one of the possible choices for $\mathcal{C}$ in steps 14–18. By definition, $v_{i_1}$ belongs to $C(i_1)$ and by the assumption that $C(q) \neq \Omega$, there is some $j \leq k$ so that $v_{i_j}$ does not belong to $C(i_1)$, but each of the vertices $v_{i_1}, \ldots, v_{i_{j-1}}$ belong to $C(i_1)$.

Now, the reason $v_{i_j}$ could not be added to $C(i_1)$ in step 9 above must have been that at this point there was already another vertex $v_b$ with $i_{j-1} < b < i_j$ that had been added to $C(i_1)$ after $v_{i_{j-1}}$ with the property that $v_{i_j}$ and $v_b$ are not adjacent. On the other hand, since every $v_{i_r}$, $j < r \leq k$ is adjacent to $v_{i_{j-1}}$ and to $v_{i_j}$, it follows that $v_{i_r}$ is also adjacent to $v_b$. However, this contradicts the fact that $\Omega$ is lexicographically smallest, since $v_b$ is adjacent to each of $v_{i_1}, \ldots, v_{i_{j-1}}$ (implying that $\{v_{i_1}, \ldots, v_{i_{j-1}}, v_b, v_{i_{j+1}}, \ldots, v_{i_k}\}$ is also a clique of the same size as $\Omega$). Hence, we have shown that $C(i_1)$ is precisely the clique $\Omega$ (it cannot contain more vertices since $\Omega$ is maximum) and by the previous remark this is exactly the clique returned by $\mathcal{C}$.

The complexity claim follows from the description of $\mathcal{C}$. Note that the total work of the loop 6–12 is $O(n + m)$, since $v_i$ is adjacent to all vertices in the interval $[v_{\ell_i}, v_{h_i}]$ and hence scanning that interval in the loop 7–12 corresponds to scanning the neighbors of $v_i$ once. $\square$

If a graph is convex-round with convex-round enumeration $v_1, v_2, \ldots, v_n$, then the same enumeration shows that $G$ is interval enumerable. Thus the class of convex-round graphs form a subclass of the interval enumerable graphs. (The inclusion is proper, since the graph obtained from an induced cycle of length 5 by adding any number of new vertices adjacent to all 5 vertices on the cycle is interval enumerable but not convex-round.) Hence we have the following corollary.

COROLLARY 6.2. *The maximum clique problem is solvable in time $O(n + m)$ for convex-round graphs.*

We remark that the complement of an interval enumerable graph may not be a circular arc graph. An example of such a graph is the 5-cycle $C$ with two extra vertices joined to all 5 vertices on $C$.

[1] We say that an ordered set of vertices $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ is *lexicographically smaller* than another ordered set $\{v_{j_1}, v_{j_2}, \ldots, v_{j_r}\}$ if either $i_s = j_s$ for $s = 1, 2, \ldots, k$, or there exist $p$ such that $i_q = j_q$ for $1 \leq q \leq p - 1$ and $i_p < j_p$.

**7. Chromatic number of concave-round and convex-round graphs.** Recall that the *chromatic number* $\chi(G)$ is the least number of sets in a partition $V(G) = V_1 \cup \cdots \cup V_t$ such that each of the graphs $G\langle V_i \rangle$ has no edges.

Note that we can have that $\chi(G) > \omega(G)$ for convex-round graphs. The complement of any odd cycle is such an example and, in general, complements of powers of Hamiltonian cycles (on an odd number of vertices) provide such examples. For each of these examples we have $\chi(G) = \omega(G) + 1$. Through the correspondence between convex-round and concave-round graphs (by taking complements) and the fact that concave-round graphs are circular arc graphs, we obtain the following results. (Note again that the $O(n^2)$ algorithm is the best we can get because we need to consider the complement of the input graph.)

THEOREM 7.1 (see [15]). *The chromatic number of a convex-round graph can be found in time $O(n^2)$.*

Translating the proof of [15, Theorem 3.2] we get that the chromatic number of a convex-round graph is closely related to the size of a largest clique.

THEOREM 7.2 (see [15]). *For every convex-round graph $G$: $\chi(G) \leq \omega(G) + 1$.*

There exist concave-round graphs for which $\chi(G) = \lceil \frac{3}{2}\omega(G) \rceil$. Namely, $\chi(C_{3i-1}^{i-1}) = \lceil \frac{3i-1}{2} \rceil$ and $\omega(C_{3i-1}^{i-1}) = \lceil \frac{3i-1}{3} \rceil$. Here $C_k^r$ denotes the $r$th power of a $k$-cycle. It follows from this example that the bound in the following theorem is best possible even for concave-round graphs.

THEOREM 7.3 (see [17]). *For every circular arc graph $G$, $\chi(G) \leq \lceil \frac{3}{2}\omega(G) \rceil$.*

THEOREM 7.4. *For concave-round graphs, the minimum coloring problem is solvable in time $O(n^2)$.*

*Proof.* We can construct $\overline{G}$ in time $O(n^2)$ and then check in linear time whether $\overline{G}$ is bipartite. Suppose first that $\overline{G}$ is not bipartite. Using the linear algorithm from [9] for recognizing and representing proper circular arc graphs we obtain a representation of $G$ by inclusion-free circular arcs. Now we apply the $O(n^{1.5})$ algorithm of [26] for coloring a proper circular arc graph to find an optimal coloring of $G$.

Suppose now that $\overline{G}$ is bipartite with bipartition $(A, B)$, where $|A| \geq |B|$. It is not difficult to check that $\chi(G) = n - k$, where $k$ is the size of a maximum matching in $\overline{G}$ (recall that $A$ and $B$ induce cliques in $G$ and hence all vertices inside these sets must receive different colors). Second, if we are given a maximum matching $M$ of $\overline{G}$, then we can construct an optimal coloring with $n - |M|$ colors in linear time as follows: Color each vertex in $A$ with its own color. Let $M = a_1b_1, \ldots, a_kb_k$. Color $b_i$ with the color of $a_i$ for $i = 1, 2, \ldots, k$. Finally give each vertex in $B - \{b_1, \ldots, b_k\}$ a new color. By Lemma 5.1, we can find a maximum matching in $\overline{G}$ in linear time and hence the whole algorithm is $O(n^2)$. $\square$

Recall that the minimum coloring problem is NP-complete for general circular arc graphs [11] and polynomially solvable for proper circular arc graphs [3, 26].

## 8. Conjectures and open problems.

CONJECTURE 8.1. *For concave-round graphs, the maximum clique problem, the maximum matching problem, and the Hamiltonian cycle problem can all be solved in time $O(n + m)$.*

PROBLEM 8.2. *Find a characterization in terms of forbidden subgraphs of concave-round (convex-round) graphs.*

CONJECTURE 8.3. *Interval enumerable graphs can be recognized in polynomial time.*

CONJECTURE 8.4. *The Hamiltonian cycle problem is solvable in polynomial time for interval enumerable graphs.*

**Acknowledgments.** We would like to thank Stephan Brandt for stimulating discussions, in particular, for his insight which lead to the results on concave-round graphs in section 7. We would also like to thank the anonymous referees for several remarks that have improved on the presentation. The third author also wishes to acknowledge the hospitality of the Department of Mathematics and Statistics, University of Victoria, where part of this research was carried out.

## REFERENCES

[1] A. Appostolico and S.E. Hambrush, *Finding maximum cliques on circular arc graphs*, Inform. Process. Lett., 26 (1987), pp. 209–215.

[2] J. Bang-Jensen, *Locally semicomplete digraphs: A generalization of tournaments*, J. Graph Theory, 14 (1990), pp. 371–390.

[3] B. Bhattacharya, P. Hell, and J. Huang, *A linear algorithm for maximum weight cliques in proper circular arc graphs*, SIAM J. Discrete Math., 9 (1996), pp. 274–289.

[4] B.K. Bhattacharya and D. Kaller, *An $O(m + n \log n)$ algorithm for the maximum clique problem in circular arc graphs*, J. Algorithms, 35 (1997), pp. 336–358.

[5] J.A. Bondy and U.S.R. Murty, *Graph Theory and Applications*, American Elsevier, New York, 1976.

[6] K.S. Booth and G.S. Lueker, *Testing for the consecutive ones property using PQ-tree algorithm*, J. Comput. System Sci., 13 (1976), pp. 335–379.

[7] A. Czumaj, K. Diks, and T.M. Przytycka, *Parallel maximum independent set in convex bipartite graphs*, Inform. Process. Lett., 59 (1996), pp. 289–294.

[8] P. Damaschke, H. Müller, and D. Kratch, *Domination in convex and chordal bipartite graphs*, Inform. Process. Lett., 36 (1990), pp. 231–236.

[9] X. Deng, P. Hell, and J. Huang, *Linear time representation for proper circular-arc graphs and proper interval graphs*, SIAM J. Comput., 25 (1996), pp. 390–403.

[10] E.M. Eschen and J. Spinrad, *$O(n^2)$ recognition and isomorphism algorithms for circular arc graphs*, in Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Austin, TX, SIAM, Philadelphia, 1993, pp. 128–137.

[11] M.R. Garey, D.S. Johnson, G.L. Miller, and C.H. Papadimitriou, *The complexity of coloring circular arcs and chords*, SIAM J. Alg. Discrete Methods, 1 (1980), pp. 216–227.

[12] F. Glover, *Maximum matching in convex bipartite graphs*, Naval. Res. Logist., 14 (1967), pp. 313–316.

[13] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[14] P. Hell, J. Bang-Jensen, and J. Huang, *Local tournaments and proper circular arc graphs*, in Algorithms, Lecture Notes in Comput. Sci. 450, T. Asano, T. Ibaraki, H. Imai, and T. Nishizeki, eds., Springer, Berlin, 1990, pp. 101–108.

[15] W.L. Hsu and K.H. Tsai, *Linear time algorithms on circular arc graphs*, Inform. Process. Lett., 40 (1991), pp. 123–129.

[16] J. Huang, *On the structure of local tournaments*, J. Combin. Theory Ser. B, 63 (1995), pp. 200–221.

[17] I.A. Karapetjan, *Coloring of arc graphs*, Akad. Nauk Armyan. SSR Dokl., 70 (1980), pp. 306–311.

[18] J.M. Keil, *Finding Hamiltonian circuits in interval graphs*, Inform. Process. Lett., 20 (1985), pp. 201–206.

[19] N. Korte and R.H. Möhring, *An incremental linear-time algorithm for recognizing interval graphs*, SIAM J. Comput., 18 (1989), pp. 68–81.

[20] Y.D. Liang and N. Blum, *Circular convex bipartite graphs: Maximum matching and Hamiltonian circuits*, Inform. Process. Lett., 56 (1995), pp. 215–219.

[21] Y.D. Liang, G.K. Manacher, C. Rhee, and T.A. Mankus, *A linear algorithm for Hamiltonian circuits in circular arc graphs*, in Proceedings of the Southeast ACM Conference, Tuscaloosa, AL, 1994, pp. 15–22.

[22] Y.D. Liang and Maw-Shang Chang, *Minimum feedback vertex sets in cocomparability graphs and convex bipartite graphs*, Acta Inform., 34 (1997), pp. 337–346.

[23] Y.D. Liang and C. Rhee, *Finding a maximum matching in a circular-arc graph*, Inform. Process. Lett., 45 (1993), pp. 185–190.

[24] W. Lipski and F.P. Preparata, *Efficient algorithms for finding a maximum matching in convex bipartite graphs and related problems*, Acta Inform., 15 (1981), pp. 329–346.

[25] U. Manber, *Introduction to Algorithms*, Addison-Wesley, Reading, MA, 1989.

[26] W.K. Shih and W.L. Hsu, *An $O(n^{1.5})$ algorithm to color proper circular arcs*, Discrete Appl. Math., 25 (1989), pp. 321–323.

[27] W.K. Shih, T. C. Chern, and W.L. Hsu, *An $O(n^2 \log n)$ algorithm for the Hamiltonian cycle problem on circular-arc graphs*, SIAM J. Comput., 21 (1992), pp. 1026–1046.

[28] D.J. Skrien, *A relationship between triangulated graphs, comparability graphs, proper interval graphs, proper circular arc graphs, and nested interval graphs*, J. Graph Theory, 6 (1982), pp. 309–316.

[29] G. Steiner and J.S. Yeomans, *A linear time algorithm for maximum matchings in convex bipartite graphs*, Comput. Math. Appl., 31 (1996), pp. 91–96.

[30] K.E. Stouffers, *Scheduling of traffic lights–A new approach*, Transportation Res., 2 (1968), pp. 199–234.

[31] A. Tucker, *Matrix characterizations of circular-arc graphs*, Pacific J. Math., 39 (1971), pp. 535–545.

[32] C.W. Yu and G.H. Chen, *Efficient parallel algorithms for doubly convex-bipartite graphs*, Theoret. Comput. Sci., 147 (1995), pp. 249–265.

SIAM J. DISCRETE MATH.
Vol. 13, No. 2, pp. 194–201

# LATTICE-SIMPLEX COVERINGS AND THE 84-SHAPE*

ROD FORCADE† AND JACK LAMOREAUX†

**Abstract.** It is known that diameters of abelian Cayley graphs with $n$ generators are related to Manhattan diameters of a particular type of lattice tile called a Cayley tile and that determining the lower bound of those diameters is related to the density of lattice-simplex coverings of $R^n$. We construct a natural tile associated with each lattice-simplex covering and show that the 84-shape (discovered in [R. Dougherty and V. Faber, *The Degree-Diameter Problem for Several Varieties of Cayley Graphs*, http://www.c3.lanl.gov/dm/pub/laces.html (1994) and C.M. Fiduccia, J.S. Zito, and E. Mann, *Network Interconnection Architectures and Translational Tilings*, Tech. report, Center for Computing Science, Bowie, MD, 1994]) represents a local mininum of the density of lattice coverings of $R^3$ by a particular simplex $D$ (the convex hull of the unit basis vectors).

**Key words.** covering, diameter, lattice, simplex

**AMS subject classifications.** 05C12, 11H31, 52C17

**PII.** S0895480198349622

## 1. Introduction and preliminaries.

The problems of finding the thinnest covering density $\vartheta(C)$ by translates of an object $C$ and finding the thinnest covering density $\vartheta_L(C)$ by lattice of translates of $C$ have been studied extensively (see Conway and Sloane [1], Rogers [5] and Fejes Tóth and Kuperberg [6]), especially when $C$ is a sphere or simplex in $n$ dimensions. We concern ourselves with the case where the covering object is a simplex and its translates are given by a lattice. Note that if $S$ and $S_1$ are $n$-dimensional simplices in $R^n$, then $\vartheta_L(S) = \vartheta_L(S_1)$, since a linear transformation can be found to carry $S$ onto a translate of $S_1$.

The simplex covering density problem is related to diameters of abelian Cayley graphs as follows. Note that an abelian group $G$ with $n$ generators provides a homomorphism from $Z^n$ onto $G$. The kernel is a sublattice $L$ of $Z^n$; and the diameter $\delta$ of the Cayley graph is the Manhattan diameter of a tile $T$ ($T + L \doteq Z^n$) which is formed by choosing (in short-lex order, see [2] and [3]) a transversal of the cosets of $L$ in the first orthant. If $C = [0, 1)^n$ (the half-open $n$-cube) is added to $T$, the set $T' = T + C$ is a tile (with $L$) of $R^n$. Thus (as observed in [2]) its Manhattan diameter ($\lambda = \delta + n$) represents a dilation $\lambda D$ of the base simplex which covers $R^n$ with $L$ ($L + \lambda D = R^n$). So $\vartheta_L(S)$ for simplices $S \in R^n$ provides a lower bound ($\sqrt[n]{\vartheta_L(S)mn!} - n$) for the diameter of Cayley graphs of abelian groups with $m$ elements and $n$ generators. The diameter bound problem for abelian Cayley graphs is unsolved for dimensions greater than two but leads to some nice mathematics in three dimensions.

For our purposes, a *lattice* will be a discrete, rank-$n$ subgroup of $R^n$, and a *simplex* will be the convex hull of $n + 1$ points in general position.

## 2. A setting for the infimum covering density problem.

Let $L$ be a lattice and $S$ a simplex in $R^n$. Note that $L + S \supseteq R^n$ iff $S$ contains a *transversal*—a set of distinct representatives—of the cosets of $L$ in $R^n$. Such a transversal $T$ is a *tile* for the lattice $L$. In other words, every element of $R^n$ is contained in exactly one translate

*Received by the editors December 22, 1998; accepted for publication (in revised form) September 22, 1999; published electronically April 6, 2000.
http://www.siam.org/journals/sidma/13-2/34962.html

†Department of Mathematics, Brigham Young University, Provo, UT 84602 (forcader@math.byu.edu, jack@math.byu.edu).

$x+T$, $x \in L$. Equivalently, every element of $R^n$ is *uniquely* expressible as the sum of an element of $L$ plus an element of $T$, a situation commonly denoted by $L+T \doteq R^n$. Thus, for a given lattice $L$ and simplex $S$, $L+S \supseteq R^n$ iff $S$ contains a tile for the lattice $L$.

Our strategy will be to choose a fixed simplex, up to scaling, and to vary the lattice. Let $D$ be the *base simplex* defined by

$$D = \left\{ (x_1, x_2, \ldots, x_n) \in R^n \middle| x_i \geq 0 \ (\forall\ i),\ \sum_{i=1}^{n} x_i \leq 1 \right\}$$

with volume $\frac{1}{n!}$, and let

$$\lambda D = \left\{ (x_1, x_2, \ldots, x_n) \in R^n \middle| x_i \geq 0 \ (\forall\ i),\ \sum_{i=1}^{n} x_i \leq \lambda \right\}$$

with volume $\frac{\lambda^n}{n!}$ for any positive real number $\lambda$. For any lattice $L$, there is a minimum $\lambda$ such that $\lambda D$ contains a tile for $L$ (see the *subtraction construction* below). Call it $\lambda(L)$. Note that $\lambda$ is often referred to as the *Manhattan diameter* or *taxicab diameter* of the contained tile. The density of the covering is then $\theta(L) = \frac{(\lambda(L)^n/n!)}{Vol(L)}$. We seek to minimize $\theta(L)$ over all lattices. Since, for real nonzero $r$, $\theta_L = \theta(rL)$, it suffices to minimize $\theta(L)$ over lattices with $Vol(L) = 1$.

**3. The subtraction construction.** Given a lattice $L$, we will construct a tile which is contained in *all* of the dilations $\lambda D$ for which $L + \lambda D \supseteq R^n$. Our construction also proves that, given any fixed lattice $L$, there exists a minimum dilation $\lambda D$ such that $L + \lambda D \supset R^n$.

Since a tile is a transversal, it suffices to choose, from each coset of $L$, an element in the smallest possible $\lambda D$. We do that by constructing a well-ordering $\prec$ of the intersection of each coset with the first orthant and then choosing the $\prec$-least element of the coset. This will work if we ensure that $\prec$ is consistent with the ordering by scalings of $D$. In other words, if $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$, and $\sum x_i < \sum y_i$, then we must have $x \prec y$.

If $x$ and $y$ are in the same coset of $L$, we will define $x \prec y$ iff $y - x \in L_0$, where $L_0$ is a special subset of $L$. To make this consistent with scalings of $D$, we need $L_0$ to contain all elements of $L_1 = \{(x_1, \ldots, x_n) \in L \big| \sum x_i > 0\}$. To make $\prec$ a linear ordering, we need antisymmetry and transitivity. Respectively, we have the following.

(i) If $x \in L$ and $x \neq 0$ then exactly one of $\{x, -x\}$ is in $L_0$.

(ii) If $x, y \in L_0$ then $x + y \in L_0$.

If $L$ has no nonzero points on the hyperplane $H = \{(x_1, \ldots, x_n) \big| \sum x_i = 0\}$, then conditions (i) and (ii) are already satisfied by letting $L_0 = L_1$. If $(L \cap H) \setminus \{0\}$ is nonempty, intersect it with a half-space $U$ whose boundary intersects $L$ only at the origin—and join that intersection to $L_1$ to make $L_0 = L_1 \cup \big(((H \cap L) \setminus \{0\}) \cap U\big)$ satisfy the required conditions.

Now each coset of $L$ is linearly ordered by $\prec$, and the intersection of each coset with the first orthant is well ordered (because initial segments are bounded, thus finite). The set $T$ of $\prec$-least elements of those intersections is thus a tile contained in all dilations $\lambda D$ for which $L + \lambda D \supseteq R^n$. One may easily verify that

$$T = \mathcal{O}_n \ \setminus \ \bigcup_{x \in L_0} (x + \mathcal{O}_n)$$

(subtract from the first orthant $\mathcal{O}_n$, by set-complementation, all translates of the first orthant by elements of $L_0$). We call this the *subtraction construction* and observe that it effectively defines the smallest dilation of $D$ which will tile with a given lattice.

It is worth mentioning that this yields a *perfect* tile, in the sense that its lattice translates do not intersect each other, even on their boundaries. Note also that this construction can be applied to more general cones than $\mathcal{O}_n$, and to any subset $L_0 \subset L$ satisfying the conditions above, to yield tiles satisfying a variety of minimality conditions. In particular, we can similarly construct a tile for the minimum dilation of any given simplex which forms a covering with a given lattice.

Since the intersection of all $\lambda D$ containing $T$ is a dilation of $D$ containing $T$, we have also shown there is a *minimum* such dilation. The corresponding $\lambda(L)$ gives the density $\theta(L) = \frac{\lambda(L)^n}{n!\,Vol(L)}$ as noted before.

The minimum of $\theta(L)$ over all lattices is $\vartheta_L(D)$ (or $\vartheta_L(S)$ for any simplex $S$). The interesting implication is that there must exist a minimum simplex covering with $D$, hence a corresponding tile for each dimension which characterizes the minimum diameter in the abelian Cayley graph problem. What does it look like, one wonders. Is it unique? One expects it to be a simple object with a lot of symmetry, but our experience in three dimensions suggests that is not the case.

## 4. The two-dimensional case.

We explore the planar situation first, although the results in that case are well known (see [7] for example), because this case exposes the basic ideas.

The subtraction construction above produces a tile which is the relative complement, in the first quadrant, of a family of translates of the first quadrant. Since the tile is necessarily bounded (being contained in the smallest covering simplex), it only touches a finite number of $L$-translates of $T$. Thus the construction of $T$ can be effected by subtracting only a finite number of translates of the first quadrant from itself. In two dimensions, this means the tile must look like one of the following (with one, two, three, or more outer corners; see Figure 4.1).

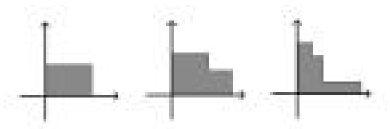


FIG. 4.1.

In other words, it must look like a rectangle at the origin with a finite number of rectangular "notches" taken out of it. It clearly cannot be a tile, however, if there is more than one notch (like the right-hand object in Figure 4.1), since the translates sitting in those notches would overlap. Thus we have only two shapes to consider, the rectangle and the "L-shape" (the first two shapes in Figure 4.1). It is easy to verify that all such shapes, with one or fewer notches, are tiles with appropriate lattices.

In each case, the density of the corresponding simplex covering can be calculated by dividing the area of the smallest $\lambda D$ containing the tile by the area of the tile itself (which is equal to the area of the lattice domains). Elementary calculus then determines that the minimum ratio is produced by an L-shape with its outer corners trisecting the outer edge of the enclosing simplex; see Figure 4.2.

Thus the minimum lattice simplex covering density in two dimensions is 1.5 (the area of the enclosing triangle divided by the area of the tile).

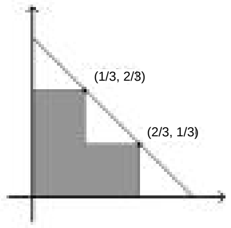


Fig. 4.2.

**5. The three-dimensional case.** The two-dimensional case is easy because we can characterize, independently of lattices, the possible tiles. There are only two combinatorial types. Unfortunately there are infinitely many combinatorial types of tiles in three dimensions. For example, if $n$ is a positive integer greater than one, let $M_n$ be the lattice generated by $(1,1,-1)$, $(1,1-n,n-1)$, and $(1-n,n,0)$. It is also the sublattice of $Z^3$ defined by the modular relation $(n+1)y+z \equiv nx$ (mod $n^2$). Using the subtraction construction (section 3 above) with this lattice yields the order-$n$ *double staircase* tile (described in [3, p. 166], discovered by Fiduccia, Zito, and Mann [4]). Its volume is $n^2$, it has $2n-1$ outer corners, and it fits inside the dilation $(n+2)D$ of the base simplex, yielding a covering density of in Figure 5.1.

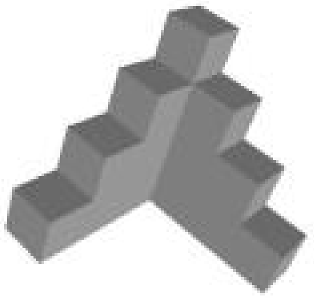


Fig. 5.1.

What does a minimum tile (characterizing the minimum simplex covering density) in three dimensions look like? How many corners does it have? How complicated is it? A possible candidate was discovered via computer search by Fiduccia, Zito, and Mann [4] and independently by Dougherty and Faber [2]. It is the so-called "84-shape," formed by applying the subtraction construction to the sublattice $L_{84}$ of $Z^3$ defined by the modular relation $2x+9y+35z \equiv 0$ (mod 84).

Denoted $T_{84}$ henceforth, it (see Figure 5.2) is a tile with volume 84 and is contained in $10D$ (its Manhattan diameter is 10), so it corresponds to a covering density of $\frac{(10^3/6)}{84} = (\,2-\frac{1}{63})$ (the smallest discovered so far). Its outer corners are at $(1,7,2)$, $(1,8,1)$, $(1,1,8)$, $(1,2,7)$, $(3,2,5)$, $(2,6,2)$, $(3,5,2)$, $(5,2,3)$, $(4,4,2)$, $(5,3,2)$, $(3,6,1)$, $(6,3,1)$, and $(8,1,1)$. Note that

$$T_{84} = \mathcal{O}_n \setminus \bigcup_{x\in S}(x+\mathcal{O}_n),$$

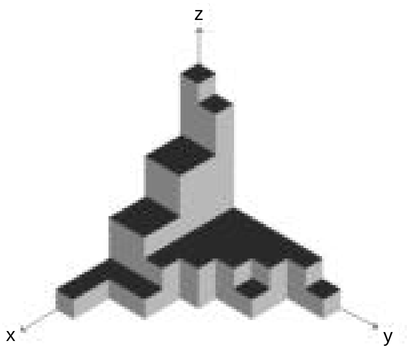


FIG. 5.2.

where $S = \{(x, y, z) \in L_{84} \mid x + y + z > 0\}$. (The nonzero elements of $L_{84}$ lying in the plane $x + y + z = 0$ prove to have no effect on the construction, so there is no need in this case to intersect those elements with a half-space $U$, as was done in the general construction.)

There are heuristic reasons for suspecting that $T_{84}$ *might* be the minimum characterizing tile for three dimensions, but no proof. First, despite extensive computer searches of finite abelian groups with three generators (the quotient groups for sublattices of $Z^3$), no lower density covering has been found. Second, the ones which come close look very similar to the 84-shape, suggesting that no simple, highly symmetric tile will do.

Third, we now show that the lattice associated with the 84-shape represents a *local minimum* in the covering-density function $\theta(L)$.

**6. The local minimum property of the 84-lattice.** Note that the 13 outer corners of $T_{84}$ represent critical points in the covering $L_{84} + 10D$, in the sense that $L_{84} + (10 - \epsilon)D$ will expose part of a neighborhood of each of those points (when $\epsilon > 0$). Thus any such corner must have $L_{84}$-coset-equivalent points (we'll call them *cousins* for brevity) on each of the other faces of $10D$. We list here the 13 corners and their respective cousins.

| | | | | |
|---|---|---|---|---|
| $(1,7,2)$ | : | $(0,1,6)$, | $(8,0,1)$, | $(3,5,0)$ |
| $(1,8,1)$ | : | $(0,2,5)$, | $(2,0,3)$, | $(8,1,0)$ |
| $(1,1,8)$ | : | $(0,2,3)$, | $(2,0,1)$, | $(6,3,0)$ |
| $(1,2,7)$ | : | $(0,3,2)$, | $(3,0,5)$, | $(2,1,0)$ |
| $(3,2,5)$ | : | $(0,5,2)$, | $(5,0,3)$, | $(2,3,0)$ |
| $(2,6,2)$ | : | $(0,1,1)$, | $(1,0,6)$, | $(4,4,0)$ |
| $(3,5,2)$ | : | $(0,1,8)$, | $(1,0,1)$, | $(5,3,0)$ |
| $(5,2,3)$ | : | $(0,7,2)$, | $(7,0,1)$, | $(2,5,0)$ |
| $(4,4,2)$ | : | $(0,1,3)$, | $(1,0,8)$, | $(6,2,0)$ |
| $(5,3,2)$ | : | $(0,8,1)$, | $(1,0,3)$, | $(7,1,0)$ |
| $(3,6,1)$ | : | $(0,2,7)$, | $(2,0,5)$, | $(1,1,0)$ |
| $(6,3,1)$ | : | $(0,2,4)$, | $(2,0,2)$, | $(1,8,0)$ |
| $(8,1,1)$ | : | $(0,2,6)$, | $(2,0,4)$, | $(3,6,0)$ |

The cousins were generated on a computer by searching for integer points on the coordinate planes having the same value of $2x+9y+35z$ (modulo 84) as the corresponding corner.

Note also that $L + \lambda D \supseteq R^n$ iff $(x + \lambda D) \cap L \neq \emptyset$ $(\forall\ x \in R^n)$, so the covering diameter is determined by the largest $\lambda D$ which can be translated to fit inside the lattice without any lattice points in its interior. Notice that any $\lambda D$, so translated, will necessarily have at least one lattice point on each of its faces. Thus the 13 corners above, with their cousins, when translated into the lattice, represent 13 four-point configurations in the lattice $L_{84}$ where a translate of $10D$ can fit neatly inside.

Our analysis of the lattice $L_{84}$, though somewhat complicated, is facilitated by the fact that each of those configurations is *simple* — consisting of exactly four points, one in the interior of each face. The analysis of other lattices is usually not so “simple.”

If we can show that every slight modification of $L_{84}$ preserving the lattice-volume causes at least one of those 13 configurations to *expand* (admitting a larger dilation of $D$), then we will know that $\theta(L)$ attains a local minimum on the lattice $L_{84}$. Note that modification of the lattice must be small enough to avoid introducing other lattice points into the simplices enclosed by the 13 configurations. That condition determines the neighborhood within which $\theta(L_{84})$ is minimum.

For the $i$th configuration $p_i : q_i, r_i, s_i$ above, we form a *configuration matrix* $C_i$ whose rows are elements of $L_{84}$, namely, $p_i - q_i$, $p_i - r_i$, and $p_i - s_i$, respectively. Since $q_i, r_i, s_i$ are on the $yz$-plane, the $xz$-plane and the $xy$-plane, respectively, the trace of each matrix represents the Manhattan diameter of the largest dilation of $D$ which will fit inside each corresponding configuration (10 in the present case). Thus the first two configurations above give rise to the matrices

$$C_1 = \begin{pmatrix} 1 & 6 & -4 \\ -7 & 7 & 1 \\ -2 & 2 & 2 \end{pmatrix},$$

$$C_2 = \begin{pmatrix} 1 & 6 & -4 \\ -1 & 8 & -2 \\ -7 & 7 & 1 \end{pmatrix},$$

etc.

If $M$ is an (arbitrary) basis matrix for the lattice (i.e., the rows of $M$ generate the lattice), we wish to consider the effect on the traces of the $C_i$ matrices when we vary the matrix $M$. We assume that $M$ is so arranged that its determinant is *positive* (not *negative*) 10. Let $L$ be a new lattice generated by $M(I + \epsilon K)$, where $I$ is the (3 by 3 identity matrix), $K$ is an arbitrary 3 by 3 matrix, and $\epsilon$ is a small positive real number. The new $i$th configuration matrix will be $C_i(I + \epsilon K)$. Its trace will be $\operatorname{tr}(C_i) + \epsilon \operatorname{tr}(C_i K)$, and the determinant of the new basis will be

$$\det(M) \cdot \det(I + \epsilon K) = \det(M)\bigl(1 + \epsilon \operatorname{tr}(K) + \epsilon^2 E_2(K) + \epsilon^3 \det(K)\bigr),$$

where $E_2(K)$ is the sum of all $2 \times 2$ principal minors of $K$.

Dividing the new trace by the cube-root of the new basis determinant and multiplying by the cube-root of the determinant of $M$ yields a function

$$f_i(\epsilon) = \frac{\operatorname{tr}(C_i) + \epsilon \operatorname{tr}(C_i K)}{\sqrt[3]{1 + \epsilon \operatorname{tr}(K) + \epsilon^2 E_2(K) + \epsilon^3 \det(K)}},$$

whose maximum over $i$ is proportional to the cube-root of the new covering density. Developing its Taylor series as a function of $\epsilon$ gives

$$f_i = \operatorname{tr}(C_i) + \epsilon\Big(\operatorname{tr}(C_iK) - \frac{1}{3}\operatorname{tr}(C_i)\operatorname{tr}(K)\Big) + \epsilon^2\left(\frac{2}{9}\operatorname{tr}(C_i)(\operatorname{tr}(K))^2 - \frac{1}{3}\operatorname{tr}(C_iK)\operatorname{tr}(K) - \frac{1}{3}\operatorname{tr}(C_i)E_2(K)\right) + \cdots$$

(we will not need the higher-order terms).

The linear coefficient $\operatorname{tr}(C_iK) - \frac{1}{3}\operatorname{tr}(C_i)\operatorname{tr}(K)$ can also be written $\big\langle (C_i - \frac{1}{3}\operatorname{tr}(C_i)I), K^\tau\big\rangle$, where $K^\tau$ is the transpose of $K$ and $\langle U, V\rangle$ denotes the standard inner product (sum of the products of corresponding entries) of matrices $U$ and $V$. By using symbolic manipulation software (we used Maple), one can check that the 13 *gradient* matrices $G_i = C_i - \frac{1}{3}\operatorname{tr}(C_i)I = C_i - \frac{10}{3}I$ lie in a seven-dimensional subspace of $R^9$. One can also check that

$$6G_1+2G_2+6G_3+G_4+47G_5+11G_6+12G_7+6G_8+11G_9+6G_{10}+6G_{11}+6G_{12}+6G_{13} = 0.$$

Thus within the seven-dimensional subspace, the origin lies in the interior of the convex hull of the $G_i$. Thus any nonzero matrix $K^\tau$ must have a positive scalar product with at least one of the $G_i$ or must have a zero scalar product with *all* of them. In the former case, one of the configurations will necessarily admit a simplex with a larger covering density (increasing the function $\theta(L)$).

In the latter case, we must examine the second-order term. Since we now assume $\operatorname{tr}(C_iK) = \frac{1}{3}\operatorname{tr}(C_i)\operatorname{tr}(K)$, $(\forall\ i)$, the second-order coefficient of $f_i$ becomes

$$\frac{1}{9}\operatorname{tr}(C_i)\big((\operatorname{tr}(K))^2 - 3E_2(K)\big)$$

$$= \frac{10}{9}\big((\operatorname{tr}(K))^2 - 3E_2(K)\big).$$

Again, using symbolic software (Maple), we find that the two-dimensional subspace of $K$'s such that $\langle G_i, K^\tau\rangle = 0$ $(\forall\ i)$ is generated by the identity matrix (corresponding to the fact that $\theta(\lambda L) = \theta(L)$) and the matrix

$$K_0 = \begin{pmatrix} 23 & -2 & 2 \\ 9 & -2 & 9 \\ 7 & 7 & 0 \end{pmatrix}.$$

We may describe a generic matrix $K$, having a zero scalar-product with all of the $G_i$'s by $K = \alpha K_0 + \beta I$. Then $\operatorname{tr}(K) = 21\alpha + 3\beta$ and $E_2(K) = -105\alpha^2 + 42\alpha\beta + 3\beta^2$, so the second-order term is

$$(21\alpha + 3\beta)^2 - 3(-105\alpha^2 + 42\alpha\beta + 3\beta^2) = 756\alpha^2.$$

This will be positive in all cases *except* when $K$ is a multiple of $I$ (corresponding once again to the fact that dilations of $L$ leave the covering density unchanged). Thus we have shown the following.

THEOREM 1. *The lattice $L_{84}$ associated with the 84-shape attains a local minimum in the covering-density function $\theta(L)$.*

CONJECTURE. *$\theta(L_{84}) = \vartheta_L(S)$, where $S$ is any full-dimensional simplex in $R^3$.*

As for that curious matrix $K_0$ above, it seems to correspond to a modification of $L_{84}$ which keeps the 13 corners at equal Manhattan distances from the origin, though not in the same positions relative to one another, while varying the lattice volume in such a way that the change in covering density is proportional to the sixth power of the distances which the corners move—until one or more of the configuration simplices are penetrated by other lattice points. Is this a characteristic phenomenon associated with a local minimum or perhaps with a global minimum?

## REFERENCES

[1] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, Springer-Verlag, New York, 1993.

[2] R. Dougherty and V. Faber, *The Degree-Diameter Problem for Several Varieties of Cayley Graphs* I: *The Abelian Case*, http://www.c3.lanl.gov/dm/pub/laces.html (1994).

[3] C. M. Fiduccia, R. W. Forcade, and J. S. Zito, *Geometry and diameter bounds of directed Cayley graphs of Abelian groups*, SIAM J. Discrete Math., 11 (1998), pp. 157–167.

[4] C. M. Fiduccia, J. S. Zito, and E. Mann, *Network Interconnection Architectures and Translational Tilings*, Tech. report, Center for Computing Science, Bowie, MD, 1994.

[5] C. A. Rogers, *Packing and Covering*, Cambridge Tracts in Math. and Mathematical Physics 54, Cambridge University Press, Cambridge, UK, 1964.

[6] G. Fejes Tóth and W. Kuperberg, *Packings and coverings with convex sets*, in Handbook of Convex Geometry, Vol. B, P.M. Gruber and J.M. Wills, eds., North-Holland, Amsterdam, 1993, pp. 799–860.

[7] C. K. Wong and D. Coppersmith, *A combinatorial problem related to multimodule memory organizations*, J. ACM, 21 (1974), pp. 392–402.

SIAM J. DISCRETE MATH.
Vol. 13, No. 2, pp. 202–211

# CONNECTED DOMINATION AND SPANNING TREES WITH MANY LEAVES[*]

YAIR CARO[†], DOUGLAS B. WEST[‡], AND RAPHAEL YUSTER[†]

**Abstract.** Let $G = (V, E)$ be a connected graph. A *connected dominating set* $S \subset V$ is a dominating set that induces a connected subgraph of $G$. The *connected domination number* of $G$, denoted $\gamma_c(G)$, is the minimum cardinality of a connected dominating set. Alternatively, $|V| - \gamma_c(G)$ is the maximum number of leaves in a spanning tree of $G$. Let $\delta$ denote the minimum degree of $G$. We prove that $\gamma_c(G) \le |V| \frac{\ln(\delta+1)}{\delta+1}(1 + o_\delta(1))$. Two algorithms that construct a set this good are presented. One is a sequential polynomial time algorithm, while the other is a randomized parallel algorithm in RNC.

**Key words.** connectivity, domination, spanning trees

**AMS subject classifications.** 05C69, 05C40, 05C05

**PII.** S0895480199353780

## 1. Introduction.

All graphs considered here are finite, undirected, and simple. For standard graph-theoretic terminology the reader is referred to [5]. A major area of research in graph theory is the theory of domination. Recently two books [14, 15] have been published that present most of the known results concerning domination parameters. Among the most popular of these parameters is the "connected domination number," which we study here.

A subset $D$ of vertices in a graph $G$ is a *dominating set* if every vertex not in $D$ has a neighbor in $D$. If the subgraph induced by $D$ is connected, then $D$ is called a *connected dominating set*. The *domination number*, denoted $\gamma(G)$, and the *connected domination number*, denoted $\gamma_c(G)$, are the minimum cardinalities of a dominating set and a connected dominating set, respectively. A graph $G$ has a connected dominating set if and only if $G$ is connected; thus $\gamma_c(G)$ is well defined on the class of connected graphs. Also, trivially, $\gamma_c(G) \ge \gamma(G)$. Results on the connected domination number appear in [3, 4, 7, 8, 10, 11, 16, 17, 23, 25, 26, 28].

Spanning trees of connected graphs are a major topic of research in the area of graph algorithms (see, e.g., [9]). The problem of finding a spanning tree that maximizes the number of leaves is equivalent to the problem of computing $\gamma_c(G)$, because a vertex subset is a connected dominating set if and only if its complement is (contained in) the set of leaves of a spanning tree.

The problem of connected domination would arise in "real life" in the following scenario. An existing computer network with direct connections described by a graph $G$ must have the property that any computer turned "on" must always be able to send a message to any other computer turned "on." One can make sure a computer is always on by connecting it to an (expensive) unlimited power supply (UPS) source. We can meet the requirement by connecting only the computers in a connected dominating set to such power sources.

[*]Received by the editors March 24, 1999; accepted for publication (in revised form) November 18, 1999; published electronically April 6, 2000.
http://www.siam.org/journals/sidma/13-2/35378.html

[†]Department of Mathematics, University of Haifa-Oranim, Tivon 36006, Israel (yairc@macam98.ac.il, raphy@macam98.ac.il).
[‡]Department of Mathematics, University of Illinois, Urbana, IL 61801 (west@math.uiuc.edu).

In this paper we consider computational and extremal aspects of $\gamma_c(G)$. Finding the maximum number of leaves in a spanning tree of $G$ is NP-hard (cf. [12, p. 206]), so we will be content with constructing a spanning tree having many leaves. The NP-hardness of the optimization problem leads us to seek constructive proofs for related extremal problems. A constructive proof that all graphs in a particular class have spanning trees with at least $\mu$ leaves becomes an algorithm to produce such a tree for graphs in this class.

When $G$ is a cycle, we can guarantee only two leaves. Let $G_{n,k}$ be the class of simple connected $n$-vertex graphs with minimum degree at least $k$. Let $l(n,k)$ be the maximum $m$ such that every $G \in G_{n,k}$ has a spanning tree with at least $m$ leaves. Alternatively, define $\gamma_c(n,k) = n - l(n,k)$ to be the minimum $m$ such that every $G \in G_{n,k}$ has a connected dominating set of size at most $m$. Finally, define $\gamma(n,k)$ to be the minimum $m$ such that each (not necessarily connected) $n$-vertex graph with minimum degree at least $k$ has a dominating set of size at most $m$.

The value $l(n,k)$ is known for $k \leq 5$. We have seen that $l(n,2) = 2$. For arbitrary $k$, $C_n$ generalizes to a well-known graph in $G_{n,k}$ having no tree with "very many" leaves. Let $m = \lfloor \frac{n}{k+1} \rfloor$. Form cliques $R_1, \dots, R_m$ of orders $k+1$ and $k+2$, altogether having $n$ vertices. Place the cliques in a ring. Delete one edge $x_i y_i$ from each $R_i$, and restore minimum degree $k$ by adding edges of the form $x_i y_{i+1}$. In forming a spanning tree of this graph, there must be a nonleaf other than $\{x_i, y_i\}$ in each $R_i$, except possibly for two of them. The construction yields $l(n,k) \leq \lceil \frac{k-2}{k+1} n \rceil + 2$.

This construction is essentially optimal when $k$ is small. Linial and Sturtevant (unpublished) proved that $l(n,3) \geq \frac{n}{4} + 2$. For $k = 4$, the optimal bound $l(n,4) \geq \frac{2}{5}n + \frac{8}{5}$ is proved in Griggs and Wu [13] and in Kleitman and West [19] (two small graphs have no tree with $\frac{2}{5}n + 2$ leaves). In [13] it is also proved that $l(n,5) \geq \frac{3}{6}n + 2$. These proofs are algorithmic, constructing a tree with at least this many leaves in polynomial time. For $k \geq 6$ the exact value of $l(n,k)$ remains unknown.

The example above was thought to be essentially optimal for all $k$. However, Alon [1] proved by probabilistic methods that when $n$ is large there exists a graph with minimum degree $k$ and with no dominating set of size less than $(1 + o_k(1)) \frac{1+\ln(k+1)}{k+1} n$. Since connected dominating sets are also dominating sets, this shows that $\gamma_c(n,k) \geq (1 + o_k(1)) \frac{1+\ln(k+1)}{k+1} n$. Equivalently, $l(n,k) \leq (1 + o_k(1)) \frac{k-\ln(k+1)}{k+1} n$.

A well-known result of Lovász [21] (see [2]) states that $\gamma(G) \leq n \frac{1+\ln(k+1)}{k+1}$ for every $n$-vertex graph $G$ with minimum degree $k > 1$. Together with Alon's result, this yields $\gamma(n,k) = (1 + o_k(1)) n \frac{1+\ln(k+1)}{k+1}$. Thus $\gamma(n,k)$ is asymptotically determined (it is known exactly for $k \leq 3$ [22, 24]).

Kleitman and West [19] gave an upper bound for $\gamma_c(n,k)$ that is only 2.5 times larger than Alon's lower bound. They proved that if $n$ is sufficiently large, then $\gamma_c(n,k) \leq C \frac{\ln k}{k} n$, where $C$ is very close to 2.5. Our main result in this paper improves this result of Kleitman and West and is asymptotically sharp. We summarize our result in the following theorem.

THEOREM 1.1.

$$\gamma_c(n,k) = (1 + o_k(1)) n \frac{\ln(k+1)}{k+1}.$$

An interesting consequence of Theorem 1.1 is that $\gamma_c(n,k)$ behaves essentially like $\gamma(n,k)$ when $k$ is sufficiently large.

We supply two proofs for Theorem 1.1. The first uses a probabilistic approach similar to the proof of the Lovász bound in [2] and to the proof in [6]. Our proof here is more complicated since we need to use several large deviation inequalities along with a sharpened form of a result of Duchet and Meyniel [11]. The proof yields the following technical statement, which immediately implies Theorem 1.1.

THEOREM 1.2. *If $G$ is an $n$-vertex connected graph with minimum degree $k$, then*

$$\gamma_c(G) \leq n\frac{145 + 0.5\sqrt{\ln(k+1)} + \ln(k+1)}{k+1}.$$

*Furthermore, there exists a polynomial time randomized algorithm that generates a connected dominating set this good, with constant probability. This algorithm can also be implemented in parallel in RNC.*

Our second proof for Theorem 1.1 yields a purely sequential algorithm for finding a spanning tree with the required number of leaves. At each stage in the algorithm we maintain a subtree of the final spanning tree, and vertices are added to the tree in ways that tend to increase the number of leaves. Unlike the first proof, this algorithm cannot be parallelized, but it has the advantage of being purely deterministic. We summarize this algorithm in the following theorem.

THEOREM 1.3. *Given $\epsilon > 0$, and given $k$ sufficiently large in terms of $\epsilon$, every connected simple graph with order $n$ and minimum degree $k$ has a spanning tree with more than $(1 - \frac{(1+\epsilon)\ln k}{k})n$ leaves. Furthermore, there is a polynomial time algorithm that constructs such a tree.*

Note that Theorem 1.1 follows also from Theorem 1.3. In the next section we prove Theorem 1.2, and in section 3 we prove Theorem 1.3.

**2. The probabilistic proof.** We begin with a lemma that sharpens a result of Duchet and Meyniel [11], who proved that $\gamma(G) \leq \gamma_c(G) \leq 3\gamma(G) - 2$.

LEMMA 2.1. *Let $G$ be a connected graph. If $X$ is a dominating set of $G$ that induces a subgraph with $t$ components, then $\gamma_c(G) \leq |X| + 2t - 2$. In particular,*

$$\gamma(G) \leq \gamma_c(G) \leq 3\gamma(G) - 2.$$

*Proof.* It suffices to show that whenever $t > 1$, we can find at most two vertices in $V \setminus X$ such that adding them to $X$ decreases the number of components by at least one. Partition $X$ into parts $X_1$ and $X_2$ such that $X_1$ and $X_2$ have no edge connecting them. Let $x_1 \in X_1$ and $x_2 \in X_2$ be two vertices whose distance in $G$ is the smallest possible. The distance between $x_1$ and $x_2$ is at most 3, because otherwise there is a vertex (in the middle of a shortest path from $x_1$ to $x_2$) that has distance at least 2 from both $X_1$ and $X_2$ and is undominated. ∎

*Proof of Theorem* 1.2. The theorem clearly holds for $k < 100$, so we assume $k \geq 100$. Let $p = \frac{\ln(k+1)}{k+1}$; note that $0 \leq p < 1$. Pick each vertex of $V$, randomly and independently, with probability $p$. Let $X$ denote the set of vertices that are picked. Let $Y$ denote the set of vertices that are not picked and have no picked neighbor. Finally, let $z$ denote the number of components in the subgraph induced by $X$. Clearly, $x = |X|, y = |Y|$, and $z$ are random variables. By definition, $S = X \cup Y$ is a dominating set of $G$. Since $S$ has at most $z + y$ components, Lemma 2.1 yields $\gamma_c(G) \leq x + y + 2(z + y) - 2 = x + 2z + 3y - 2$. It therefore suffices to show that with positive probability,

$$x + 2z + 3y - 2 \leq n\frac{145 + 0.5\sqrt{\ln(k+1)} + \ln(k+1)}{k+1}. \tag{2.1}$$

Claim 1.

$$\text{Prob}\left[x > n\frac{\ln(k+1)}{k+1} + n\frac{0.5\sqrt{\ln(k+1)}}{k+1}\right] < 0.91.$$

*Proof of Claim* 1. The expectation of $x$ is $E[x] = np = n\frac{\ln(k+1)}{k+1}$. We shall use the following large deviation result attributed to Chernoff (cf. [2, Appendix A]):

$$\text{Prob}\Big[x - E[x] > a\Big] < \exp\left(-a^2/(2pn) + a^3/(2p^2n^2)\right).$$

(We use here the fact that $x$ is a sum of $n$ independent indicator random variables each having probability $p$ of success.) Substituting $n\frac{0.5\sqrt{\ln(k+1)}}{k+1}$ for $a$ in this inequality yields

$$\text{Prob}\left[x > n\frac{\ln(k+1)}{k+1} + n\frac{0.5\sqrt{\ln(k+1)}}{k+1}\right]$$

$$< \exp\left(-\frac{n^2\ln(k+1)}{8(k+1)^2 n\frac{\ln(k+1)}{k+1}} + \frac{n^3\ln^{3/2}(k+1)}{16(k+1)^3 n^2\frac{\ln^2(k+1)}{(k+1)^2}}\right)$$

$$= \exp\left(-\frac{n}{8(k+1)} + \frac{n}{16(k+1)\sqrt{\ln(k+1)}}\right) \leq \exp\left(-\frac{1}{8} + \frac{1}{16\sqrt{\ln(k+1)}}\right)$$

$$\leq \exp\left(-\frac{1}{8} + \frac{1}{16\sqrt{\ln(101)}}\right) < 0.91.$$

In the last inequality we used $n \geq k+1 \geq 101$. This establishes Claim 1.

Claim 2.

$$\text{Prob}\left[y > 25\frac{n}{k+1}\right] < 0.04.$$

*Proof of Claim* 2. For each vertex $v$, the probability that $v \in Y$ is exactly $(1-p)^{d_v+1}$ where $d_v$ is the degree of $v$. Since $d_v \geq k$, it follows that the expectation of $y$ satisfies $E[y] \leq n(1-p)^{k+1}$. By using the well-known inequality from elementary calculus

$$\left(1 - \frac{\ln(k+1)}{k+1}\right)^{k+1} < \frac{1}{k+1},$$

we obtain $E[y] < n/(k+1)$. It now follows immediately from Markov's inequality that

$$\text{Prob}\left[y > 25\frac{n}{k+1}\right] < \frac{1}{25} = 0.04.$$

This establishes Claim 2.

It is not easy to bound $z$ directly. Instead, we will show that the total number of vertices in small components in $X$ is rather small. We say that a vertex $v \in V$ is *weakly dominated* if $v$ has fewer than $0.1\ln(k+1)$ neighbors in $X$. We now bound the probability that a vertex is weakly dominated. Let $X_v$ denote the number of neighbors of $v$ in $X$. Clearly,

$$E[X_v] = pd_v \geq pk = \frac{\ln(k+1)}{k+1}k \geq \frac{100}{101}\ln(k+1).$$

CLAIM 3.

$$\text{Prob}\Big[X_v < 0.1\ln(k+1)\Big] < \left(\frac{1}{k+1}\right)^{0.4}.$$

*Proof of Claim* 3. Once again, we use a large deviation inequality. However, now we need to bound the lower tail, so we use the inequality (cf. [2, Appendix A])

$$\text{Prob}\Big[X_v - E[X_v] < -a\Big] < \exp\left(-\frac{a^2}{2E[X_v]}\right),$$

which is valid for every $a > 0$. Using $a = (1 - 101/1000)E[X_v]$, we obtain

$$\text{Prob}\Big[X_v < 0.1\ln(k+1)\Big] \leq \text{Prob}\left[X_v < \frac{101E[X_v]}{1000}\right]$$

$$= \text{Prob}\left[X_v - E[X_v] < -(1 - \frac{101}{1000})E[X_v]\right]$$

$$< \exp\left(-\frac{(1-101/1000)^2E[X_v]^2}{2E[X_v]}\right) = \exp\left(-\frac{(1-101/1000)^2}{2}E[X_v]\right)$$

$$< \exp\left(-\ln(k+1)\frac{100}{101}\frac{(1-101/1000)^2}{2}\right) < \left(\frac{1}{k+1}\right)^{0.4}.$$

This establishes Claim 3.

The event that a vertex $v$ is picked for $X$ is independent of the event that $v$ is weakly dominated. Therefore, by Claim 3, the probability that a vertex is weakly dominated and in $X$ is

$$\text{Prob}\Big[v \in X \text{ and } X_v < 0.1\ln(k+1)\Big] < p\left(\frac{1}{k+1}\right)^{0.4} = \frac{\ln(k+1)}{(k+1)^{1.4}}.$$

Thus, the expected number of weakly dominated vertices in $X$ is at most

$$n\frac{\ln(k+1)}{(k+1)^{1.4}}.$$

Let $U$ be the set of weakly dominated vertices in $X$. By Markov's inequality,

$$\text{Prob}\left[|U| > 20n\frac{\ln(k+1)}{(k+1)^{1.35}}\right] < \frac{n\frac{\ln(k+1)}{(k+1)^{1.4}}}{20n\frac{\ln(k+1)}{(k+1)^{1.35}}} = \frac{1}{20(k+1)^{0.05}} \leq \frac{1}{20\cdot 101^{0.05}} < 0.04.$$

From Claim 1, Claim 2, and the last inequality, it follows that with probability *at least*

$$1 - 0.91 - 0.04 - 0.04 = 0.01 > 0$$

all of the following events happen simultaneously:

$$x \le n\frac{\ln(k+1)}{k+1} + n\frac{0.5\sqrt{\ln(k+1)}}{k+1}, \qquad y \le 25\frac{n}{k+1}, \qquad |U| \le 20n\frac{\ln(k+1)}{(k+1)^{1.35}}. \tag{2.2}$$

We now fix a choice of $X$ where all these events happen simultaneously. Every component of $X$ that contains no weakly dominated vertex has size at least $0.1\ln(k+1)$. Thus the number $z$ of components in $X$ satisfies

$$z \le \frac{x}{0.1\ln(k+1)} + 20n\frac{\ln(k+1)}{(k+1)^{1.35}} \le \frac{n\frac{\ln(k+1)}{k+1} + n\frac{0.5\sqrt{\ln(k+1)}}{k+1}}{0.1\ln(k+1)} + 20n\frac{\ln(k+1)}{(k+1)^{1.35}}$$

$$\le \frac{10n}{k+1} + \frac{5n}{k+1} + \frac{20n}{k+1} = \frac{35n}{k+1}. \tag{2.3}$$

(We have used that $k \ge 100$ implies $(k+1)^{0.35} > \ln(k+1)$.) Finally, (2.2) and (2.3) yield

$$x + 2z + 3y - 2 < n\frac{145 + 0.5\sqrt{\ln(k+1)} + \ln(k+1)}{k+1},$$

as required in (2.1). ◻

We end this section by describing a parallel implementation of Theorem 1.2. The reader unfamiliar with the parallel random access machines (PRAM) model of parallel computing is referred to [18]. Let $M$ be a Boolean array of order $n$. At the end of the algorithm, the set of indices with $M(i) = True$ will be a connected dominating set in $G$. In constant parallel time, initialize all of $M$ to $False$. Next, each vertex picks itself for $X$ with probability $p$. If $v \in X$, then we put $M(v) = True$. Since each vertex is selected independently, this step also runs in constant parallel time.

Each $v \notin X$ now checks to see whether all its neighbors are also not in $X$ (the set of such vertices is denoted $Y$ in the proof). In the concurrent real and concurrent write (CRCW) PRAM model, this test can also be done in constant time (it is a Boolean "and" operation for each $v \notin X$). If $v \in Y$, then we also put $M(v) = True$. Clearly, $X \cup Y$ is a dominating set.

Using well-known NC algorithms for finding components (such as [27]), we can compute the components of the subgraph of $G$ induced by $X \cup Y$ in $O(\log n)$ parallel time on a CRCW PRAM. If there is only one component, then we are finished.

Otherwise, we proceed as follows. We compute distances joining all pairs of vertices of $G$ (namely, the $n \cdot n$ matrix $A$ of the distances). It is well known that computing $A$ can be done in NC using iterated matrix multiplication. We now create a graph $H$ whose vertices are the components of $X \cup Y$ and whose edges connect two components $C_1$ and $C_2$ of $X \cup Y$ if, and only if, for some $c_1 \in C_1$ and some $c_2 \in C_2$, the distance between $c_1$ and $c_2$ is at most 3. Given $A$, one can clearly construct $H$ in constant time on a CRCW PRAM.

By the proof of Lemma 2.1, $H$ is connected. Thus $H$ has a spanning tree. Spanning trees are also known to be computed in NC. (In fact, spanning trees are byproducts of the algorithms that find components.) Since $H$ has at most $z + y$ vertices

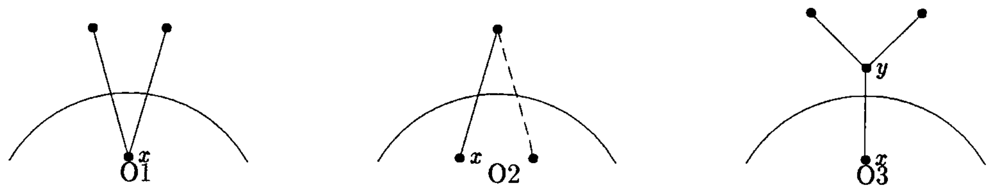


FIG. 1. *The admissible operations for $k = 3$.*

(recall that $z$ is the number of components of $X$), the spanning tree has at most $z + y - 1$ edges. Each such edge corresponds to a path of length 3 between two components of $X \cup Y$. Thus the total number of internal vertices in these paths is at most $2(z + y - 1)$. If $u$ is such an internal vertex, we put $M(u) = True$. This can be done in constant time on a CRCW PRAM.

Thus we conclude that finding a connected dominating set of $G$ is in NC. The proof of Theorem 1.2 shows that with constant positive probability, this set has size at most $n\frac{145+0.5\sqrt{\ln(k+1)}+\ln(k+1)}{k+1}$. This proves the existence of the desired RNC algorithm. In fact, the overall running time is $O(\log n)$ on a CRCW PRAM using a polynomial number of (in fact, $O(n^3)$) parallel processors.

**3. The deterministic proof.** Before giving the proof of Theorem 1.3, we describe the method as it applies to the iterative construction of a tree with at least $n/4 + 1.5$ leaves when $k = 3$. When $G \neq K_4$, we can start with a star at a vertex of degree at least 4 including all its neighbors, or with a double star where both centers have degree 3. (A double star is a tree with exactly two adjacent nonleaves called centers.) Let $T$ denote the current tree, with $s$ vertices and $l$ leaves. If $x$ is a leaf of $T$, then the *external degree* of $x$, denoted $d'(x)$, is the number of neighbors it has in $G - V(T)$. An *expansion at* $x$ is performed by adding to $T$ the $d'(x)$ edges from $x$ to $N(x) - V(T)$. We grow $T$ by *operations*, which are sequences of expansions. After each operation, all edges from $T$ to $G - V(T)$ are incident to leaves of $T$.

A leaf $x$ of $T$ with $d'(x) = 0$ is *dead*; no expansion is possible at a dead leaf, and it must be a leaf in the final tree. Let $m$ be the number of dead leaves in $T$. We call an operation *admissible* if its effect on $T$ satisfies the "augmentation inequality" $3\Delta l + \Delta m \geq \Delta s$, where $\Delta l, \Delta m, \Delta s$ denote the change in the numbers of leaves, dead leaves, and vertices, respectively, in $T$.

If $T$ is grown to a spanning tree with $L$ leaves by admissible operations, then all leaves eventually die. We begin with four leaves and six vertices if $G$ is 3-regular (the double-star case above); otherwise we begin with $r$ leaves and $r + 1$ vertices for some $r > 3$. In any case, we start with at least four leaves, so the total of $\Delta m$ from the augmentation inequalities for the operations is at most $L - 4$. Summing the augmentation inequalities over all operations yields $3(L - 4) + (L - 4) \geq n - 6$ if $G$ is 3-regular and $3(L - r) + (L - 4) \geq n - r - 1$ otherwise. These simplify to $4L \geq n + 6$ and $4L \geq n + 2r + 1 \geq n + 7$, respectively, which yield $L \geq n/4 + 1.5$.

The proof for $k = 3$ is now completed by providing a set of admissible operations that can be applied until $T$ absorbs all vertices. The three operations suggested in Figure 1 suffice.

If some leaf $x$ of the current tree has external degree at least 2, we perform operation O1, which is an expansion at $x$. Otherwise, if two leaves $x$ and $y$ of $T$ have external degree 1, and both have the same unique neighbor outside of $T$, we perform

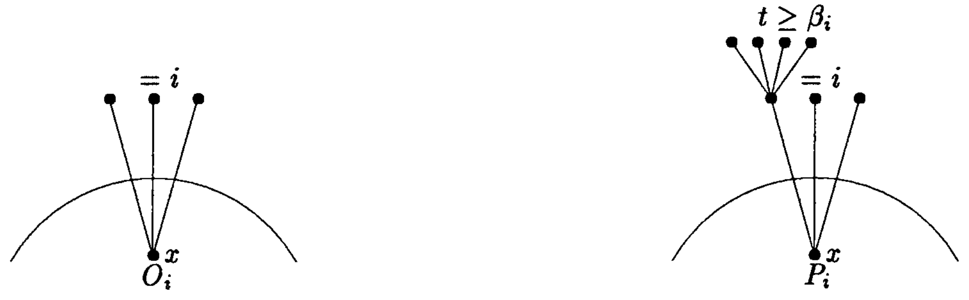


FIG. 2. *The operations $O_i$ and $P_i$.*

an expansion at one of them, say $x$. Note that $y$ becomes dead after the expansion. This is operation O2. The only remaining case is where some leaf has external degree 1, and the unique neighbor outside of $T$, denoted $y$, has no other neighbor in $T$. So $y$ has at least two other neighbors outside of $T$. We perform an expansion at $x$ and then an expansion at $y$. This is operation O3. Note that all operations satisfy the augmentation inequality.

We use the notions of deadness, augmentation inequality, and admissible operation to develop an algorithm to grow a tree with many leaves when the minimum degree is large. Our operations generalize operations O1 and O3 in Figure 1.

*Proof of Theorem* 1.3. We describe a polynomial time algorithm that grows the desired tree. Beginning with a star at a vertex of degree $k$, we again proceed by expanding the current tree $T$, which has order $s$, leaf count $l$, and external degree $d'(x)$ at each leaf $x$. We seek operations satisfying the augmentation inequality $r\Delta l+\Delta M \geq (r-1)\Delta s$, where the parameter $r$ depends on $k$ and $M$ is a measure of total "deadness" of leaves. The final value of $M$ is a multiple counting of the leaves of the final tree. Each expansion at a leaf adds all outside neighbors, and an operation with one or more expansions is *admissible* if it satisfies the augmentation inequality.

For coefficients $r > \alpha_0 \geq \alpha_1 \geq \cdots \geq \alpha_r = 0$ to be chosen later, we define $M = \sum_{i=0}^{r-1} \alpha_i m_i$, where $T$ has $m_i$ leaves with external degree $i$. For the final tree, $M = \alpha_0 L$. If we grow a tree by admissible operations, then summing the augmentation inequalities yields $r(L-k) + \alpha_0 L \geq (r-1)(n-k-1)$. Solving for $L$ yields $L \geq \frac{(r-1)n+k+1-r}{r+\alpha_0}$. We choose $r < k$; this permits dropping the additive constant. We then divide top and bottom by $r$ and apply $1/(1+\frac{\alpha_0}{r}) > 1 - \frac{\alpha_0}{r}$ to obtain $L > (1-\frac{1}{r})(1-\frac{\alpha_0}{r})n > (1-\frac{1+\alpha_0}{r})n$.

We define operations $O_i$ and $P_i$ for each $i$, applied only when the maximum external degree of current leaves is $i$. The operation $O_i$ is a single expansion at a vertex of external degree $i$. The operation $P_i$ is expansion at a vertex of external degree $i$ and expansion at one of its new neighbors that introduces the maximum number of additional leaves. The operations are depicted in Figure 2. When the maximum external degree is $i$, we perform a $P_i$ if the number of vertices introduced by the second expansion is at least $\beta_i = 2r + \alpha_i - i$; if no such $P_i$ exists, we apply an $O_i$. By construction, some such operation is always available until we grow a spanning tree. It remains to choose $r$ and the constants $\alpha_i$ so that all operations are admissible and so that $(1+\alpha_0)/r < (1+\epsilon)\ln k/k$.

The net change to $M$ by $O_i$ or $P_i$ includes $-\alpha_i$ for the loss of $x$ as a leaf; other changes are gains. We ignore contributions to $M$ from deadness of new leaves, since we have no control over their external degree. For each edge between a new vertex $y$

and a current leaf $z$ other than $x$, we gain $\alpha_{j-1} - \alpha_j \geq 0$ if this edge reduces $d'(z)$ from $j$ to $j-1$. Note that $j \leq i$. We choose constants $c_1 \geq \cdots \geq c_r \geq 0$ and define $\alpha_i$ to be $\sum_{j=i+1}^{r} c_j$. This yields $\alpha_{j-1} - \alpha_j = c_j \geq c_i$ if $j \leq i$. Hence $\Delta M \geq -\alpha_i + qc_i$, where $q$ is the number of edges from new vertices to old leaves other than $x$.

It thus suffices to show that $\Delta s - r(\Delta s - \Delta l) - \alpha_i + c_i q \geq 0$ for each operation performed when the maximum external degree is $i$. If when applying $P_i$ the second expansion introduces $t$ leaves, then $\Delta s = t + i$ and $\Delta l = t + i - 2$. The desired inequality becomes $t + i - 2r - \alpha_i + c_i q \geq 0$, which holds since we use $P_i$ only when $t \geq 2r + \alpha_i - i$.

When we apply $O_i$ at $x$, our inability to apply $P_i$ ensures that each of the $i$ new vertices has at most $2r + \alpha_i - i$ neighbors not yet in $T$ and at most $i$ neighbors among $x$ and the other new vertices. Hence it has at least $k - 2r - \alpha_i$ edges to other leaves of $T$. With $i$ new vertices, this yields $q \geq i(k - 2r - \alpha_i)$. With $\Delta s = i$ and $\Delta l = i - 1$, the desired inequality becomes $c_i i(k - 2r - \alpha_i) \geq r - i + \alpha_i$.

We must choose $r$ and nonincreasing $\{c_i\}$ to satisfy this inequality for all $i$. We set $c_i = x/i$ for $1 \leq i \leq r$, where $x$ is a positive constant to be chosen in terms of $\epsilon$. The desired inequality becomes $x(k - 2r - \alpha_i) \geq r - i + \alpha_i$. Since $\alpha_i \geq \alpha_{i+1}$ and $i < i + 1$, it suffices to choose $r$ so that the inequality holds when $i = 1$, where it becomes $k - (2 + 1/x)r \geq (1 + 1/x)\alpha_1 - 1/x$. Our choice of $c_i$ yields $\alpha_0 = x \sum_{i=1}^{r} \frac{1}{i} \leq x[\ln r + (1/2r) + 0.577]$. (Knuth [20, pp. 73–78] discusses $\sum_{i=1}^{r} 1/i$.) Similarly, $\alpha_1 \leq x[\ln r + (1/2r) - 0.423] < x \ln r$. Hence it suffices to choose $r$ so that $k - (2 + 1/x)r \geq (1 + x)\ln r$. We choose $r = \lceil \frac{k}{2+1/x} - \frac{1+x}{2} \ln k \rceil$; this satisfies the last inequality.

We have achieved $\frac{1+\alpha_0}{r} = \frac{(2x+1)\ln k}{k} + o(\frac{\ln k}{k})$. By making $x < \epsilon/2$ and $k$ sufficiently large, we have the desired lower bound on the number of leaves in the final tree. ☐

**Acknowledgments.** The authors wish to thank Noga Alon, Teresa W. Haynes, and Jerry Griggs for helpful references.

## REFERENCES

[1] N. Alon, *Transversal numbers of uniform hypergraphs*, Graphs Combin., 6 (1990), pp. 1–4.

[2] N. Alon and J. H. Spencer, *The Probabilistic Method*, John Wiley, New York, 1991.

[3] L. Arseneau, A. Finbow, B. Hartnell, A. Hynick, D. Maclean, and L. O'Sullivan, *On minimal connected dominating sets*, J. Combin. Math. Combin. Comput., 24 (1997), pp. 185–191.

[4] C. Bo and B. Liu, *Some inequalities about the connected domination number*, Discrete Math., 159 (1996), pp. 241–245.

[5] B. Bollobás, *Extremal Graph Theory*, Academic Press, London, UK, 1978.

[6] Y. Caro, *On k-domination and k-transversal numbers of graphs and hypergraphs*, Ars Combin., 29A (1990), pp. 49-55.

[7] E.J. Cockayne, T.W. Haynes, and S.T. Hedetniemi, *Extremal Graphs for Inequalities Involving Domination Parameters*, manuscript, 1997.

[8] C.J. Colbourn and L.K. Stewart, *Permutation graphs: Connected domination and Steiner trees*, Discrete Math., 86 (1990), pp. 179-189.

[9] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[10] A. D'Atri and M. Moscarini, *Distance-hereditary graphs, Steiner trees, and connected domination*, SIAM J. Comput., 17 (1988), pp. 521–538.

[11] P. Duchet and H. Meyniel, *On Hadwiger's number and stability numbers*, North-Holland Math. Stud., 62 (1982), pp. 71–73.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[13] J.R. Griggs and M. Wu, *Spanning trees in graphs with minimum degree* 4 *or* 5, Discrete Math., 104 (1992), pp. 167-183.

[14] T. Haynes, S.T. Hedetniemi, and P. Slater, *Domination in Graphs: The Theory*, Marcel Dekker Publishers, New York, 1997.

[15] T. Haynes, S.T. Hedetniemi, and P. Slater, *Domination in Graphs: Selected Topics*, Marcel Dekker Publishers, New York, 1997.

[16] S.T. Hedetniemi and R. Laskar, *Connected domination in graphs*, in Graph Theory and Combinatorics, B. Bollobás, ed., Academic Press, London, UK, 1984, pp. 209–218.

[17] S.T. Hedetniemi and R. Laskar, *Bibliography of domination in graphs and some basic definitions of domination parameters*, Discrete Math., 86 (1990), pp.257–277.

[18] R.M. Karp and V. Ramachandran, *Parallel algorithms for shared memory machines*, in Handbook of Theoretical Computer Science A, Chapter 17, J. van Leeuwen, ed., Elsevier, Amsterdam, 1990, pp. 869–941.

[19] D. J. Kleitman and D. B. West, *Spanning trees with many leaves*, SIAM J. Discrete Math., 4 (1991), pp. 99-106.

[20] D.E. Knuth, *The Art of Computer Programming*, Addison-Wesley, Reading, MA, 1973.

[21] L. Lovász, *On the ratio of optimal and integral fractional covers*, Discrete Math., 13 (1975), pp. 383–390.

[22] W. McCuaig and B. Shepherd, *Domination in graphs with minimum degree two*, J. Graph Theory, 13 (1989), pp. 749–762.

[23] M. Moscarini, *Doubly chordal graphs, Steiner trees, and connected domination*, Networks, 23 (1993), pp. 59–69.

[24] B. Reed, *Paths, stars and the number three*, Combin. Probab. Comput., 5 (1996), pp. 267–276.

[25] E. Sampathkumar and H. Walikar, *The connected domination number of a graph*, J. Math. Phys. Sci., 13 (1979), pp. 607–613.

[26] L.A. Sanchis, *On the Number of Edges of a Graph with a Given Connected Domination Number*, manuscript, 1998.

[27] Y. Shiloach and U. Vishkin, *An $O(\log n)$ parallel connectivity algorithm*, J. Algorithms, 3 (1982), pp. 57-63.

[28] K. White, M. Farber, and W.R. Pulleyblank, *Steiner trees, connected domination and strongly chordal graphs*, Networks, 15 (1985), pp. 109-124.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 2, pp. 212–226

# BOOLEAN NORMAL FORMS, SHELLABILITY, AND RELIABILITY COMPUTATIONS[*]

ENDRE BOROS[†], YVES CRAMA[‡], OYA EKIN[§], PETER L. HAMMER[†], TOSHIHIDE IBARAKI[¶], AND ALEXANDER KOGAN[†]

**Abstract.** Orthogonal forms of positive Boolean functions play an important role in reliability theory, since the probability that they take value 1 can be easily computed. However, few classes of disjunctive normal forms are known for which orthogonalization can be efficiently performed. An interesting class with this property is the class of shellable disjunctive normal forms (DNFs). In this paper, we present some new results about shellability. We establish that every positive Boolean function can be represented by a shellable DNF, we propose a polynomial procedure to compute the dual of a shellable DNF, and we prove that testing the so-called lexico-exchange (LE) property (a strengthening of shellability) is NP-complete.

**Key words.** Boolean functions, orthogonal DNFs, dualization, shellability, reliability

**AMS subject classifications.** Primary, 90B25; Secondary, 05C65, 68R05

**PII.** S089548019732180X

## 1. Introduction.

A classical problem of Boolean theory is to derive an orthogonal form, or disjoint products form, of a positive Boolean function given in DNF (see section 2 for definitions). In particular, this problem has been studied extensively in reliability theory, where it arises as follows. One of the fundamental issues in reliability is to compute the probability that a positive Boolean function (describing the state—operating or failed—of a complex system) take value 1 when each variable (representing the state of individual components) takes value 0 or 1 randomly and independently of the value of the other variables (see, for instance, [3, 25]). For functions in orthogonal form, this probability is very easily computed by summing the probabilities associated to all individual terms, since any two terms correspond to pairwise incompatible events. This observation has prompted the development of several reliability algorithms based on the computation of orthogonal forms (see, e.g., [18, 21]).

In general, however, orthogonal forms are difficult to compute and few classes of DNFs seem to be known for which orthogonalization can be efficiently performed. An interesting class with this property, namely, the class of *shellable* DNFs, has been introduced and investigated by Ball and Provan [2, 22]. As discussed by these authors, the DNFs describing several important classes of reliability problems ($k$-out-of-$n$ systems, all-terminal connectedness, all-point reachability, etc.) are shellable. Moreover,

[*]Received by the editors May 21, 1997; accepted for publication October 14, 1999; published electronically April 6, 2000. This research was partially supported by the National Science Foundation (grants DMS 98-06389 and INT 9321811), NATO (grant CRG 931531), and the Office of Naval Research (grant N00014-92-J1375).

http://www.siam.org/journals/sidma/13-2/32180.html

[†]Rutgers Center for Operations Research, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854 (boros@rutcor.rutgers.edu, hammer@rutcor.rutgers.edu, kogan@rutcor. rutgers.edu).

[‡]Ecole d'Administration des Affaires, Université de Liège, 4000 Liège, Belgium (y.crama@ulg. ac.be).

[§]Department of Industrial Engineering, Bilkent University, Bilkent, Ankara 06533, Turkey (karasan@bilkent.edu.tr).

[¶]Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto, Japan 606-8501 (ibaraki@kuamp.kyoto-u.ac.jp).

besides its unifying role in reliability theory, shellability also provides a powerful theoretical and algorithmic tool in the study of simplicial polytopes, abstract simplicial complexes, and matroids. (This is actually where the shellability concept originates (see, e.g., [7, 8, 11, 16]); let us simply mention here, without further details, that abstract simplicial complexes are in a natural one-to-one relationship with positive Boolean functions.)

Shellability is the main topic of this paper. In section 2, we briefly review the basic concepts and notations to be used in this paper. In section 3, we establish that every positive Boolean function can be represented by a shellable DNF, and we characterize those orthogonal forms that arise from shellable DNFs by a classical orthogonalization procedure. In section 4, we prove that the dual (or, equivalently, the inverse) of a shellable DNF can be computed in polynomial time. Finally, in section 5, we define an important subclass of shellable DNFs, namely, the class of DNFs which satisfy the so-called LE property, and we prove that testing membership in this class is NP-complete.

**2. Notations, definitions, and basic facts.** Let $\mathbb{B} = \{0, 1\}$ and let $n$ be a natural number. For any subset $S \subseteq \{1, 2, \dots, n\}$, $\mathbf{1}_S$ is the characteristic vector of $S$, i.e., the vector of $\mathbb{B}^n$ whose $j$th coordinate is 1 if and only if $j \in S$. Similarly, $\mathbf{0}_S \in \mathbb{B}^n$ denotes the binary vector whose $j$th coordinate is 0 exactly when $j \in S$. The *lexicographic order* $\prec_L$ on subsets of $\{1, 2, \dots, n\}$ is defined as usual: for all $S, T \subseteq \{1, 2, \dots, n\}$, $S \prec_L T$ if and only if $\min\{j \in \{1, 2, \dots, n\} \,|\, j \in S \setminus T\} < \min\{j \in \{1, 2, \dots, n\} \,|\, j \in T \setminus S\}$.

We assume that the reader is familiar with the basic concepts of Boolean algebra and we introduce here only the notions that we explicitly use in the paper (see, e.g., [19, 20] for more information).

A *Boolean function* of $n$ variables is a mapping $f : \mathbb{B}^n \longrightarrow \mathbb{B}$. We denote by $x_1, x_2, \dots, x_n$ the variables of a Boolean function and we let $\mathbf{x} = (x_1, \dots, x_n)$. The *complement* of variable $x_j$ is $\overline{x}_j = 1 - x_j$. A DNF is a Boolean expression of the form

$$\Psi(x_1, \dots, x_n) = \bigvee_{k=1}^{m} \bigwedge_{j \in I_k} x_j \bigwedge_{j \in J_k} \overline{x}_j, \tag{2.1}$$

where $I_k, J_k \subseteq \{1, 2, \dots, n\}$ and $I_k \cap J_k = \emptyset$ for all $1 \le k \le m$. The *terms* of $\Psi$ are the elementary conjunctions

$$T_k(x_1, \dots, x_n) = T_{I_k, J_k}(x_1, \dots, x_n) = \bigwedge_{j \in I_k} x_j \bigwedge_{j \in J_k} \overline{x}_j \quad (k = 1, 2, \dots, m).$$

(By abuse of terminology, we sometimes call "terms" the pairs $(I_k, J_k)$ themselves.)

It is customary to view any DNF $\Psi$ (or, more generally, any Boolean expression) as defining a Boolean function: for any assignment of $0-1$ values to the variables $(x_1, \dots, x_n)$, the value of $\Psi(x_1, \dots, x_n)$ is simply computed according to the usual rules of Boolean algebra. With this in mind, we say that the DNF $\Psi$ *represents* the Boolean function $f$ (and we simply write $f = \Psi$) if $f(\mathbf{x}) = \Psi(\mathbf{x})$ for all binary vectors $\mathbf{x} \in \mathbb{B}^n$. It is well known that every Boolean function admits (many) DNF representations.

A Boolean function $f$ is called *positive* if $f(\mathbf{x}) \ge f(\mathbf{y})$ whenever $\mathbf{x} \ge \mathbf{y}$, where the latter inequality is meant componentwise. For a positive Boolean function $f$, there is a unique minimal family of subsets of $\{1, 2, \dots, n\}$, denoted $\mathcal{P}_f$, such that $f(\mathbf{1}_S) = 1$ if and only if $S \supseteq P$ for some $P \in \mathcal{P}_f$. A subset $S$ for which $f(\mathbf{1}_S) = 1$ is called an *implicant* set (or *implicant*, for short) of $f$, and if $S \in \mathcal{P}_f$, then $S$ is called a *prime implicant* (set) of $f$.

Prime implicants of positive Boolean functions have a natural interpretation in many applied contexts. For instance, in reliability theory, prime implicants of a *coherent structure function* are in one-to-one correspondence with the *minimal pathsets* of the system under study, i.e., with those minimal subsets of elements which, when working correctly, allow the whole system to work (see, e.g., [3, 25]).

Every positive Boolean function can be represented by at least one *positive DNF*, i.e., by a DNF of the form

$$\Phi(x_1, \ldots, x_n) = \bigvee_{k=1}^{m} \bigwedge_{j \in I_k} x_j. \tag{2.2}$$

Clearly, if $I_k \subseteq I_l$ for some $k \neq l$, then the Boolean function represented by (2.2) does not change when we drop the term corresponding to $I_l$. Hence, $\Phi$ represents $f$ if and only if the (containment wise) minimal subsets of $\mathcal{I} = \{I_1, \ldots, I_m\}$ are exactly the prime implicants of $f$.

Besides its representations by positive DNFs, every positive Boolean function can also be represented by a variety of nonpositive DNFs. Let us record the following fact for further reference.

LEMMA 2.1. *If the DNF $\Psi$ given by* (2.1) *represents a positive Boolean function $f$, then $f = \bigvee_{k=1}^{m} \bigwedge_{j \in I_k} x_j$ (and $f \equiv 1$ if $I_k = \emptyset$ for some $k \in \{1, 2, \ldots, m\}$).*

*Proof.* If $\Psi$ represents $f$, then $f(\mathbf{x}) = \Psi(\mathbf{x}) \leq \bigvee_{k=1}^{m} \bigwedge_{j \in I_k} x_j$ for all $\mathbf{x} \in \mathbb{B}^n$ (since the inequality holds termwise).

To prove the reverse inequality, assume that $\bigvee_{k=1}^{m} \bigwedge_{j \in I_k} x_j^* = 1$ for some $\mathbf{x}^* \in \mathbb{B}^n$. Then there is an index $k$, $1 \leq k \leq m$, such that $\bigwedge_{j \in I_k} x_j^* = 1$, or, equivalently, $\mathbf{1}_{I_k} \leq \mathbf{x}^*$. Now, $f(\mathbf{1}_{I_k}) = \Psi(\mathbf{1}_{I_k}) = 1$ and hence, since $f$ is positive, $f(\mathbf{x}^*) = 1$. ◻

As explained in the Introduction, this paper pays special attention to orthogonal DNFs: the DNF (2.1) is said to be *orthogonal* (or is an *ODNF*, for short) if, for every pair of terms $T_k, T_l$ ($k, l \in \{1, 2, \ldots, m\}, k \neq l$) and for every $\mathbf{x} \in \mathbb{B}^n$, $T_k(\mathbf{x})T_l(\mathbf{x}) = 0$. Equivalently, (2.1) is orthogonal if and only if $(I_k \cap J_l) \cup (I_l \cap J_k) \neq \emptyset$ for all $k \neq l$.

In subsequent sections, we use the following basic properties of ODNFs.

LEMMA 2.2. *Let us assume that* (2.1) *is an ODNF of a positive Boolean function, let $k \in \{1, 2, \ldots, m\}$, and let $\mathcal{A}_k = \{I_l \mid l \in \{1, 2, \ldots, m\}$, and $I_l \cap J_k \neq \emptyset\}$. Then $J_k$ is a minimal transversal of $\mathcal{A}_k$, and $S \cap I_k \neq \emptyset$ holds for all other minimal transversals $S \neq J_k$ of $\mathcal{A}_k$.*

*Proof.* Let us assume that $S$ is a transversal of $\mathcal{A}_k$ for which $S \cap I_k = \emptyset$. Then $\mathbf{0}_S \geq \mathbf{1}_{I_k}$, and hence $\Psi(\mathbf{0}_S) \geq \Psi(\mathbf{1}_{I_k}) = 1$. Furthermore, for every term $T_l(\mathbf{x})$ of $\Psi$, $l \neq k$, we have $T_l(\mathbf{0}_S) = 0$, since either $I_k \cap J_l \neq \emptyset$ or $I_l \cap J_k \neq \emptyset$, i.e., $I_l \in \mathcal{A}_k$ and hence $I_l \cap S \neq \emptyset$, and in both cases the literals of $T_l$ corresponding to these intersections have value 0 at the vector $\mathbf{0}_S$. Thus $T_k(\mathbf{0}_S) = 1$ must hold, and hence $J_k \subseteq S$ is implied.

On the other hand, $J_k$ itself is a transversal of $\mathcal{A}_k$ (by definition of $\mathcal{A}_k$), which proves that $J_k$ is the only minimal transversal of $\mathcal{A}_k$ which is disjoint from $I_k$. ◻

LEMMA 2.3. *Let us assume that* (2.1) *is an ODNF of a positive Boolean function, let $k \in \{1, 2, \ldots, m\}$, and let $\Psi^k$ denote the disjunction of all terms of $\Psi$ but term $T_k$. Then $\Psi^k$ represents a positive Boolean function if and only if $J_k \cap I_l \neq \emptyset$ for all $l \in \{1, 2, \ldots, m\} \setminus k$.*

*Proof.* Assume first that $\Psi^k$ represents a positive Boolean function and let $l \in \{1, 2, \ldots, m\}$, $l \neq k$. Then, by Lemma 2.1, $\Psi^k(\mathbf{1}_{I_l}) = 1$. On the other hand, $T_k(\mathbf{0}_{J_k}) =$

1 and, since $\Psi$ is an ODNF, all terms other than $T_k$ vanish at $\mathbf{0}_{J_k}$, so that $\Psi^k(\mathbf{0}_{J_k}) = 0$. Thus we conclude that $\mathbf{1}_{I_l} \not\leq \mathbf{0}_{J_k}$, or, equivalently, $J_k \cap I_l \neq \emptyset$.

Let us assume next that $\Psi^k$ does not represent a positive Boolean function. In particular, $\Psi^k$ does not represent $\bigvee_{\substack{l=1 \\ l\neq k}}^{m} \bigwedge_{j\in I_l} x_j$. Hence there exists $l \neq k$ and there exists a set $S$ containing $I_l$ such that $\Psi^k(\mathbf{1}_S) = 0$. On the other hand, since $\Psi$ defines a positive Boolean function, $\Psi(\mathbf{1}_S) = 1$ must hold (by Lemma 2.1). This implies that $T_k(\mathbf{1}_S) = 1$, i.e., $J_k \cap S = \emptyset$. Therefore, $J_k \cap I_l = \emptyset$ follows. □

**3. Shellable DNFs.** As mentioned earlier, any positive Boolean function can be represented by a variety of DNFs. We now introduce one particular way of generating such a DNF representation.

In what follows, the symbol $\mathcal{I}$ always denotes an arbitrary family of subsets of $\{1, 2, \ldots, n\}$, and $\pi$ denotes a permutation of the sets in $\mathcal{I}$. Let us denote by $\pi(I)$ the rank of the set $I \in \mathcal{I}$ (i.e., its placement order) in the order of $\pi$.

DEFINITION 3.1. *For every family $\mathcal{I}$ of subsets of $\{1, 2, \ldots, n\}$, every permutation $\pi$ of the sets in $\mathcal{I}$, and every set $I \in \mathcal{I}$, the $(\mathcal{I}, \pi)$-*shadow *$J_{\mathcal{I},\pi}(I)$ of $I$ is the set*

$$J_{\mathcal{I},\pi}(I) = \{j \in \{1, 2, \ldots, n\} \mid \exists\ I' \in \mathcal{I},\ \pi(I') < \pi(I),\ I' \setminus I = \{j\}\}. \tag{3.1}$$

LEMMA 3.1. *For every permutation $\pi$ of the sets of $\mathcal{I}$, the positive Boolean function $f = \bigvee_{I\in\mathcal{I}} \bigwedge_{j\in I} x_j$ is represented by the DNF*

$$\Psi_{\mathcal{I},\pi} = \bigvee_{I\in\mathcal{I}} \bigwedge_{j\in I} x_j \bigwedge_{j\in J_{\mathcal{I},\pi}(I)} \overline{x}_j. \tag{3.2}$$

*Proof.* Clearly, $f(\mathbf{x}) \geq \Psi_{\mathcal{I},\pi}(\mathbf{x})$ for every Boolean vector $\mathbf{x}$. In order to prove the reverse inequality, let us consider any Boolean vector $\mathbf{x}^*$ such that $f(\mathbf{x}^*) = 1$. Denote by $I \in \mathcal{I}$ the first set (according to the permutation $\pi$) for which $\bigwedge_{j\in I} x_j^* = 1$. We claim that $x_j^* = 0$ for all $j \in J_{\mathcal{I},\pi}(I)$, from which there follows $\bigwedge_{j\in I} x_j^* \bigwedge_{j\in J_{\mathcal{I},\pi}(I)} \overline{x}_j^* = 1$ and $\Psi_{\mathcal{I},\pi}(\mathbf{x}^*) = 1$, as required. To establish the claim, notice that, for every $j \in J_{\mathcal{I},\pi}(I)$, there is a set $I' \in \mathcal{I}$ such that $I' \subseteq I \cup \{j\}$ and $\pi(I') < \pi(I)$. By choice of $I$, $\bigwedge_{k\in I'} x_k^* = 0$, and thus $x_j^* = 0$. □

*Example* 3.1. Let us consider the family $\mathcal{I} = \{I_1 = \{1, 2\}, I_2 = \{2, 3\}, I_3 = \{3, 4\}\}$ and the permutation $\pi = (I_1, I_3, I_2)$. Then $J_{\mathcal{I},\pi}(I_1) = J_{\mathcal{I},\pi}(I_3) = \emptyset$, $J_{\mathcal{I},\pi}(I_2) = \{1, 4\}$, and thus the positive Boolean function $f = x_1x_2 \vee x_2x_3 \vee x_3x_4$ is also represented by the DNF

$$f = \Psi_{\mathcal{I},\pi} = x_1x_2 \vee x_3x_4 \vee \overline{x}_1x_2x_3\overline{x}_4.$$

The notion of "shadow" has been put to systematic use by Ball and Provan [2] in their discussion of shellability and upper bounding procedures for reliability problems, and by Boros [9] in his work on "aligned" Boolean functions (a special class of shellable functions). Let us now recall one of the definitions of shellable DNFs.

DEFINITION 3.2. *A positive DNF $\Psi = \bigvee_{I\in\mathcal{I}} \bigwedge_{j\in I} x_j$ is called* shellable *if there exists a permutation $\pi$ of $\mathcal{I}$ (called* shelling order *of $\mathcal{I}$, or of $\Psi$) with the following property: for every pair of sets $I_1, I_2 \in \mathcal{I}$ with $\pi(I_1) < \pi(I_2)$, there exists $j \in I_1 \cap J_{\mathcal{I},\pi}(I_2)$ (or equivalently: there exists $j \in I_1$ and $I_3 \in \mathcal{I}$ such that $\pi(I_3) < \pi(I_2)$ and $I_3 \setminus I_2 = \{j\}$).*

Definition 3.2 is due to Ball and Provan [2], who observe that it is essentially equivalent (up to complementation of all sets in $\mathcal{I}$) to the "classical" definition of shellability used, for instance, in [8, 11, 16]. The connection between Lemma 3.1 and the notion of shellability is clarified in the next lemma (this result is implicit in [2], where alternative characterizations of shellability can also be found).

LEMMA 3.2. *Permutation $\pi$ is a shelling order of $\mathcal{I}$ if and only if the DNF $\Psi_{\mathcal{I},\pi}$ defined by* (3.2) *is orthogonal.*

*Proof.* Consider any two terms, e.g., $T_1 = \bigwedge_{j\in I_1} x_j \bigwedge_{j\in J_{\mathcal{I},\pi}(I_1)} \overline{x}_j$ and $T_2 = \bigwedge_{j\in I_2} x_j \bigwedge_{j\in J_{\mathcal{I},\pi}(I_2)} \overline{x}_j$ of $\Psi_{\mathcal{I},\pi}$, for which $\pi(I_1) < \pi(I_2)$.

Assume first that $\pi$ is a shelling order of $\mathcal{I}$. By Definition 3.2, there is an index $j$ in $I_1 \cap J_{\mathcal{I},\pi}(I_2)$. This shows that $\Psi_{\mathcal{I},\pi}$ is orthogonal.

Conversely, assume that $\Psi_{\mathcal{I},\pi}$ is orthogonal. If $I_1 \cap J_{\mathcal{I},\pi}(I_2)$ is nonempty, then $I_1$ and $I_2$ satisfy the condition in Definition 3.2. So, assume now that $I_1 \cap J_{\mathcal{I},\pi}(I_2) = \emptyset$, and assume further that $I \cap J_{\mathcal{I},\pi}(I_2) \neq \emptyset$ for all $I \in \mathcal{I}$ such that $\pi(I) < \pi(I_1)$ (if this is not the case, simply replace $I_1$ by $I$ in the proof). Since $\Psi_{\mathcal{I},\pi}$ is orthogonal, there must be some index $j$ in $I_2 \cap J_{\mathcal{I},\pi}(I_1)$. By Definition 3.1, there exists a set $I_3 \in \mathcal{I}$ with $\pi(I_3) < \pi(I_1)$ such that $I_3 \setminus I_1 = \{j\}$. Now, we derive the following contradiction: on the one hand, by our choice of $I_1$, $I_3 \cap J_{\mathcal{I},\pi}(I_2)$ may not be empty (since $\pi(I_3) < \pi(I_1)$); on the other hand, $I_3 \cap J_{\mathcal{I},\pi}(I_2)$ must be empty, since $j \notin J_{\mathcal{I},\pi}(I_2)$ and $I_1 \cap J_{\mathcal{I},\pi}(I_2) = \emptyset$. □

Observe that the DNF $\Psi_{\mathcal{I},\pi}$ associated to a shelling order $\pi$ of $\mathcal{I}$ is orthogonal in a rather special way: namely, for any two terms $T_1$ and $T_2$ such that $\pi(I_1) < \pi(I_2)$, the "positive part" $\bigwedge_{j\in I_1} x_j$ of the first term is orthogonal to the "negative part" $\bigwedge_{j\in J_{\mathcal{I},\pi}(I_2)} \overline{x}_j$ of the second term (this follows directly from Definition 3.2).

As one may expect, not every positive DNF is shellable: a minimal counterexample is provided by the DNF

$$\Phi(x_1,\dots,x_4) = x_1x_2 \vee x_3x_4.$$

On the other hand, it can be shown that every positive Boolean function can be represented by shellable DNFs (see also [9, Theorem 1]).

THEOREM 3.3. *Every positive Boolean function $f$ can be represented by a shellable DNF.*

*Proof.* As a first proof, let us consider the DNF

$$\Phi = \bigvee_{I\in\mathcal{I}} \bigwedge_{j\in I} x_j,$$

where $\mathcal{I}$ denotes the family of all implicants of the function $f$, and let $\pi$ be a permutation ordering these implicants in a nonincreasing order by their cardinality. Then $\Phi$ represents $f$, and it is easy to see by Definition 3.2 that $\pi$ is a shelling order of $\Phi$.

Since the above DNF can, in general, be very large compared to the number of prime implicants of $f$, let us show below another construction, using only a smaller subset of the implicants.

Call a *leftmost implicant* of $f$ any implicant $I$ of $f$ for which $I \setminus \{h(I)\}$ is not an implicant of $f$, where $h(I)$ denotes the highest-index element of the subset $I$. Let $\mathcal{L}$ denote the family of leftmost implicants of $f$. Clearly, all prime implicants of $f$ are in $\mathcal{L}$; therefore $f$ is represented by the DNF $\Psi_{\mathcal{L}} = \bigvee_{I\in\mathcal{L}} \bigwedge_{j\in I} x_j$. Let us now consider the permutation $\pi$ of $\mathcal{L}$ induced by the lexicographic order of these implicants. We claim that $\pi$ is a shelling order of $\mathcal{L}$.

To prove the claim, let $I_1$ and $I_2$ be two leftmost implicants of $f$ with $I_1 \prec_L I_2$, and let $j = \min\{i | i \in I_1 \setminus I_2\}$. If $j = h(I_1)$, then $j \in I_1 \cap J_{\mathcal{I},\pi}(I_2)$ (take $I_3 = I_1$ in Definition 3.2), and we are done. So, assume next that $j < h(I_1)$. Let $T = I_2 \cup \{j\}$ and denote by $j_1, \dots, j_h$ the elements of $\{i \in I_2 | i \geq j\}$ ordered by increasing value: $j = j_1 < j_2 < \cdots < j_h = h(I_2)$. Clearly, $T$ is an implicant of $f$, while $T \setminus \{j_2, j_3, \dots, j_h\}$ is not (since the latter set is contained in $I_1 \setminus \{h(I_1)\}$). Consider now the last implicant in the sequence $T$, $T \setminus \{j_h\}$, $T \setminus \{j_{h-1}, j_h\}, \dots$, $T \setminus \{j_2, j_3, \dots, j_h\}$, and call it $I_3$. By definition, $I_3$ is a leftmost implicant of $f$. Moreover, $I_3 \prec_L I_2$ and $I_3 \setminus I_2 = \{j\}$. Thus, here again $j \in I_1 \cap J_{\mathcal{I},\pi}(I_2)$, and we conclude that $\pi$ is a shelling order of $\mathcal{L}$. □

*Example* 3.2. The leftmost implicants of the function $f(x_1, \dots, x_4) = x_1x_2 \vee x_3x_4$ are the sets $\{1,2\}$, $\{1,3,4\}$, $\{2,3,4\}$, and $\{3,4\}$, listed here in lexicographic order. The corresponding DNF

$$\Psi_{\mathcal{L}} = x_1x_2 \vee x_1x_3x_4 \vee x_2x_3x_4 \vee x_3x_4$$

represents $f$ and is shellable, since the DNF

$$\Psi_{\mathcal{L},\prec_L} = x_1x_2 \vee x_1\overline{x}_2x_3x_4 \vee \overline{x}_1x_2x_3x_4 \vee \overline{x}_1\overline{x}_2x_3x_4$$

(which also represents $f$, by Lemma 3.1) is orthogonal.

Observe that, as illustrated by the above example, the number of leftmost implicants of a positive function $f$ is usually (much) larger than the number $p_f$ of its prime implicants. As a matter of fact, one can construct functions with $n$ variables and $p_f$ prime implicants for which the smallest shellable DNF representation involves a number of terms that grows exponentially with $n$ and $p_f$. A proof of this statement will be provided in the next section.

In the remainder of this section, we concentrate on characterizing those ODNFs that arise from shellable DNFs in the following sense. Let us consider an arbitrary DNF

$$\Psi(x_1, \dots, x_n) = \bigvee_{k=1}^{m} \bigwedge_{j \in I_k} x_j \bigwedge_{j \in J_k} \overline{x}_j. \tag{3.3}$$

We say that DNF $\Psi$ is a *shelled ODNF* if $\Psi$ is orthogonal and $\Psi$ is of the form $\Psi_{\mathcal{I},\pi}$ (see (3.2)), where $\mathcal{I} = \{I_1, \dots, I_m\}$ and $\pi$ is a shelling order of $\mathcal{I}$.

The *initial segments* of $\Psi$ are the $m$ DNFs $\Psi_1, \dots, \Psi_m$ defined by

$$\Psi_l(x_1, \dots, x_n) = \bigvee_{k=1}^{l} \bigwedge_{j \in I_k} x_j \bigwedge_{j \in J_k} \overline{x}_j \tag{3.4}$$

for $l = 1, 2, \dots, m$. The next lemma provides a partial characterization of shelled ODNFs.

LEMMA 3.4. *For a DNF $\Psi$ of the form* (3.3)*, the following statements are equivalent.*

(i) *$\Psi$ is orthogonal and $\Psi$ is of the form $\Psi_{\mathcal{I},id}$, where id denotes the identity permutation $(I_1, \dots, I_m)$.*

(ii) *$I_l \cap J_k \neq \emptyset$ for all $1 \leq l < k \leq m$, and, for every $k \leq m$ and every index $j \in J_k$, there exists $l < k$ such that $I_l \setminus I_k = \{j\}$.*

(iii) *$\Psi$ is orthogonal and each initial segment of $\Psi$ represents a positive function.*

*Proof.* The equivalence of statements (i) and (ii) follows easily from Definition 3.1 and from the comments formulated after the proof of Lemma 3.2.

In view of Lemma 3.1, statement (i) implies statement (iii).

Finally, let us assume that statement (iii) holds, and let us establish statement (ii). Repeated use of Lemma 2.3 implies that $I_l \cap J_k \neq \emptyset$ for all $1 \leq l < k \leq$

$m$. Together with Lemma 2.2, this also implies that $J_k$ is a minimal transversal of $\mathcal{A}_k = \{I_1, I_2, \ldots, I_{k-1}\}$ for every $1 < k \leq m$. Fix $j \in J_k$ and consider the set $L(j) = \{l \,|\, 1 \leq l < k, I_l \cap J_k = \{j\}\}$. Since $J_k$ is a *minimal* transversal of $\mathcal{A}_k$, $L(j)$ is not empty. Moreover, for each $l \in L(j)$, $j \in I_l \setminus I_k$ (since $I_k$ and $J_k$ are disjoint). Now there are two cases.

• There is some $l \in L(j)$ such that $I_l \setminus I_k = \{j\}$: then statement (ii) holds, and we are done.

• For all $l \in L(j)$, there exists $i_l \in \{1, 2, \ldots, n\}, i_l \neq j$, such that $i_l \in I_l \setminus I_k$. In this case, the set $S := (J_k \setminus \{j\}) \cup \{i_l \mid l \in L(j)\}$ is a transversal of $\mathcal{A}_k$, is disjoint of $I_k$, and does not contain $J_k$. But this contradicts Lemma 2.2, and so the proof is complete. ☐

Notice that, for a positive DNF $\Psi = \bigvee_{I \in \mathcal{I}} \bigwedge_{j \in I} x_j$ and a permutation $\pi$ of $\mathcal{I}$, Definition 3.2 provides a straightforward polynomial-time procedure to test whether $\pi$ is a shelling order of $\Psi$. By contrast, the complexity of recognizing shellable DNFs is an important and intriguing open problem (mentioned, for instance, in [2, 11]). We shall return to this issue in section 5. For now, let us show that Lemma 3.4 allows for easy recognition of shelled ODNFs, even when $\pi$ is not given.

THEOREM 3.5. *One can test in polynomial time whether a given DNF is a shelled ODNF.*

*Proof.* Consider a DNF $\Psi = \bigvee_{k=1}^{m} \bigwedge_{j \in I_k} x_j \bigwedge_{j \in J_k} \overline{x}_j$, as in (3.3). First, we can easily check in polynomial time whether $\Psi$ is orthogonal. In the affirmative, then we test whether $\Psi$ represents a positive Boolean function: in view of Lemma 2.1, it suffices to test whether $\Psi$ represents the function $f = \bigvee_{k=1}^{m} \bigwedge_{j \in I_k} x_j$ or, equivalently, whether $\Psi(\mathbf{1}_{I_k}) \equiv 1$ for $k = 1, 2, \ldots, m$; this is again polynomial, since $\Psi$ is orthogonal.

Now we try to find the last term in the shelling order of $f$ (assuming that there is one) with the help of Lemma 3.4(iii) and Lemma 2.3. Indeed, in view of these lemmas, $I_k$ ($k \in \{1, 2, \ldots, m\}$) is a *candidate* for being the last term in a shelling order of $f$ if and only if $J_k \cap I_l \neq \emptyset$ for all $l \in \{1, 2, \ldots, m\} \setminus \{k\}$. This condition can be tested easily in polynomial time. Moreover, it is clear that every candidate remains a candidate after deletion of any other term. Thus by Lemma 2.3, we can choose an arbitrary candidate as the last term, delete it from $\Psi$, and repeat the procedure with the remaining terms.

We conclude that $\Psi$ is a shelled ODNF if and only if this procedure terminates with a complete order of its terms. ☐

## 4. Dualization of shellable DNFs.

Let us start by recalling some definitions and facts about dualization (see, e.g., [12, 20] for more information). The *dual* of a Boolean function $f(\mathbf{x})$ is the Boolean function $f^d(\mathbf{x})$ defined by

$$f^d(x_1, x_2, \ldots, x_n) = \overline{f}\,(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n).$$

It is well known that the dual of a positive function is positive. If $f = \bigvee_{I \in \mathcal{I}} \bigwedge_{j \in I} x_j$, then $f^d = \bigwedge_{I \in \mathcal{I}} \bigvee_{j \in I} x_j$ (by De Morgan's laws), and a DNF representation of $f^d$ can be obtained by applying the distributive laws to the latter expression. As a result, it is easy to see that the prime implicants of $f^d$ are exactly the minimal transversals of the family of prime implicants of $f$, i.e., $\mathcal{P}_f$ . In the context of reliability theory, the prime implicants of $f^d$ represent the *minimal cutsets* of the system under study, namely, the minimal subsets of elements whose failure causes the whole system to fail (see [3, 25]).

The *dualization problem* can now be stated as follows: given the list of prime implicants of a positive Boolean function $f$ (or, more generally, given a positive DNF

$\Psi$ representing $f$), compute all prime implicants of $f^d$. Because of the fundamental role played by duality in many applications, the dualization problem has attracted some attention in the literature (sometimes under the name of "inversion" or "complementation" problem; see, e.g., [17, 26] and the thorough discussions in [6, 12]). The question of the algorithmic complexity of dualization, however, has not been completely settled yet. Observe that, in general, $f^d$ may have many more prime implicants than $f$, as illustrated by the following example.

*Example* 4.1. For each $n \geq 1$, define the function

$$h_n(x_1, x_2, \dots, x_{2n}) = \bigvee_{j=1}^{n} x_{2j-1}x_{2j}.$$

Then $h_n$ has $n$ prime implicants, but its dual is easily seen to have $2^n$ prime implicants (each prime implicant of $h_n^d$ contains exactly one of the indices $2j-1$ and $2j$ for $j = 1, 2, \dots, n$).

Recently, Fredman and Khachiyan [13] gave a dualization algorithm which runs in time $O(L^{o(\log L)})$ on an arbitrary positive function $f$, where $L$ is the number of prime implicants of $f$ and $f^d$. The existence of a dualization algorithm whose running time is bounded by a polynomial of $L$ is, however, still an open problem. (It is generally assumed that such a dualization algorithm does not exist; see, e.g., [6, 12, 15] for further discussion.)

In this section, we are going to prove that shellable DNFs can be dualized in time polynomial in their input size. Notice that this implies, in particular, that the number of prime implicants of the dual is polynomially bounded in the number of prime implicants of the shellable DNF. These results generalize a sequence of previous results on regular and aligned DNFs, since these are special classes of shellable DNFs (see [5, 9, 10, 23, 24]).

We first state an easy lemma.

LEMMA 4.1. *If $I_1, I_2 \in \mathcal{I}$, and $I_1 \subset I_2$, then $\pi(I_2) < \pi(I_1)$ in any shelling order of $\mathcal{I}$.*

*Proof.* This is an immediate consequence of Definition 3.2. □

THEOREM 4.2. *If a positive Boolean function $f(x_1, x_2, \dots, x_n)$ can be represented by a shellable DNF of $m$ terms, then its dual $f^d$ can be represented by a shellable DNF of at most $nm$ terms.*

*Proof.* We prove the theorem by induction on $m$.

If $f$ has a shellable DNF consisting of 1 term, i.e., if $f = \bigwedge_{j \in I} x_j$ is an elementary conjunction, then its dual is represented by $\bigvee_{j \in I} x_j$, which is a shellable DNF with at most $n$ terms.

Let us now assume that the statement has been established for functions representable by shellable DNFs of at most $m-1$ terms. Let $\Psi = \bigvee_{I \in \mathcal{I}} \bigwedge_{j \in I} x_j$ be a shellable DNF of $f$ and $\sigma$ be a shelling order of $\mathcal{I}$.

Then, by Lemma 3.2,

$$\Psi_{\mathcal{I},\sigma} = g \vee \bigwedge_{j \in A} x_j \bigwedge_{j \in B} \overline{x}_j$$

is an ODNF of $f$, where $A$ is the last element of $\mathcal{I}$ according to $\sigma$, $B$ is the $(\mathcal{I}, \sigma)$-shadow of $A$, and $g$ is the function represented by the disjunction of the first $m-1$ terms of $\Psi$.

Notice that $g$ is a positive function (by Lemma 3.4(iii)) and that $g$ has a shellable DNF of $m-1$ terms. So, according to the induction hypothesis, $g^d$ has a shellable DNF of at most $n(m-1)$ terms. Let us write

$$g^d = \bigvee_{R\in\mathcal{R}} \bigwedge_{j\in R} x_j,$$

where $\mathcal{R}$ is a family of implicants of $g^d$ containing all of its prime implicants, $|\mathcal{R}| \leq n(m-1)$, and $\mathcal{R}$ admits a shelling order that we denote by $\pi$.

By Lemmas 2.2 and 2.3, $B$ is a prime implicant of $g^d$ (so that $B \in \mathcal{R}$), and, for all other sets $R \in \mathcal{R} \setminus \{B\}$, there holds $R \cap A \neq \emptyset$. Hence, for all $R \in \mathcal{R} \setminus \{B\}$, we have the identity

$$\left(\bigwedge_{j\in R} x_j\right)\left(\bigvee_{j\in A} x_j \bigvee_{j\in B} \overline{x}_j\right) = \bigwedge_{j\in R} x_j,$$

and for $R = B$ we have

$$\left(\bigwedge_{j\in B} x_j\right)\left(\bigvee_{j\in A} x_j \bigvee_{j\in B} \overline{x}_j\right) = \bigvee_{i\in A} \bigwedge_{j\in B\cup\{i\}} x_j.$$

Thus we can write

$$f^d = g^d\left(\bigvee_{j\in A} x_j \bigvee_{j\in B} \overline{x}_j\right) = \left(\bigvee_{R\in\mathcal{R}\setminus\{B\}} \bigwedge_{j\in R} x_j\right) \vee \left(\bigvee_{i\in A} \bigwedge_{j\in B\cup\{i\}} x_j\right). \tag{4.1}$$

Let $B_i = B \cup \{i\}$ for all $i \in A$, and define

$$\mathcal{R}' = (\mathcal{R} \setminus \{B\}) \cup \{B_i \,|\, i \in A \text{ and } \nexists R \in \mathcal{R} \text{ with } (R \subseteq B_i, \pi(R) < \pi(B))\}.$$

In view of (4.1), the DNF

$$\Phi = \bigvee_{R\in\mathcal{R}'} \bigwedge_{j\in R} x_j \tag{4.2}$$

represents $f^d$. We are going to show that $\mathcal{R}'$ admits a shelling order. Since $|\mathcal{R}'| < mn$, this will complete the proof.

Let us define a permutation $\pi'$ of $\mathcal{R}'$ by inserting the sets $B_i$ ($i \in A, B_i \in \mathcal{R}'$) in place of $B$ in $\pi$. More precisely, for each pair of sets $R, S \in \mathcal{R}'$, $R \neq S$, we let $\pi'(R) < \pi'(S)$ if any of the following holds:

- $R, S \in \mathcal{R}$ and $\pi(R) < \pi(S)$, or
- $R \in \mathcal{R}$, $S = B_i$ for some $i \in A$, and $\pi(R) < \pi(B)$, or
- $R = B_i$ for some $i \in A$, $S \in \mathcal{R}$, and $\pi(B) < \pi(S)$, or
- $R = B_i$ and $S = B_j$ for some $i, j \in A$ and $i < j$.

We claim that $\pi'$ is a shelling order of $\mathcal{R}'$. To prove the claim, let us show first that

$$J_{\mathcal{R}',\pi'}(R) \supseteq J_{\mathcal{R},\pi}(R) \tag{4.3}$$

for every $R \in \mathcal{R} \cap \mathcal{R}'$. If $\pi(R) < \pi(B)$, this is obvious. So let us assume that $\pi(R) > \pi(B)$, and let $j \in J_{\mathcal{R},\pi}(R)$. If $\{j\} = S \setminus R$ for some $S \in \mathcal{R} \cap \mathcal{R}'$, then clearly

$j \in J_{\mathcal{R}',\pi'}(R)$ too. On the other hand, if $\{j\} = B \setminus R$, let $i$ be any element in $A \cap R$ (remember that $A \cap R \neq \emptyset$ by Lemma 2.2). Then $\{j\} = B_i \setminus R$. If $B_i \in \mathcal{R}'$, we conclude again that $j \in J_{\mathcal{R}',\pi'}(R)$. If $B_i \notin \mathcal{R}'$, there is a set $S \in \mathcal{R} \cap \mathcal{R}'$ such that $\pi(S) < \pi(B)$ and $S \subseteq B_i$. Moreover, $j \in S$ since otherwise $S \subset R$ would follow, contradicting Lemma 4.1. Thus $\{j\} = S \setminus R$, implying again $j \in J_{\mathcal{R}',\pi'}(R)$. This establishes (4.3).

Let us show next that

$$J_{\mathcal{R}',\pi'}(B_i) \supseteq J_{\mathcal{R},\pi}(B) \cup \{j \in A \,|\, B_j \in \mathcal{R}', j < i\} \tag{4.4}$$

for every $B_i \in \mathcal{R}'$. Clearly, if $B_j \in \mathcal{R}'$ and $j < i$, then $\{j\} = B_j \setminus B_i$, and thus $j \in J_{\mathcal{R}',\pi'}(B_i)$. If $j \in J_{\mathcal{R},\pi}(B)$, let $R \in \mathcal{R}$ be such that $\pi(R) < \pi(B)$ and $\{j\} = R \setminus B$. Since $R \subseteq B_j$ and $B_i \in \mathcal{R}'$, we deduce $j \neq i$, and thus $\{j\} = R \setminus B_i$, which implies $j \in J_{\mathcal{R}',\pi'}(B_i)$.

Relations (4.3) and (4.4), together with the hypothesis that $\pi$ is a shelling order of $\mathcal{R}$, prove that the DNF $\Phi_{\mathcal{R}',\pi'}$ associated to (4.2) is orthogonal. Hence by Lemma 3.2, $\pi'$ is a shelling order of $\mathcal{R}'$ and the proof is complete. ☐

As a side remark, we notice that equality actually holds in relations (4.3) and (4.4). More interestingly, we can now prove the following result.

COROLLARY 4.3. *Let $f(x_1, x_2, \ldots, x_n)$ be a positive Boolean function. Given $\Psi$, a positive DNF of $f$, and given $\pi$, a shelling order of $\Psi$, we can generate all prime implicants of $f^d$ in $O(nm^2)$ time, where $m$ is the number of terms of $\Psi$.*

*Proof.* Let $\Psi = \bigvee_{k=1}^{m} \bigwedge_{j \in I_k} x_j$ and assume that the identity permutation $(I_1, \ldots, I_m)$ is a shelling order of $\Psi$. The proof of Theorem 4.2 immediately suggests a recursive dualization procedure, whereby the initial segments $\Psi_1, \Psi_2, \ldots, \Psi_m = \Psi$ (see (3.4)) are sequentially dualized. Using relation (4.1), all prime implicants of $\Psi_{i+1}$ can easily be generated in $O(nm)$ time once the prime implicants of $\Psi_i$ are known for $i = 1, 2, \ldots, m-1$. The overall $O(nm^2)$ time bound follows. ☐

Let us mention here that in the special cases of aligned and regular functions, there are more efficient dualization algorithms known in the literature (see, e.g., [5, 9, 10, 24]), which run in $O(n^2 m)$ time. None of those procedures, however, seem to be extendable for the class of shellable functions.

In the previous section, we have established that every positive function can be represented by a shellable DNF (Theorem 3.3). This result, combined with Theorem 4.2, might raise the impression that every positive function can be dualized in polynomial time. This, however, is *not* the case. In fact, there exist positive Boolean functions in $2n$ variables which have only $n$ prime implicants but for which every shellable DNF representation involves at least $2^n - 1$ terms. Consider, e.g., the family of functions $h_n$ $(n = 1, 2, \ldots)$ introduced in Example 4.1. It was shown in [1] that any ODNF, thus in particular any shellable DNF of $h_n$, must have at least $2^n - 1$ terms.

Observing the similarity between dualization and orthogonalization procedures, one might think that the main reason one needs so many terms in an ODNF of $h_n$ is that its dual $h_n^d$ has many prime implicants. (As we observed in Example 4.1, $h_n^d$ has $2^n$ prime implicants.) While this might be true, such a direct relation between the size of an ODNF of a Boolean function $f$ and the size of its dual $f^d$, as far as we know, has not been established yet.

## 5. The LE property for DNFs.

As mentioned before, the computational complexity of recognizing shellable DNFs is currently unknown (see [2, 11]). In this section, we consider a closely related problem, namely, the problem of recognizing DNFs with the so-called *LE property*, and we show that this problem is NP-complete.

Definition 5.1. *A positive DNF* $\Psi(x_1, x_2, \ldots, x_n) = \bigvee_{I \in \mathcal{I}} \bigwedge_{j \in I} x_j$ *has the* LE property *with respect to* $(x_1, x_2, \ldots, x_n)$ *if, for every pair of terms* $I_1, I_2 \in \mathcal{I}$ *with* $I_1 \prec_L I_2$*, there exists* $I_3 \in \mathcal{I}$ *such that* $I_3 \prec_L I_2$ *and* $I_3 \setminus I_2 = \{j\}$*, where* $j = \min\{i \,|\, i \in I_1 \setminus I_2\}$.

*We say that* $\Psi$ *has the* LE property with respect to *a permutation* $\sigma$ *of* $(x_1, x_2, \ldots, x_n)$*, or that* $\sigma$ *is an* LE order for $\Psi$*, if*

$$\Psi^\sigma(x_1, \ldots, x_n) = \bigvee_{I \in \mathcal{I}} \bigwedge_{j \in I} \sigma(x_j)$$

*has the LE property with respect to* $(x_1, x_2, \ldots, x_n)$.

*Finally, we simply say that* $\Psi$ *has the* LE property *if* $\Psi$ *has the LE property with respect to some permutation of its variables.*

The LE property has been studied extensively by Ball and Provan [2, 22]. The interest in this concept is motivated by the simple observation that every DNF with the LE property is also shellable: more precisely, if $\sigma$ is an LE order for $\Psi$, then the lexicographic order is a shelling order of $\Psi^\sigma$ (just compare Definition 3.2 and Definition 5.1). As a matter of fact, most classes of shellable DNFs investigated in the literature do have the LE property (see [2, 9]).

In view of Definition 5.1, verifying whether a DNF $\Psi$ has the LE property with respect to $(x_1, x_2, \ldots, x_n)$ can easily be done in polynomial time. Provan and Ball [22] present an $O(n^2 m)$ procedure for this problem, where $n$ is the number of variables and $m$ is the number of terms of $\Psi$. However, these authors also point out that the existence of an efficient procedure to determine whether a given DNF has the LE property (with respect to *some* unknown order of its variables) is an "interesting open question." The remainder of this paper will be devoted to a proof that such an efficient procedure is unlikely to exist.

Theorem 5.1. *It is NP-complete to decide whether a given DNF has the LE property.*

The proof of Theorem 5.1 involves a transformation from the following *balanced partition* problem.

**Input:** A finite set $V$ and a family $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$ of subsets of $V$ such that $|H_i| = 4$ for $i = 1, 2, \ldots, m$.

**Question:** Is there a *balanced partition* of $(V, \mathcal{H})$, i.e., a partition of $V$ into $V_1 \cup V_2$ such that $|V_1 \cap H_i| = |V_2 \cap H_i| = 2$ for $i = 1, 2, \ldots, m$?

Lemma 5.2. *The balanced partition problem is NP-complete.*

*Proof.* We provide a transformation from *hypergraph* 2*-colorability* to balanced partition, where hypergraph 2-colorability is defined as follows.

**Input:** A finite set $X$ and a family $\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$ of subsets of $X$ such that $|E_i| = 3$ for $i = 1, 2, \ldots, m$.

**Question:** Is there a 2*-coloring* of $(X, \mathcal{E})$, i.e., a partition of $X$ into $X_1 \cup X_2$ such that $X_1 \cap E_i \neq \emptyset$ and $X_2 \cap E_i \neq \emptyset$ for $i = 1, 2, \ldots, m$?

Hypergraph 2-colorability is NP-complete (see [14]). Given an instance $(X, \mathcal{E})$ of this problem, we let $V = X \cup \{e_1, e_2, \ldots, e_m\}$, where $e_1, e_2, \ldots, e_m$ are $m$ new elements, and we let $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$, where $H_i = E_i \cup \{e_i\}$ for $i = 1, 2, \ldots, m$. Then $(X, \mathcal{E})$ has a 2-coloring if and only if $(V, \mathcal{H})$ has a balanced partition. □

The proof of Theorem 5.1 also requires a series of technical lemmas (the proofs of which may be skipped in a first reading).

Lemma 5.3. *If the DNF* $\Psi(x_1, x_2, \ldots, x_n) = \bigvee_{I \in \mathcal{I}} \bigwedge_{j \in I} x_j$ *has the LE property with respect to* $(x_1, x_2, \ldots, x_n)$*, then the DNF* $\Psi|_{x_i=0}$ *obtained by fixing* $x_i$ *to* 0 *in* $\Psi$*,* *that is,*

$$\Psi|_{x_i=0}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \bigvee_{\substack{I\in\mathcal{I} \\ i\notin I}} \bigwedge_{j\in I} x_j,$$

*has the LE property with respect to* $(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ *for all* $i = 1, 2, \ldots, n$.

*Proof.* This is a straightforward consequence of Definition 5.1. ☐

Lemma 5.4. *The DNF*

$$\phi(a, b, c, d, y) = abcd \vee aby \vee acy \vee ady \vee bcy \vee bdy \vee cdy$$

*does not have the LE property with respect to any permutation of* $\{a, b, c, d, y\}$ *in which* $y$ *has rank either* 4 *or* 5*. On the other hand,* $\phi$ *has the LE property with respect to all permutations in which* $y$ *has rank* 3*.*

*Proof.* Consider first any permutation $\pi$ of $\{a, b, c, d, y\}$ in which $y$ has rank 5. By symmetry, we can assume without loss of generality that $\pi = (a, b, c, d, y)$. Then the first two terms of $\phi$ in lexicographic order are $I_1 = abcd$ and $I_2 = aby$, and these terms do not fulfill the condition in Definition 5.1.

An identical reasoning applies when $y$ has rank 4 (say, as in $\pi = (a, b, c, y, d)$).

Assume next that $y$ has rank 3 in $\pi$. By symmetry, we can assume that $\pi = (a, b, y, c, d)$. Then the terms of $\phi$ satisfy

$$aby \prec_L abcd \prec_L acy \prec_L ady \prec_L bcy \prec_L bdy \prec_L cdy,$$

and it is easy to verify that $\phi$ has the LE property with respect to $\pi$. ☐

Lemma 5.5. *The DNF*

$$\theta(a, b, c, d, y) = aby \vee cdy \vee abc \vee abd \vee acd \vee bcd$$

*does not have the LE property with respect to any permutation of* $\{a, b, c, d, y\}$ *in which* $y$ *has rank* 1*, nor with respect to any of the eight permutations* $(a, y, c, d, b)$, $(a, y, d, c, b)$, $(b, y, c, d, a)$, $(b, y, d, c, a)$, $(c, y, a, b, d)$, $(c, y, b, a, d)$, $(d, y, a, b, c)$, $(d, y, b, a, c)$ *(in words, these are all permutations* $\pi = (\pi_1, \ldots, \pi_5)$ *in which* $y$ *has rank* 2 *and either* $\{\pi_3, \pi_4\} = \{a, b\}$ *or* $\{\pi_3, \pi_4\} = \{c, d\}$*). On the other hand,* $\theta$ *has the LE property with respect to all permutations in which* $y$ *has rank* 3*.*

*Proof.* Consider any permutation $\pi$ of $\{a, b, c, d, y\}$ in which $y$ has rank 1. Then, $I_1 = aby$ and $I_2 = cdy$ are the first two terms of $\theta$ in lexicographic order, and these terms do not satisfy Definition 5.1.

Consider next any of the 8 permutations listed; without loss of generality say $\pi = (a, y, c, d, b)$ (the other cases are similar due to simple symmetries). Then the first two terms of $\theta$ are $aby$ and $acd$, and they violate again Definition 5.1.

Now let $\pi$ be an arbitrary permutation of $\{a, b, c, d, y\}$ in which $y$ has rank 3. By symmetry, we only need to distinguish between the permutations $\pi^1 = (a, b, y, c, d)$, $\pi^2 = (a, c, y, b, d)$, and $\pi^3 = (a, c, y, d, b)$. The lexicographic order of the terms of $\theta$ with respect to $\pi^1$ is

$$aby \prec_L abc \prec_L abd \prec_L acd \prec_L bcd \prec_L cdy.$$

Similarly, with respect to $\pi^2$ we get

$$abc \prec_L acd \prec_L aby \prec_L abd \prec_L cdy \prec_L bcd,$$

and with respect to $\pi^3$ we get

$$acd \prec_L abc \prec_L aby \prec_L abd \prec_L cdy \prec_L bcd.$$

In all three cases, $\theta$ has the LE property with respect to the corresponding permutation. $\square$

LEMMA 5.6. *Let $\psi_1, \psi_2, \ldots, \psi_k$ be positive DNFs in the same variables $x_1, \ldots, x_n$, and assume that $\psi_1, \psi_2, \ldots, \psi_k$ all have the LE property with respect to $(x_1, \ldots, x_n)$. Then, the DNF*

$$\Psi(t_1, \ldots, t_k, x_1, \ldots, x_n) = \bigvee_{1 \le i < j \le k} t_i t_j \vee \bigvee_{i=1}^{k} t_i \psi_i(x_1, \ldots, x_n)$$

*has the LE property with respect to $(t_1, \ldots, t_k, x_1, \ldots, x_n)$.*

*Proof.* Let $I_1$ and $I_2$ be two terms of $\Psi$ with $I_1 \prec_L I_2$ in the lexicographic order induced by $\pi = (t_1, \ldots, t_k, x_1, \ldots, x_n)$. We consider four cases which together exhaust all possibilities.

*Case* 1: $I_1 = t_i t_j$ and $I_2 = t_i T$, where $i, j \in \{1, \ldots, k\}$ and $T$ is either one of the variables $\{t_{j+1}, \ldots, t_k\}$ or a term of $\Psi_i$. In both cases, we can set $I_3 = I_1$ in Definition 5.1.

*Case* 2: $I_1 = t_i t_j$ and $I_2 = t_r T$, where $i, j, r \in \{1, \ldots, k\}$, $i < j$, $i < r$, and $T$ is either one of the variables $\{t_{r+1}, \ldots, t_k\}$ or a term of $\Psi_r$. Here, we can set $I_3 = t_i t_r$.

*Case* 3: $I_1 = t_i T_1$ and $I_2 = t_j T_2$, where $i, j \in \{1, \ldots, k\}$, $i < j$, and $T_1, T_2$ are terms of $\Psi_i$ and $\Psi_j$, respectively. Then $I_3 = t_i t_j$ satisfies Definition 5.1.

*Case* 4: $I_1 = t_i T_1$ and $I_2 = t_i T_2$, where $i \in \{1, \ldots, k\}$ and $T_1, T_2$ are terms of $\Psi_i$. Since $I_1 \prec_L I_2$ and $\Psi_i$ has the LE property with respect to $(x_1, \ldots, x_n)$, there exists a term of $\Psi_i$, say, $T_3$, such that $T_3 \prec_L T_2$ and $T_3 \setminus T_2 = \{\min(j \mid j \in T_1 \setminus T_2)\}$. Then we can set $I_3 = t_i T_3$ in Definition 5.1. $\square$

We are now ready for a proof of Theorem 5.1.

*Proof of Theorem* 5.1. Let $V = \{1, 2, \ldots, n\}$ and $\mathcal{H} = \{H_1, \ldots, H_m\}$ define an instance of the balanced partition problem. With this instance, we associate $n+1+4m$ variables, denoted $x_i$ $(i = 1, 2, \ldots, n)$, $y$, and $t_{jk}$ $(j = 1, 2, \ldots, m; k = 1, 2, 3, 4)$. We also define $4m$ DNFs $\psi_{jk}$ $(j = 1, 2, \ldots, m; k = 1, 2, 3, 4)$ as follows: if $H_j = \{i_1, i_2, i_3, i_4\}$, with $i_1 < i_2 < i_3 < i_4$, then we let

$$\begin{aligned}
\psi_{j1}(x_1, x_2, \ldots, x_n, y) &= \phi(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, y),\\
\psi_{j2}(x_1, x_2, \ldots, x_n, y) &= \theta(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, y),\\
\psi_{j3}(x_1, x_2, \ldots, x_n, y) &= \theta(x_{i_1}, x_{i_3}, x_{i_2}, x_{i_4}, y),\\
\psi_{j4}(x_1, x_2, \ldots, x_n, y) &= \theta(x_{i_1}, x_{i_4}, x_{i_2}, x_{i_3}, y),
\end{aligned}$$

where $\phi$ and $\theta$ are the functions introduced in Lemma 5.4 and Lemma 5.5, respectively. We look at $\psi_{j1}, \ldots \psi_{j4}$ as DNFs in the variables $(x_1, x_2, \ldots, x_n, y)$.

Now we define

$$\Psi(t_{11}, \ldots, t_{m4}, x_1, \ldots, x_n, y) = \bigvee_{\substack{i,j \in \{1,2,\ldots,m\} \\ l,k \in \{1,2,3,4\} \\ (i,l) \neq (j,k)}} t_{il} t_{jk} \vee \bigvee_{\substack{j \in \{1,2,\ldots,m\} \\ k \in \{1,2,3,4\}}} t_{jk} \psi_{jk}(x_1, \ldots, x_n, y).$$

We claim that $\Psi$ has the LE property if and only if $(V, \mathcal{H})$ has a balanced partition.

Indeed, assume that $\Psi$ has the LE property with respect to some permutation $\pi$. We associate with $\pi$ the following partition of $V$ into $V_1 \cup V_2$:

$$V_1 = \{i \in \{1, 2, \ldots, n\} \mid x_i \text{ precedes } y \text{ in } \pi\},$$
$$V_2 = \{i \in \{1, 2, \ldots, n\} \mid x_i \text{ follows } y \text{ in } \pi\}.$$

To show that this partition is balanced, consider an arbitrary subset in $\mathcal{H}$, say, $H_1$, and assume without loss of generality that $H_1 = \{1, 2, 3, 4\}$. Since $\Psi$ has the LE property with respect to $\pi$, we deduce from Lemma 5.3 that each of $\psi_{11}, \psi_{12}, \psi_{13}$, and $\psi_{14}$ has the LE property with respect to the permutation of $\{x_1, x_2, x_3, x_4, y\}$ induced by $\pi$ (to see this, simply fix all variables $t_{jk}$ to 0 except one of them, e.g., $t_{11}$).

Now, combining Lemma 5.4 and Lemma 5.5, we conclude that $y$ must have rank 3 in the permutation of $\{x_1, x_2, x_3, x_4, y\}$ induced by $\pi$ (notice that Lemma 5.5, applied simultaneously to $\psi_{12}, \psi_{13}$, and $\psi_{14}$, excludes all 24 permutations in which $y$ has rank 2). Hence, $|V_1 \cap H_1| = |V_2 \cap H_1| = 2$, as required of a balanced partition.

Conversely, assume now that $(V, \mathcal{H})$ has a balanced partition $(V_1, V_2)$. Say without loss of generality that $V_1 = \{1, 2, \ldots, l\}$ and $V_2 = \{l + 1, \ldots, n\}$, and define the permutation

$$\pi = (t_{11}, t_{12}, \ldots, t_{m4}, x_1, x_2, \ldots, x_l, y, x_{l+1}, \ldots, x_n).$$

Consider any set $H_j \in \mathcal{H}$, say, $H_j = \{i_1, i_2, i_3, i_4\}$. Since $(V_1, V_2)$ is balanced, $y$ has rank 3 in the permutation of $\{x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, y\}$ induced by $\pi$. Thus Lemma 5.4 and Lemma 5.5 imply that $\psi_{j1}, \psi_{j2}, \psi_{j3}$, and $\psi_{j4}$ all have the LE property with respect to $\pi$. By Lemma 5.6, we conclude that $\Psi$ also has the LE property with respect to $\pi$.

This concludes the proof. $\square$

The proof of Theorem 5.1 establishes that testing the LE property is already NP-complete for DNFs of degree 5 or more (if we call *degree* of a DNF the number of literals in its longest term). On the other hand, for a DNF $\Psi = \bigvee_{(i,j)\in E} x_i x_j$ of degree 2, it can be shown that $\Psi$ has the LE property if and only if $\Psi$ is shellable, or equivalently, if and only if the graph $G = (\{1, 2, \ldots, n\}, E)$ is cotriangulated (see Theorem 2 in [4]). This implies, in particular, that the LE property can be tested in polynomial time for DNFs of degree 2. The complexity of this problem remains open for DNFs of degree 3 or 4.

## REFERENCES

[1] M. O. Ball and G. L. Nemhauser, *Matroids and a reliability analysis problem*, Math. Oper. Res., 4 (1979), pp. 132–143.

[2] M. O. Ball and J. S. Provan, *Disjoint products and efficient computation of reliability*, Oper. Res., 36 (1988), pp. 703–715.

[3] R. E. Barlow and F. Proschan, *Statistical Theory of Reliability and Life Testing*, Holt, Rinehart and Winston, New York, 1975.

[4] C. Benzaken, Y. Crama, P. Duchet, P. L. Hammer, and F. Maffray, *More characterizations of triangulated graphs*, J. Graph Theory, 14 (1990), pp. 413–422.

[5] P. Bertolazzi and A. Sassano, *An $O(mn)$ algorithm for regular set-covering problems*, Theoret. Comput. Sci., 54 (1987), pp. 237–247.

[6] J. C. Bioch and T. Ibaraki, *Complexity of identification and dualization of positive Boolean functions*, Inform. and Comput., 123 (1995), pp. 50–63.

[7] A. Björner, *Homology and shellability of matroids and geometric lattices*, in Matroid Applications, N. White, ed., Cambridge University Press, Cambridge, UK, 1992, pp. 226–283.

[8] A. Björner, *Topological methods*, in Handbook of Combinatorics, R. Graham, M. Grötschel, and L. Lovász, eds., Elsevier, Amsterdam, 1995, pp. 1819–1872.

[9] E. Boros, *Dualization of Aligned Boolean Functions*, RUTCOR Research Report RRR 9-94, Rutgers University, New Brunswick, NJ, 1994.

[10] Y. Crama, *Dualization of regular Boolean functions*, Discrete Appl. Math., 16 (1987), pp. 79–85.

[11] G. Danaraj and V. Klee, *Which spheres are shellable?*, in Algorithmic Aspects of Combinatorics, Ann. Discrete Math., 2 (1978), pp. 33–52.

[12] T. Eiter and G. Gottlob, *Identifying the minimal transversals of a hypergraph and related problems*, SIAM J. Comput., 24 (1995), pp. 1278–1304.

[13] M. Fredman and L. Khachiyan, *On the complexity of dualization of monotone disjunctive normal forms*, J. Algorithms, 21 (1996), pp. 618–628.

[14] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, 1979.

[15] V. Gurvich and L. Khachiyan, *On Generating the Irredundant Conjunctive and Disjunctive Normal Forms of Monotone Boolean Functions*, Discrete Appl. Math., 96/97 (1999), pp. 363–373.

[16] V. Klee and P. Kleinschmidt, *Convex polytopes and related complexes*, in Handbook of Combinatorics, R. Graham, M. Grötschel, and L. Lovász, eds., Elsevier, Amsterdam, 1995, pp. 875–917.

[17] M. O. Locks, *Inverting and minimalizing path sets and cut sets*, IEEE Trans. Reliability, R-27 (1978), pp. 107–109.

[18] M. O. Locks, *Recursive disjoint products: A review of three algorithms*, IEEE Trans. Reliability, R-31 (1982), pp. 33–35.

[19] E. Mendelson, *Theory and Problems of Boolean Algebra and Switching Circuits*, Schaum's Outline Series, McGraw-Hill, New York, London, Sydney, 1970.

[20] S. Muroga, *Threshold Logic and its Applications*, Wiley-Interscience, New York, 1971.

[21] J. S. Provan, *Boolean decomposition schemes and the complexity of reliability computations*, DIMACS Ser. Discrete Math. 5, Amer. Math. Soc., Providence, RI, 1991, pp. 213–228.

[22] J. S. Provan and M. O. Ball, *Efficient recognition of matroid and 2–monotonic systems*, in Applications of Discrete Mathematics, Proceedings in Applied Mathematics 33, R.D. Ringeisen and F.S. Roberts, eds., SIAM, Philadelphia, PA, 1988, pp. 122–134.

[23] U. N. Peled and B. Simeone, *Polynomial time algorithms for regular set-covering and threshold synthesis*, Discrete Appl. Math., 12 (1985), pp. 57–69.

[24] U. N. Peled and B. Simeone, *An $O(nm)$-time algorithm for computing the dual of a regular Boolean function*, Discrete Appl. Math., 49 (1994), pp. 309–323.

[25] K. G. Ramamurthy, *Coherent Structures and Simple Games*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.

[26] D. R. Shier and D. E. Whited, *Algorithms for generating minimal cutsets by inversion*, IEEE Trans. Reliability, R-34 (1985), pp. 314–318.

SIAM J. Discrete Math. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 2, pp. 227–254

# A STUDY ON $r$-CONFIGURATIONS—A RESOURCE ASSIGNMENT PROBLEM ON GRAPHS*

SATOSHI FUJITA†, MASAFUMI YAMASHITA‡, AND TIKO KAMEDA§

**Abstract.** Let $G$ be an undirected graph with a set of vertices $V$ and a set of edges $E$. Given an integer $r$, we assign at most $r$ labels (representing "resources") to each vertex. We say that such an assignment is an $r$-configuration if, for each label $c$, the vertices labeled by $c$ form a dominating set for $G$. In this paper, we are interested in the maximum number, $D_r(G)$, of labels that can be assigned to the vertices of graph $G$ by an $r$-configuration. The decision problem "$D_1(G) \geq K$?" (known as the domatic number problem) is NP-complete.

We first investigate $D_r(G)$ for general graphs and establish bounds on $D_r(G)$ in terms of $D_1(G)$ and the minimum vertex degree of $G$. We then discuss $D_r(G)$ for $d$-regular graphs. We clearly have $D_r(G) \leq r(d+1)$. We show that the problem of testing if $D_r(G) = r(d+1)$ is solvable in polynomial time for $d$-regular graphs with $|V| = 2(d+1)$, but is NP-complete for those with $|V| = a(d+1)$ for some integer $a \geq 3$. Finally, we discuss cubic (i.e., 3-regular) graphs. It is easy to show $2r \leq D_r(G) \leq 4r$ for cubic graphs. We show that the decision problem for $D_1(G) = K$ is co-NP-complete for $K = 2$ and is NP-complete for $K = 4$. Although there are many cubic graphs $G$ with $D_1(G) = 2$, surprisingly, every cubic graph has a 2-configuration with five labels, i.e., $D_2(G) \geq 5$ and such a 2-configuration can be constructed in polynomial time. We use this fact to show $D_r(G) \geq \lfloor 5r/2 \rfloor$ in general.

**Key words.** resource assignment, dominating set, vertex labeling, cubic graph

**AMS subject classifications.** 05C70, 68M07, 68R10, 90B80

**PII.** S0895480196311328

## 1. Introduction.

Consider a computer network modeled by an undirected graph $H = (V(H), E(H))$ with the vertices in $V(H)$ and the edges in $E(H)$ representing computers and communication links, respectively. In resource allocation within a computer network we are faced with the problem of distributing the minimum number of copies of a (hardware or software) resource, say, a printer or a (part of a) database, to computers in such a way that every computer in the network can *quickly* access a copy of the resource. $V(H)$ could also model towns in a geographic region, in which case each edge in $E(H)$ connects two neighboring towns that are willing to share expenses for public facilities such as a library, a fire station, a stadium, etc.

We say that a computer $u$ can quickly access a copy allocated to a computer $v$ if $d_H(u,v) \leq d$ for some $d$, where the distance $d_H(u,v)$ between $u$ and $v$ in $H$ is defined by the length of a shortest path in $H$ connecting $u$ and $v$. Let $G = (V(G), E(G)) = H^d$ be defined by $V(G) = V(H)$ and $\{u,v\} \in E(G)$ if and only if $d_H(u,v) \leq d$. A set

*Received by the editors October 28, 1996; accepted for publication (in revised form) August 3, 1999; published electronically April 6, 2000. Earlier versions of some results contained in this paper appear as *A resource assignment problem on graphs*, in Proceedings of the 6th International Symposium on Algorithms and Computations, Cairns, Australia, 1995, Lecture Notes in Comput. Sci. 1004, Springer-Verlag, Berlin, 1995, pp. 418–427. This work was partly supported by the Science Research Grant-in-aid from the Ministry of Education, Science and Culture of Japan and the Natural Sciences and Engineering Research Council of Canada.

http://www.siam.org/journals/sidma/13-2/31132.html

†Department of Electrical Engineering, Faculty of Engineering, Hiroshima University, Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527, Japan (fujita@se.hiroshima-u.ac.jp).

‡Department of Computer Science and Communication Engineering, Kyushu University, 6-10-1, Hakozaki, Fukuoka, 812-8581, Japan (mak@csce.kyushu-u.ac.jp).

§School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (tiko@cs.sfu.ca).

$D \subseteq V(G)$ is said to be a *dominating set* for $G$ if, for each $u \in V(G)$, there exists a $v \in D$ such that $d_G(u,v) \leq 1$. Given any dominating set $D$ for $G$, by definition, for each $u \in V(H) = V(G)$, there exists a $v \in D$ such that $d_H(u,v) \leq d$. Therefore, our problem of resource allocation on $H$ is reducible to the problem of determining the size of a minimum dominating set in $G$, i.e., to the *dominating set problem* [9, 10].

Another problem of interest is finding the maximum number of resources to which the computer network as a whole can provide quick access, provided that each computer has bounded "capacity" and can maintain copies of at most $r$ distinct resources. When $r = 1$, under the above criterion of efficiency, the problem is reducible to the *domatic number problem* for $G$ [9], i.e., the problem of determining the size of a finest partition of $V(G)$ such that each partition block is a dominating set for $G$. Our study of $r$-configurations reported in this paper was motivated by this problem for general $r$.

Let $G$ be an undirected graph with a set of vertices $V(G)$ and a set of edges $E(G)$. Let $C$ be a countably infinite set of labels representing the set of resources.[1] Given an integer $r$, we assign at most $r$ labels to each vertex. Such an assignment can be expressed by a function $f : V \to 2^C$, such that $|f(v)| \leq r$ for each $v \in V(G)$. Let $C_f = \{c \in C \mid c \in f(v) \text{ for some } v \in V\}$, i.e., the set of labels that are actually assigned to a vertex by $f$. We say that $f$ is an *$r$-configuration* if, for each label $c \in C_f$, the vertices in $\{v \mid c \in f(v)\}$ form a dominating set for $G$.

It is known that the problem of determining if a given graph $G$ has a dominating set of size $K$ is NP-complete for planar graphs with maximum vertex degree 3, as well as for planar graphs that are regular of degree 4 [8].

We are particularly interested in the maximum number, $D_r(G)$, of labels that can be assigned to the vertices of graph $G$ by an $r$-configuration, i.e.,

$$D_r(G) = \max\{|C_f| : f \text{ is an } r\text{-configuration}\}.$$

$D_1(G)$ is known as the domatic number [6], and obviously $D_1(G) \leq \delta(G) + 1$ holds by definition, where $\delta(G)$ denotes the minimum degree of graph $G$. However, in general, testing if $D_1(G) \geq K$ is known to be NP-complete, for any fixed $K \geq 3$. Conversely, for $K = 2$, the problem is solvable in linear time. To see this note that if $G$ contains no isolated vertex, for any maximal independent set $U$ ($\subset V$) of $G$, subsets $U$ and $V - U$ are both dominating sets for $G$. If $G$ contains an isolated vertex, however, we clearly have $D_1(G) = 1$ [8]. Therefore, $D_1(G) \geq 2$ if and only if $G$ contains no isolated vertex.

For restricted classes of graphs several positive results on the domatic number are known. For example,

1. In [7], Farber showed that any strongly chordal graph $G$ is domatically full, i.e., $D_1(G) = \delta(G) + 1$, and based on this fact, several linear time algorithms have been proposed to compute an optimal domatic partition for strongly chordal graphs [7] and interval graphs [3, 14, 17].
2. If $G$ is a *triangulated disk* (i.e., a triangulation of a planar graph), then $D_1(G) \geq 3$ [15].

A dominating set $V'$ for $G = (V, E)$ is said to be *perfect* if each vertex not in $V'$ is adjacent to exactly one vertex in $V'$ [4, 13] and is said to be *independent* if it is an independent set with respect to graph $G$. The problems of finding a perfect

[1]Although a finite set $C$ with $r|V|$ elements is enough for our definition of $r$-configuration, because $r$ and $V$ are not bounded, it is convenient to assume that $C$ is infinite, without making its definition overly complicated.

dominating set and finding an independent, perfect dominating set are both NP-hard. Several positive results concerning those notions have been obtained, including polynomial-time algorithms for exactly solving several domination related problems by restricting the class of graphs [1, 2, 5, 18] and an algorithm to find an approximate solution for a nonrestricted graph [12].

Obviously, given a 1-configuration $f$ with $k$ labels $\{c_1, c_2, \ldots, c_k\}$, we can obtain an $r$-configuration with $kr$ labels by replacing each label $c_i$ $(i = 1, 2, \ldots, k)$ with a set of $r$ labels $\{c_{i,1}, \ldots, c_{i,r}\}$. We thus have the following proposition.

Proposition 1.1. *For any graph $G$ and integer $r \geq 1$,*

$$rD_1(G) \leq D_r(G).$$

In this paper, we first estimate $D_r(G)$ for general graph $G$. Our intuition tells us that $D_1(G)$ should be roughly proportional to $\delta(G)$ and that $D_r(G)$ could be much greater than $rD_1(G)$, since we could take advantage of the freedom provided by a large $\delta(G)$ and a large $r$ to pack more labels. However, it turns out that this intuition is only partially correct, as we show below. It is easy to see that $D_1(G) \leq \delta(G) + 1$ generalizes to the following.

Proposition 1.2. *For any graph $G$ and integer $r \geq 1$,*

$$D_r(G) \leq r(\delta(G) + 1).$$

However, we can also show the following: For any constant $\epsilon > 0$ and for any $r$, there is a class of graphs $G$ such that

$$D_r(G) < r(\epsilon\delta(G) + o(\delta(G))).$$

Observe that for the graphs $G$ in this class the above intuition is wrong; i.e., we have (1) $D_1(G) = o(\delta(G))$ and (2) $D_r(G) - rD_1(G) = r \times o(\delta(G))$.

It turns out that our intuition is not totally wrong. We show that for any $\lambda > 0$ and $r \geq 2$, there is a graph $G$ such that $D_r(G) - rD_1(G) \geq r\lambda$. Therefore, the difference between $D_r(G)$ and $rD_1(G)$ can grow arbitrarily large. We also derive an upper bound on $D_2(G)$ in terms of $D_1(G)$.

Next, we discuss the time complexity issues. Since the domatic number problem is NP-complete, so also is the problem of determining if $D_r(G) \geq K$, in general. We therefore restrict our attention to $d$-regular graphs and try to find the boundary between NP-complete and polynomially solvable problems. Since a vertex is adjacent to exactly $d$ other vertices and each vertex can be assigned up to $r$ labels, we clearly have

$$D_r(G) \leq r(d + 1). \tag{1.1}$$

We are particularly interested in testing if $D_r(G) = r(d + 1)$. If $D_r(G) = r(d + 1)$, then $|V(G)| = a(d+1)$ for some integer $a \geq 1$. Note that when $G$ is a complete graph $K_{d+1}$, then it is $d$-regular and $a = 1$. So, testing if $D_r(G) = r(d + 1)$ in the case where $a = 1$ is trivial. We show that for any $d$-regular graph, the problem of testing if $D_r(G) = r(d + 1)$ is solvable in polynomial time if $|V| = 2(d + 1)$ but is NP-complete if $|V| = 3(d + 1)$. It is worth pointing out that graphs with $|V(G)| = 2(d + 1)$ are more dense than those with $|V(G)| = 3(d + 1)$, and therefore the former would have more $r$-configurations than the latter.

Testing if $D_r(G) = r(d + 1)$ is NP-complete in general. We therefore explore the boundary between NP-complete and polynomially solvable problems by further

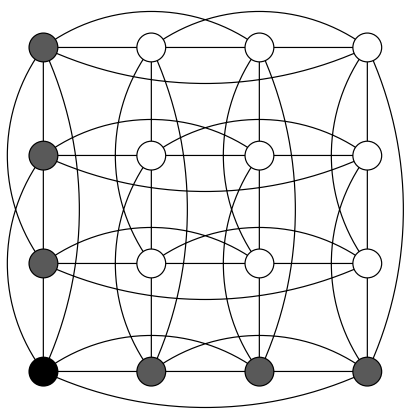


FIG. 2.1. *Graph $G_1$ in the proof of Lemma* 2.1.

restricting our attention to the 3-regular (also called cubic) graphs. Since any graph with no isolated vertex has a domatic partition of size 2, it follows that $2r \leq D_r(G)$, and from (1.1) we have $D_r(G) \leq 4r$. Hence, $D_1(G) = 2, 3$, or 4. We can show that the problem of testing if $D_1(G) = K$ is, unfortunately, co-NP-complete for $K = 2$ and is NP-complete for $K = 4$. For $r = 2$, the above inequalities yield $4 \leq D_2(G) \leq 8$. Although there are infinitely many cubic graphs $G$ with $D_1(G) = 2$, surprisingly, any cubic graph has a 2-configuration with five labels, i.e., $D_2(G) \geq 5$ in general, and such a 2-configuration can be constructed in polynomial time. We use this fact to show $D_r(G) \geq \lfloor 5r/2 \rfloor$.

The rest of this paper is organized as follows. Section 2 is devoted to investigating $D_r(G)$ for general graphs $G$. Section 3 discusses $d$-regular graphs with $D_r(G) = r(d+1)$. In section 4, we discuss the problem of testing if $D_1(G) = K$ for cubic graphs. In section 5, we show that *any* cubic graph has a 2-configuration with five labels. Section 6 concludes the paper with some open problems.

## 2. $r$-Configurations of general graphs.

### 2.1. Graphs $G$ with $D_r(G) = o(\delta(G))r$.

As we saw in Proposition 1.2, $D_r(G) \leq r(\delta(G) + 1)$ holds for any graph $G$. In this subsection, we show that for any integer $r \geq 1$ and for any constant $\epsilon > 0$, there is a class of graphs $G$ such that $D_r(G) \leq r(\epsilon\delta(G) + o(\delta(G)))$. The key to our constructive proof is the following lemma.

LEMMA 2.1. *Let $U$ be a minimum dominating set for a given graph $G = (V, E)$. Then, $D_r(G) \leq \lfloor r|V|/|U| \rfloor$.*

*Proof.* Let $f$ be an $r$-configuration for $G$ such that $|C_f| = D_r(G)$, and imagine that the vertices of $G$ are labeled by $f$. Each label is assigned to at least $|U|$ vertices and at most $r$ labels are assigned to each vertex, which implies $D_r(G)|U| \leq r|V|$. ◻

As the first step, consider the following graph $G_1$: $V(G_1) = \{(i,j) \mid 1 \leq i,j \leq m\}$, and two vertices $(i_1,j_1), (i_2,j_2) \in V(G_1)$ are connected by an edge, i.e., $((i_1,j_1),(i_2,j_2)) \in E(G_1)$, if and only if $i_1 = i_2$ or $j_1 = j_2$. Note that $G_1$ is isomorphic to the Cartesian product of two copies of $K_m$. See Figure 2.1 for an illustration ($m = 4$). In the figure, the neighbors of the black vertex are painted gray. Note that $|V(G_1)| = m^2$ and $\delta(G_1) = 2m - 2$. It is easy to see that a minimum dominating set for $G_1$ has size $m$. Hence, by Lemma 2.1,

$$D_r(G_1) \quad \leq \quad \left\lfloor \frac{r|V(G_1)|}{m} \right\rfloor \quad = \quad rm.$$

Since $m = \delta(G_1)/2 + 1$, $D_r(G_1)$ is at most $r\delta(G_1)/2 + r$.

As the next step, consider a three-dimensional "cube" instead of a two-dimensional "mesh," and define graph $G_2$ as follows: $V(G_2) = \{(i,j,k) \mid 1 \leq i,j,k \leq m\}$ and two vertices in $V(G_2)$ are connected by an edge in $E(G_2)$ if and only if they agree on at least one coordinate. Note that $|V(G_2)| = m^3$ and $\delta(G_2) = 3m^2 - 3m$. The size of a minimum dominating set for $G_2$ is clearly $m$. Since $D_r(G_2) \leq \lfloor\, r|V(G_2)|/m \,\rfloor = rm^2$ and $m^2 = \delta(G_2)/3 + m$, we have

$$D_r(G_2) \quad \leq \quad \frac{r\delta(G_2)}{3} + ro\left(\delta(G_2)\right).$$

The idea described above can be extended to higher dimensions in a natural way. Let $\ell = \lceil 1/\epsilon \rceil + 1$, and define graph $G_3$ based on an $\ell$-dimensional cube, in a similar way to $G_1$ and $G_2$. Note that $|V(G_3)| = m^\ell$ and that the size of a minimum dominating set for $G_3$ is $m$ as well. Since $D_r(G_3) \leq r|V(G_3)|/m = rm^{\ell-1}$ and

$$\delta(G_3) = \sum_{i=1}^{\ell} \binom{\ell}{i} (-1)^{i+1} m^{\ell-i} - 1,$$

we have

$$D_r(G_3) \leq rm^{\ell-1} = \frac{r\delta(G_3)}{\ell} + rO\left((\delta(G_3))^{1-1/(\ell-1)}\right) < r(\epsilon\delta(G_3) + o(\delta(G_3))).$$

Hence, we have the following theorem.

THEOREM 2.2. *For any integer $r \geq 1$ and for any constant $\epsilon > 0$, there is a class of graphs $G$ such that $D_r(G) < r(\epsilon\delta(G) + o(\delta(G)))$.*

**2.2. Relation between $D_1(G)$ and $D_r(G)$.** The objective of this subsection is to investigate the relation between $D_1(G)$ and $D_r(G)$ for a general graph $G$.

PROPOSITION 2.3. *For any graph $G$, there is a graph $G'$ such that $D_1(G') = D_r(G)$.*

*Proof.* Given a graph $G = (V,E)$, let $G'$ denote the graph constructed from $G$ as follows: For each $v \in V$, $G'$ has a clique of size $r$, and for each $(u,v) \in E$, $G'$ has a complete bipartite subgraph connecting the two $r$-cliques representing $u$ and $v$, respectively; i.e., each $(u,v) \in E$ corresponds to a joining of two $r$-cliques representing $u$ and $v$, respectively. It is easy to see that $D_1(G') = D_r(G)$. □

PROPOSITION 2.4. *Let $k$ and $l$ be any nonnegative integers. Then $D_{2k+l}(G) \geq kD_2(G) + lD_1(G)$ holds for any graph $G$.*

*Proof.* We use the approach used in the proof of Proposition 1.1 in the previous section. Start with a 2-configuration, $C_2$, with $D_2(G)$ labels, and a 1-configuration, $C_1$, with $D_1(G)$ labels. Make $k$ copies of $C_2$, using a distinct set of labels for each copy. Similarly, make $l$ copies of $C_1$, using a distinct set of labels for each copy. Consider these $k + l$ copies as one $(2k + l)$-configuration with $kD_2(G) + lD_1(G)$ labels. □

Our initial conjecture that $D_1(G) = \lfloor D_r(G)/r \rfloor$ for any $r \geq 1$ has turned out to be incorrect.

THEOREM 2.5. *There is a graph $G = (V,E)$ such that $D_1(G) \leq \lfloor D_2(G)/2 \rfloor - 1$.*

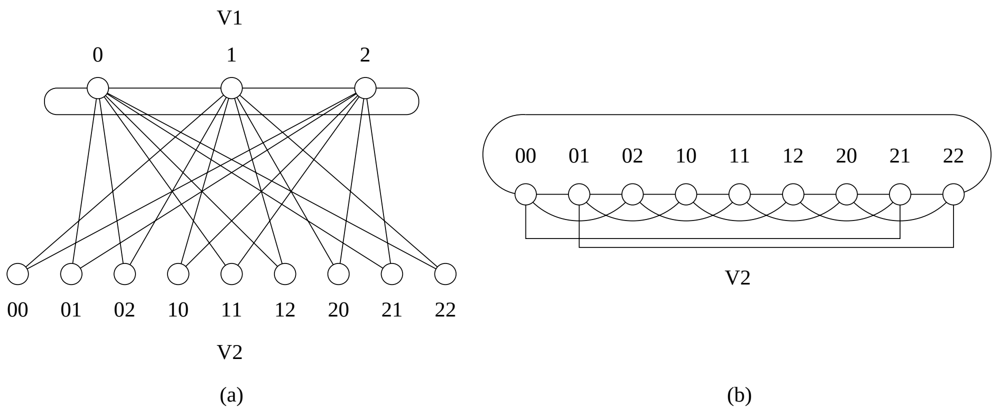


FIG. 2.2. *Graph $G$ in the proof of Theorem* 2.5. (a) *Connection between $V_1$ and $V_2$.* (b) *Connection among the vertices of $V_2$.*

*Proof.* In the following, we use strings of {0,1,2} as vertex labels. Given such a string, $v = v_p v_{p-1} \ldots v_1$, by $\bar{v}$ we denote its value interpreted as a ternary number, i.e., $\bar{v} = \sum_{i=1}^{p} v_i 3^{i-1}$. Let $V = V_1 \cup V_2$, where $V_1 = \{0,1,2\}$ and $V_2 = \{00, 01, 02, 10, 11, 12, 20, 21, 22\}$. We connect the 12 vertices of $G$ as follows (see Figure 2.2):

- For any $u, v \in V_1$, $\{u, v\} \in E$.
- For any $u, v \in V_2$, $\{u, v\} \in E$ if and only if $\bar{u} = \bar{v} + 1 \pmod 9$ or $\bar{u} = \bar{v} + 2 \pmod 9$.
- For any $u \in V_1$ and $v \in V_2$, $\{u, v\} \in E$ if and only if $u \neq v_1$, where $v = v_2 v_1$.

Any dominating set $V'$ ($\subseteq V$) for $G$ must contain at least two vertices in $V_1$ or at least two vertices in $V_2$. For example, if $V'$ contains only one vertex in $V_2$, then this vertex cannot cover all other vertices in $V_2$, and $V'$ must contain at least two vertices from $V_1$. Hence,

$$D_1(G) \quad \leq \quad \lfloor |V_1|/2 \rfloor + \lfloor |V_2|/2 \rfloor \quad = \quad 4 + 1 \quad = \quad 5.$$

Conversely, we can label the vertices in $V$ using 12 labels in a 2-configuration as observed below. Any two vertices $u, v \in V_1$ and any two vertices $u, v \in V_2$ such that $\bar{v} = \bar{u} + 4 (\mathrm{mod}\ 3^2)$ form a dominating set for $G$. Thus there are 12 different dominating sets altogether, and every vertex belongs to exactly two dominating sets. Now we prepare 12 different labels corresponding to dominating sets and label a vertex using the two labels corresponding to the dominating sets to which it belongs. Clearly, it is a 2-configuration using 12 labels. We thus have

$$D_2(G) \quad \geq \quad 12,$$

which completes the proof. □

This proof can be extended to prove the following theorem.

THEOREM 2.6. *For any integer $m \geq 1$, there is a graph $G = (V, E)$ such that $D_2(G) \geq m$ and $2D_1(G) \leq D_2(G) - \log_3(2m/3 + 1)$.*

*Proof.* Let us use strings of {0,1,2} as vertex labels, as in the proof of Theorem 2.5. Let $p = \lceil \log_3(2m/3 + 1) \rceil$. Then we have $\sum_{i=1}^{p} 3^i \geq m$. Let $V = \bigcup_{i=1}^{p} V_i$, where, for $1 \leq i \leq p$, $V_i$ is the set of all ternary strings of length $i$. Thus we have $|V_i| = 3^i$. A vertex in $V_i$ is denoted as $u = u_i u_{i-1} \ldots u_1$, where $u_j \in \{0, 1, 2\}$ for $1 \leq j \leq i$. Vertices in $V$ are connected as follows:

- For any $u, v \in V_1$, $u$ and $v$ are connected by an edge in $E$.
- For each $i \geq 2$, vertices $u = u_i u_{i-1} \dots u_1$ and $v = v_i v_{i-1} \dots v_1$ in $V_i$ such that $\bar{u} < \bar{v}$ are connected by an edge in $E$ if and only if $|\bar{u} - \bar{v}| \leq K_i$ or $|\bar{u} + 3^i - \bar{v}| \leq K_i$, where $K_i = \lceil\, (3^i - 1)/4 \,\rceil$.
- For any $u = u_i u_{i-1} \dots u_1 \in V_i$ and $v = v_j v_{j-1} \dots v_1 \in V_j$, where $j > i$, vertices $u$ and $v$ are connected by an edge in $E$ if and only if $u_1 \neq v_{j-i}$.

We now compare $D_1(G)$ and $D_2(G)$ by establishing upper and lower bounds on them, respectively. To show $D_1(G) \leq \sum_{i=1}^{p} \lfloor 3^i/2 \rfloor$, we first show that every dominating set $V'$ $(\subseteq V)$ for $G$ contains two vertices from the same set $V_i$ for some $1 \leq i \leq p$, where $p \geq 2$. Suppose that there is a dominating set $V'$ for $G$ such that $|V' \cap V_i| \leq 1$ for all $1 \leq i \leq p$. Without loss of generality, we assume that $|V' \cap V_i| = 1$ for all $i$. For $i = 1, 2, \dots, p$, let $w^{(i)} = w_i^{(i)} w_{i-1}^{(i)} \cdots w_1^{(i)} \in V' \cap V_i$.

Construct a ternary string $x$ of length $p$ as follows: $x = x_p w_1^{(1)} w_1^{(2)} \dots w_1^{(p-1)}$, where $x_p \neq w_p^{(p)}$. We claim that vertex $x \in V_p$ is not dominated by any vertex in $\{w^{(i)} \mid i = 1, 2, \dots, p\}$. By definition $w^{(i)}$ does not dominate $x$ for $i = 1, 2, \dots, p-1$. Vertex $w^{(p)}$ does not dominate $x$ either, since both $|\bar{x} - \bar{w}^{(p)}| \geq 3^{p-1}$ and $|\bar{x} + 3^p - \bar{w}^{(p)}| \geq 3^{p-1}$ hold (because $x$ and $w^{(p)}$ differ in the most significant digit) and $3^{p-1} > K_p$ for all integers $p \geq 2$.

Hence, every dominating set for $G$ contains at least two vertices from some subset $V_i$. Since there are at most $\lfloor 3^i/2 \rfloor$ disjoint pairs of elements in subset $V_i$ for all $i$,

$$D_1(G) \;\leq\; \sum_{i=1}^{p} \lfloor 3^i/2 \rfloor. \tag{2.1}$$

Next, we show $D_2(G) \geq \sum_{i=1}^{p} 3^i$. Fix any integer $i$ $(1 \leq i \leq p)$ and let $u = u_i u_{i-1} \dots u_1$ and $v = v_i v_{i-1} \dots v_1$ be two vertices in $V_i$ such that

$$\bar{u} - \bar{v} = (3^i - 1)/2.$$

Then the set $\{u, v\}$ is a dominating set as observed below. Every vertex in $V_i$ is dominated by either $u$ or $v$, since (1) $(3^i - 1)/2 \leq 2K_i$ and (2) every $w \in V_i - \{u, v\}$ satisfies $|\bar{w} - \bar{u}| \leq K_i$, $|\bar{w} + 3^i - \bar{u}| \leq K_i$, $|\bar{w} - \bar{v}| \leq K_i$, or $|\bar{w} + 3^i - \bar{v}| \leq K_i$. Also, every vertex in $V_{i'} (i' \neq i)$ is dominated by either $u$ or $v$, since $\sum_{j=1}^{i} 3^{j-1} u_j = \sum_{j=1}^{i} 3^{j-1}(v_j + 1)$. Thus there are $|V_i|$ such dominating sets $\{u, v\}$, and every vertex in $V_i$ belongs to exactly two of them.

Then as observed in the proof of Theorem 2.5, we can label the vertices in $V$ using the labels corresponding to the dominating sets $\{u, v\}$ in a 2-configuration; we label a vertex using the two labels corresponding to the dominating sets to which it belongs. It is certainly a 2-configuration of $G$, and hence,

$$D_2(G) \;\geq\; \sum_{i=1}^{p} |V_i| \;=\; \sum_{i=1}^{p} 3^i. \tag{2.2}$$

For all integers, $i \geq 1$, since $3^i$ is odd, we have $\lfloor 3^i/2 \rfloor = (3^i - 1)/2$. Thus, from (2.1) and (2.2) we obtain

$$2D_1(G) \;\leq\; D_2(G) - p.$$

Also, from $D_2(G) \geq \sum_{i=1}^{p} 3^i \geq m$ we get $D_2(G) \geq m$. It follows that

$$2D_1(G) \;\leq\; D_2(G) - p \;\leq\; D_2(G) - \log_3(2m/3 + 1). \qquad \square$$

COROLLARY 2.7. *For any fixed $\lambda$, there is a graph $G$ such that $D_2(G) \geq 2D_1(G)+\lambda$.*

The following theorem states that there is a class of graphs for which the difference between $D_r(G)$ and $rD_1(G)$ becomes arbitrarily large.

THEOREM 2.8. *For any $\lambda > 0$ and $r \geq 2$, there is a graph $G$ such that $D_r(G) - rD_1(G) \geq r\lambda$.*

*Proof.* By Proposition 2.4, for any graph $G$ and integer $k \geq 1$, we have $D_{2k}(G) \geq kD_2(G)$ and $D_{2k+1}(G) \geq kD_2(G) + D_1(G)$. Together with Corollary 2.7, we can derive

$$\begin{aligned} D_{2k}(G) - 2kD_1(G) &\geq kD_2(G) - 2kD_1(G) \\ &\geq k(2D_1(G)+\lambda) - 2kD_1(G) \\ &\geq k\lambda \end{aligned}$$

and

$$\begin{aligned} D_{2k+1}(G) - (2k+1)D_1(G) &\geq kD_2(G) + D_1(G) - (2k+1)D_1(G) \\ &\geq k(2D_1(G)+\lambda) - 2kD_1(G) \\ &\geq k\lambda. \end{aligned}$$

It thus follows that

$$D_r(G) - rD_1(G) \geq \lceil (r-1)/2 \rceil \lambda \geq r\lambda/3.$$

The proof is complete if we rename $\lambda/3$ as $\lambda$. □

In the rest of this subsection, we establish an upper bound on $D_2(G)$ in terms of $D_1(G)$.

THEOREM 2.9. *For any graph $G = (V, E)$, $D_2(G) \leq 2\alpha D_1(G)$, where $\alpha = \lceil \sqrt{2|V|/D_1(G)}\, \rceil$.*

*Proof.* Let $\mathcal{S}$ be a collection of $D_2(G)$ dominating sets corresponding to a 2-configuration for $G$ so that $|\mathcal{S}| = D_2(G)$. First, we partition $\mathcal{S}$ into subsets of disjoint dominating sets by using the following procedure.

*Step* 1. Let $j = 1$ and $\tilde{\mathcal{S}} = \mathcal{S}$. Repeat Steps 2–5 until $\tilde{\mathcal{S}} = \emptyset$.

*Step* 2. Let $\mathcal{S}_j = \emptyset$.

*Step* 3. If there is an element $Y \in \tilde{\mathcal{S}}$ which does not intersect with $\bigcup_{X\in\mathcal{S}_j} X$, then remove $Y$ from $\tilde{\mathcal{S}}$, and add it to $\mathcal{S}_j$. If there is no such element $Y$, then go to Step 5.

*Step* 4. Go to Step 3.

*Step* 5. Increment $j$ by 1.

Let $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_q\}$ be the resulting partition of $\mathcal{S}$, and for $1 \leq j \leq q$, let $U_j = \bigcup_{X\in\mathcal{S}_j} X$ $(\subseteq V)$. Note that $|\mathcal{S}_j| \leq D_1(G)$ for all $1 \leq j \leq q$, since the dominating sets in $\mathcal{S}_j$ are mutually disjoint. Note also that $\sum_{i=1}^{q} |U_i| \leq 2|V|$ since, in a 2-configuration, each vertex in $V$ can contribute to at most two dominating sets in $\mathcal{S}$. Moreover, we claim that, for any $j$ $(1 \leq j \leq q-1)$, each element in $\mathcal{S}_{j+1} \cup \mathcal{S}_{j+2} \cup \cdots \cup \mathcal{S}_q$ share at least one vertex with $U_j$ that is not shared with any other element in $\bigcup_{i=j+1}^{q} \mathcal{S}_i$. To prove this claim, consider any dominating set $d \in \bigcup_{i=j+1}^{q} \mathcal{S}_i$, and let $v \in U_j \cap d$. If there were a dominating set $d' \in \bigcup_{i=j+1}^{q} \mathcal{S}_i$ such that $d' \neq d$ and $v \in d'$, then $v$ would belong to at least three dominating sets in $\mathcal{S}$. It follows that, for any $1 \leq j \leq q-1$, we have

$$\sum_{i=j+1}^{q} |\mathcal{S}_i| \quad \leq \quad |U_j|.$$

Adding $\sum_{i=1}^{j} |\mathcal{S}_i|$ to both sides of the above inequality, we obtain

$$\sum_{i=1}^{q} |\mathcal{S}_i| \quad \leq \quad |U_j| + \sum_{i=1}^{j} |\mathcal{S}_i| \qquad \text{for all } j.$$

It follows from this inequality that

$$D_2(G) \quad \leq \quad \min_{1 \leq j \leq q} \left\{ |U_j| + \sum_{i=1}^{j} |\mathcal{S}_i| \right\}. \tag{2.3}$$

Let $\alpha = \lceil \sqrt{2|V|/D_1(G)}\ \rceil$. If $q \leq \alpha$, then, since $|\mathcal{S}_i| \leq D_1(G)$ for all $1 \leq i \leq q$, we have

$$D_2(G) \quad = \quad \sum_{i=1}^{q} |\mathcal{S}_i| \quad \leq \quad \alpha D_1(G).$$

Conversely, if $q \geq \alpha$, then we derive from (2.3)

$$D_2(G) \quad \leq \quad \min_{1 \leq j \leq \alpha} \left\{ |U_j| + \sum_{i=1}^{j} |\mathcal{S}_i| \right\} \quad \leq \quad \min_{1 \leq j \leq \alpha} \{|U_j|\} + \alpha D_1(G).$$

Since the minimum size, $\min_{1 \leq j \leq \alpha}\{|U_j|\}$, cannot be larger than the average size of the sets in $\{U_j \mid 1 \leq j \leq q\}$, we finally obtain

$$D_2(G) \quad \leq \quad \frac{2|V|}{\alpha} + \alpha D_1(G) \quad \leq \quad 2\alpha D_1(G). \qquad \square$$

Note that $|V|/D_1(G)$ represents the average size of the $D_1(G)$ dominating sets in a 1-configuration with $D_1(G)$ labels. Hence, for example, if $D_1(G) = \Omega(|V|)$ (i.e., if the average size is a constant), then $D_2(G)/D_1(G)$ is bounded from above by some constant.

## 3. $d$-Regular graphs with $D_r(G) = r(d+1)$.

Proposition 1.2 states that $D_r(G) \leq r(\delta(G)+1)$ holds in general. In this section, we restrict our attention to regular graphs and investigate the problem of deciding if a given $d$-regular $G$ has this extremal value $r(\delta(G)+1)$ as $D_r(G)$, i.e., if $D_r(G) = r(d+1)$. By Proposition 1.1, we have the following.

PROPOSITION 3.1. *For any $d$-regular graphs $G$, $D_r(G) = r(d+1)$ for some $r \geq 2$ if and only if $D_1(G) = d+1$.*

Let $\mathcal{R}_d$ denote the set of all $d$-regular graphs $G$ with $D_1(G) = d+1$. By the above proposition, $\mathcal{R}_d$ is exactly the set of $d$-regular graphs $G$ with $D_r(G) = r(d+1)$ for some $r$. Our objective in this section is to test the membership in $\mathcal{R}_d$. If $G \in \mathcal{R}_d$, then $|V(G)| = a(d+1)$ for some integer $a$, since (1) any 1-configuration $f$ with $|C_f| = d+1$ assigns exactly one label to each vertex and (2) for each label $c$ in $C_f$, the set of vertices having $c$ corresponds to a perfect dominating set. It is clear that any $d$-regular graph with $d+1$ vertices (i.e., when $a = 1$) is a complete graph $K_{d+1}$, which obviously belongs to $\mathcal{R}_d$. We therefore assume that $|V(G)| = a(d+1)$ for some integer $a \geq 2$. This section is devoted to proving that the membership problem for $\mathcal{R}_d$ is solvable in polynomial time if $a = 2$, but is NP-complete if $a = 3$.

**3.1. The case $|V| = 2(d+1)$.** When $a = 2$, given a $d$-regular graph $G$, construct a graph $H(G) = (V(H), E(H))$ from $G$ so that $G \in \mathcal{R}_d$ if and only if $H(G)$ has a maximum matching of size $d+1$. We then solve the membership problem for $\mathcal{R}_d$ by solving the maximum matching problem for $H(G)$. Let $d_G(u,v)$ denote the distance between the vertices $u$ and $v$ in $G$. The following algorithm, TEST, tests if $G \in \mathcal{R}_d$ and finds a configuration $f$ with $d+1$ labels if the answer is positive by constructing a maximum matching for $H = H(G)$.

Algorithm TEST.

```
Input: $G$ {a $d$-regular graph with $2(d+1)$ vertices.}
Step 1: Construct a graph $H$: Let $V(H) = V(G)$ be its vertex set. For each pair
        $u, v \in V(H)$, let $\{u,v\} \in E(H)$ if $d_G(u,v) = 3$.
Step 2: Find a maximum matching $M$ for graph $H$.
Step 3: If the size of $M$ is $d+1$, then return a 1-configuration implied by $M$;[2]
        otherwise return NO.
```

To show the correctness of TEST, first let $f$ be a 1-configuration with $D_1(G) = d{+}1$ labels. For any $v \in V(G)$, let $N_G[v]$ be the set of vertices $u$ satisfying $d_G(u,v) \le 1$. By definition $|N_G[v]| = d+1$. Since $|V| = 2(d+1)$, for any vertex $v$, each vertex $u \in N_G[v]$ has a distinct label, and for any label $c \in C_f$, there are exactly two vertices $u$ and $v$ having $c$. Moreover, the distance (in $G$) between them is always 3. To see this, note that, if $d_G(u,v) \le 2$, there is a $w$ such that $N_G[w]$ contains both $u$ and $v$, which contradicts the uniqueness of labels assigned to vertices in $N_G[w]$, and that, if $d_G(u,v) \ge 4$, there is a vertex $w$ such that $N_G[w]$ contains neither $u$ nor $v$. Hence, the collection of edges $\{u,v\}$ in $H(G)$ connecting the vertices having the same label forms a matching of size $d+1$, which is in fact maximum since $|V(H)| = 2(d+1)$.

Next, given a maximum matching $M \subseteq E(H)$ of $H(G)$, construct a 1-configuration $f$ with $D_1(G) = d+1$ labels as follows: For each edge $e \in M$, we prepare label $c_e$. Then for each $e = \{u,v\} \in M$, we label $u$ and $v$ by label $c_e$. Since the set $\{u,v\}$ is a dominating set for $G$ (because $|N_G[u] \cup N_G[v]| = 2(d+1)$), this labeling is a 1-configuration with $d+1$ labels. Therefore, we conclude that TEST returns $M$ if and only if $D_1(G) = d+1$ and we can construct such a 1-configuration from $M$ as described above.

Finally, since each step of the above algorithm can be carried out in polynomial time, TEST runs in polynomial time.

Lemma 3.2. *If $G = (V,E)$ is a $d$-regular graph with $2(d+1)$ vertices, then we can test if $G \in \mathcal{R}_d$ in polynomial time.*

**3.2. The case $|V| = 3(d+1)$.** Unfortunately, it turns out that we cannot test the membership in $\mathcal{R}_d$ efficiently when $|V| = 3(d+1)$. This subsection is devoted to showing the NP-hardness of the problem. To this end, we transform 3SAT to our problem via a restricted version of the three-dimensional matching problem (3DM). That is, we first transform 3SAT to a variation of 3DM and then transform it to our problem. The first transformation (stated in Lemma 3.3) is similar to the transformation from 3SAT to 3DM in Garey and Johnson [8, pp. 50–53].

Let $U = \{u_1, u_2, \ldots, u_n\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ be the sets of variables and clauses, respectively, in a given instance of 3SAT.

Lemma 3.3. *For any given $C$, we can construct a set $M$ of 3-sets in $S_1 \times S_2 \times S_3$, where $S_1, S_2$, and $S_3$ are disjoint sets of elements (drawn from some set) of the same*

[2] $M$ partitions $V(H)$ into pairs of vertices. Assign the same label to each pair of vertices and distinct labels to different pairs.

*size, in polynomial time, that satisfies the following two conditions:* (1) *there is a subset $M'$ of $M$ that exactly covers $S_1 \cup S_2 \cup S_3$ if and only if $C$ is satisfiable and* (2) *for any $\{a,b,c\}, \{a,b,d\} \in M$, if $\{x,y,c\} \in M$ for some $x, y$, then $\{x,y,d\} \in M$.*

*Proof.* For the proof see Appendix A. □

Next, we proceed to the second transformation. Let $M$ be a set of 3-sets that satisfies the following property in Lemma 3.3: For any $\{a,b,c\}, \{a,b,d\} \in M$, if $\{x,y,c\} \in M$ for some $x, y$, then $\{x,y,d\} \in M$. We construct a $d$-regular graph $G = (V,E)$ with $3(d+1)$ vertices from $M$, such that $D_1(G) = d+1$ if and only if there is an exact cover $M'$ ($\subseteq M$) of $\bigcup_{\{a,b,c\}\in M}\{a,b,c\}$.

Lemma 3.4. *Let $M$ be a set of 3-sets in $X \times Y \times Z$, where $X, Y$, and $Z$ are disjoint sets of elements of an equal size, to satisfy the following condition: For any $\{a,b,c\}, \{a,b,d\} \in M$, if $\{x,y,c\} \in M$ for some $x, y$, then $\{x,y,d\} \in M$. Then, we can construct a $d$-regular graph $G$ with $3(d+1)$ vertices, such that $D_1(G) = d+1$ if and only if there is an exact cover $M'$ ($\subseteq M$) of $X \cup Y \cup Z$, in polynomial time.*

*Proof.* For the proof see Appendix B. □

By combining the previous two lemmas, we conclude that the problem of testing if $D_1(G) = d+1$ for any given $d$-regular graph $G$ with order $3(d+1)$ is NP-hard. The problem is obviously in the class *NP*.

Theorem 3.5. *The problem of deciding if a given $d$-regular graph with $3(d+1)$ vertices is in $\mathcal{R}_d$ is NP-complete.*

**4. 1-configuration of cubic graphs.** In the previous section, we discussed $d$-regular graphs and showed that testing $D_r(G) = r(d+1)$ (or equivalently, testing $D_1(G) = d+1$) is NP-complete, in general. In this section, we further restrict our attention to the case $d = 3$, i.e., to the cubic graphs, and investigate the problem of testing $D_1(G) = K$. Since $2 \le D_1(G) \le 4$ for cubic graphs, we first examine the case $K = 4$ and then proceed to the case $K = 2$ (or equivalently the case $K \ge 3$).

**4.1. Testing if $D_1(G) = 4$.** In this subsection, we prove that the problem of testing if $D_1(G) = 4$ is NP-complete even for cubic graphs. The problem is clearly in NP, so we concentrate on showing the NP-hardness of the problem. The proof is by a transformation from the 3-edge-coloring problem for cubic graphs, which is known to be NP-hard [11]. The transformation uses a graph $\mathcal{A}$, illustrated in Figure 4.1 as an "adaptor" graph. The figure shows an example of four-labeling of the degree-3 vertices. Although the labels of degree-1 vertices are not shown, it is easy to observe that the degree-1 vertices labeled $A$ must be white, those labeled $C$ must be black, and so on, to complete a 1-configuration with four labels. In general, $\mathcal{A}$ has the following property.

Property 1. *Consider any 1-configuration of $\mathcal{A}$ with four labels. Then*

1. *the two degree-1 vertices having the same label $A, B, C$, or $D$, have the same label, and*
2. *the two vertices adjacent to the degree-1 vertices having the same label have the same label.*

The transformation proceeds as follows: Let $G$ be any given cubic graph. Without loss of generality, we assume that $G$ is connected. We prepare four copies, $G_A, G_B, G_C$, and $G_D$, of $G$ and $|E(G)|$ copies of $\mathcal{A}$, each corresponding to an edge $e \in E(G)$. For each edge $e = \{u,v\} \in E(G)$, we have four copies $e_I = \{u_I, v_I\} (I \in \{A,B,C,D\})$ of $e$ and a copy $\mathcal{A}_e$ of $\mathcal{A}$. Then we remove the four edges $e_I$ and place $\mathcal{A}_e$, by identifying $u_I$ and $v_I$ with the two degree-1 vertices labeled $I$ in $\mathcal{A}_e$, as illustrated in Figure 4.2, where each rectangle represents a copy of the adaptor $\mathcal{A}$. The resulting graph $G'$ is clearly connected and cubic.

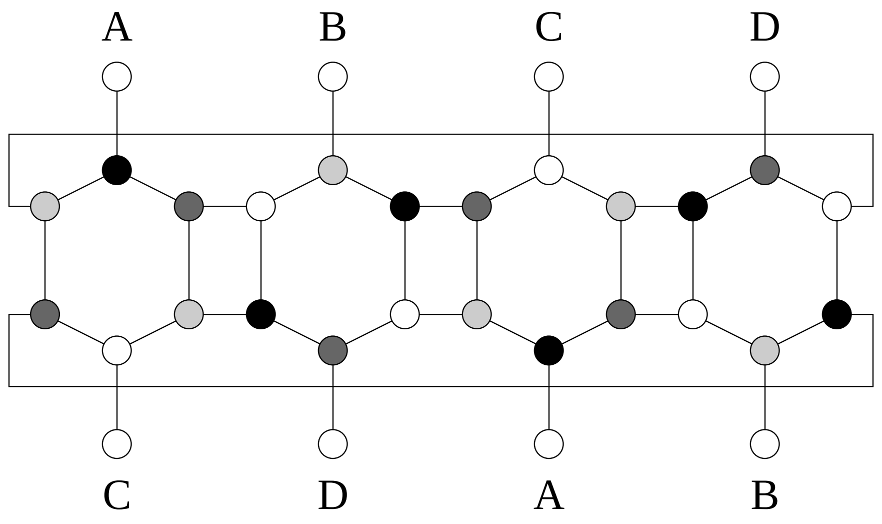


FIG. 4.1. *An "adaptor" $\mathcal{A}$ used in the transformation from the* 3*-edge-coloring problem for cubic graphs.*

LEMMA 4.1. *A cubic graph $G$ is* 3*-edge-colorable if and only if $D_1(G') = 4$, where $G'$ is the graph constructed from $G$ as described above.*

*Proof.* Suppose that $G'$ has a 1-configuration with four labels. Since $G'$ is connected, all vertices $u_A$ of copy $G_A$ of $G$ have the same label by Property 1. We label each edge $e = \{u, v\}$ by the label assigned to the vertex of $\mathcal{A}_e$ adjacent to $u_A$. (This labeling is well defined, since the two vertices of $\mathcal{A}_e$ adjacent to $u_A$ and $v_A$ have the same label by Property 1.) By definition, the resulting edge labeling uses three labels, and the three edges incident to $u$ have distinct labels.

Suppose that $G$ is 3-edge-colorable. We prepare four labels $A, B, C$, and $D$, and label $G'$ as follows: All vertices $u_I$ in copy $G_I$ of $G$ are labeled with $I$, for each $I \in \{A, B, C, D\}$. Consider any edge $e = \{u, v\} \in E(G)$, and let $c_I$ be the label assigned to the copy $e_I = \{u_I, u_v\}$ of $e$ in $G_I$. The two vertices of $\mathcal{A}_e$ adjacent to $u_I$ and $v_I$ are labeled with $c_I$. The other degree-3 vertices of $\mathcal{A}_e$ are labeled as illustrated in Figure 4.1. Since $c_I$ are distinct, the last part of labeling is actually possible. It is easy to check that the defined labeling is in fact a 1-configuration with four labels. Since $D_1(G') \leq 4$, we have $D_1(G) = 4$. $\square$

Hence, we have the following theorem.

THEOREM 4.2. *The problem of testing if $D_1(G) = 4$ for cubic graphs is NP-complete.*

**4.2. Testing if $D_1(G) \geq 3$.** In this subsection, we prove that the problem of testing if $D_1(G) \geq 3$ for cubic graphs is NP-complete. Again, the problem is obviously in NP. We concentrate on proving the NP-hardness of the problem.

LEMMA 4.3. *The problem of testing if $D_1(G) \geq 3$ for cubic graphs is NP-hard.*

*Proof.* We transform 3SAT to the problem at hand. To this end, we first transform 3SAT to the problem of testing if $D_1(G) \geq 3$ for graphs with minimum degree 2, and then transform the latter problem to our problem. The second transformation consists of two subtransformations. First we transform the testing problem for graphs with degree $\geq 2$ to that for graphs with degrees either 2 or 3, and then the latter problem is further transformed to the testing problem for cubic graphs. We start with the

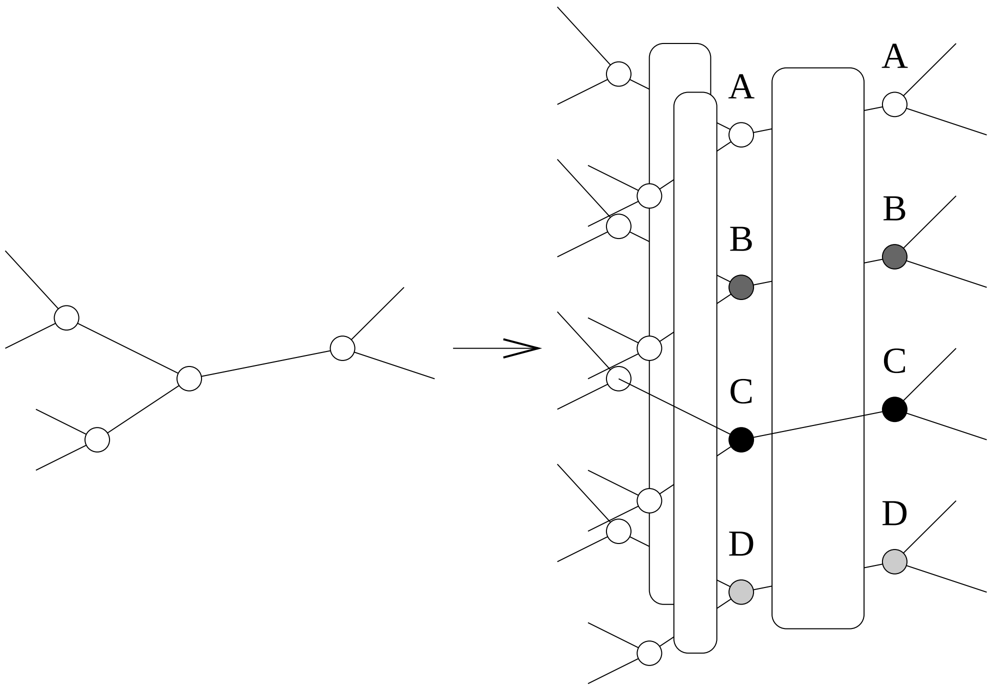


FIG. 4.2. *A transformation used in the proof of the NP-hardness of testing $D_1(G) = 4$ for cubic graphs.*

transformation of 3SAT to the testing problem for general graphs with degree $\geq 2$.

Let $U = \{u_1, u_2, \ldots, u_n\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ be sets of variables and clauses, respectively, arbitrarily given as an instance of 3SAT. Let $G = (V, E)$ be a graph defined from the instance as follows: $V = V_0 \cup V_1 \cup V_2$, where

$$\begin{aligned} V_0 &= \{s, t_1, t_2, t_3\}, \\ V_1 &= \{a_i^1, a_i^2 \mid 1 \leq i \leq m\}, \\ V_2 &= \{b_j^1, b_j^2, b_j^3 \mid 1 \leq j \leq n\}, \end{aligned}$$

and connect vertices in $V$ as follows:

- $\{t_1, t_2\}, \{t_2, t_3\}, \{t_3, t_1\} \in E$.
- For all $1 \leq i \leq m$, let $\{s, a_i^1\}, \{a_i^1, a_i^2\} \in E$.
- For all $1 \leq j \leq n$ and all $1 \leq k \leq 3$, let $\{t_k, b_j^1\}, \{b_j^1, b_j^2\}, \{b_j^2, b_j^3\}, \{b_j^3, t_k\} \in E$.
- For all $1 \leq i \leq m$, if clause $c_i$ contains a positive (respectively, negative) literal of variable $u_j$, then let $\{a_i^2, b_j^1\} \in E$ (respectively, $\{a_i^2, b_j^3\} \in E$).

For example, Figure 4.3 shows the graph $G$ to which an instance $(u_1 + u_2 + u_3)(u_1 + \overline{u}_2 + \overline{u}_3)(\overline{u}_1 + \overline{u}_2 + u_3)$ of 3SAT has been transformed. Note that in $G$, only vertices $a_i^1$ $(1 \leq i \leq m)$ and $b_j^2$ $(1 \leq j \leq n)$ have degree 2.

Since the minimum degree of $G$ is 2, we have $D_1(G) \leq 3$. We first prove that if the given formula is satisfiable, then the resulting graph $G$ has a 1-configuration with three labels, and hence $D_1(G) = 3$. Fix an assignment that satisfies the formula. Let $U_1$ $(\subseteq U)$ be the set of variables to which value 1 is assigned. For clause $c_m$, there is at least one variable $u_\ell$ which contributes to satisfying it, i.e., $u_\ell \in U_1$ if and only if $c_m$ contains $u_\ell$ as a positive literal.

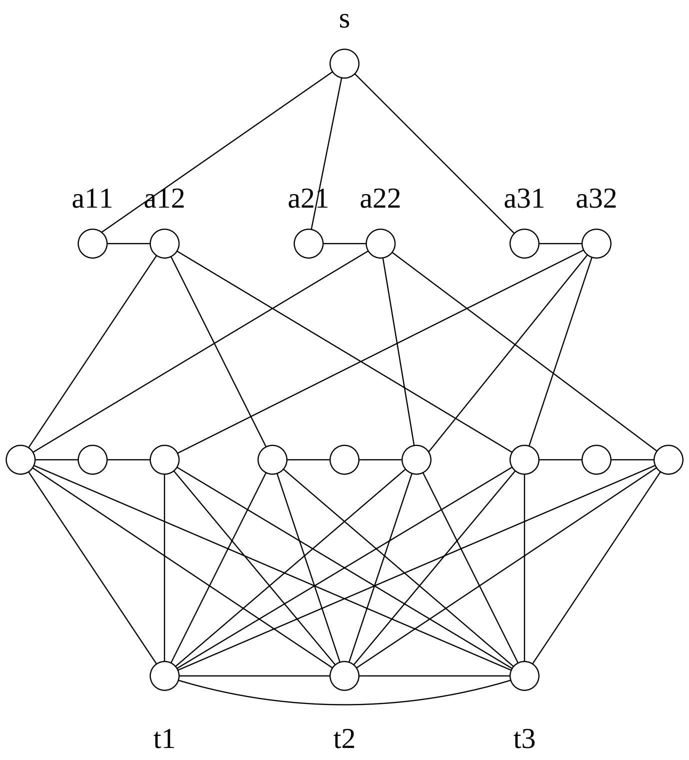


FIG. 4.3. *The graph $G$ to which the instance $(u_1 + u_2 + u_3)(u_1 + \overline{u}_2 + \overline{u}_3)(\overline{u}_1 + \overline{u}_2 + u_3)$ of 3SAT is transformed.*

Construct a labeling function $f$ which maps vertices in $V$ to three labels $\{1, 2, 3\}$ as follows:

- $f(s) = 1$, $f(t_1) = 1$, $f(t_2) = 2$, and $f(t_3) = 3$.
  Thus, for any pair of a vertex $u \in \{b_j^1, b_j^3 \mid 1 \le j \le n\} \cup \{t_k \mid 1 \le k \le 3\}$ and a label $c \in \{1, 2, 3\}$, there is a vertex $v$ with label $c$ that dominates $u$, i.e., that satisfies $d_G(u, v) \le 1$.
- – For $1 \le i \le m - 1$, we have $f(a_i^1) = 2$ and $f(a_i^2) = 3$.
  – $f(a_m^1) = 3$ and $f(a_m^2) = 2$.
  Thus, for any pair of a vertex $u \in \{a_i^1 \mid 1 \le i \le m\}$ and a label $c \in \{1, 2, 3\}$, there is a vertex $v$ with label $c$ that dominates $u$.
- – For each $u_j \in U_1 \setminus \{u_\ell\}$, we have $f(b_j^1) = 1$, $f(b_j^2) = 2$, and $f(b_j^3) = 3$.
  – For each $u_j \in U \setminus (U_1 \cup \{u_\ell\})$, we have $f(b_j^1) = 3$, $f(b_j^2) = 2$, and $f(b_j^3) = 1$.
  – If $u_\ell \in U_1$, then $f(b_\ell^1) = 1$, $f(b_\ell^2) = 3$, and $f(b_\ell^3) = 2$. If $u_\ell \notin U_1$, then $f(b_\ell^1) = 2$, $f(b_\ell^2) = 3$, and $f(b_\ell^3) = 1$.
  Thus, for any pair of a vertex $u \in \{b_j^2 \mid 1 \le j \le n\}$ and a label $c \in \{1, 2, 3\}$, there is a vertex $v$ with label $c$ that dominates $u$.

Vertex $a_m^2$ (with label 2) is dominated both by $a_m^1$ with label 3 and by a vertex (either $b_m^1$ or $b_m^3$) with label 1, since if $u_\ell \in U_1$, then $f(b_\ell^1) = 1$, and otherwise $f(b_\ell^3) = 1$. Similarly, vertex $a_j^2$ (with label 3) is dominated by both $a_j^1$ with label 2 and a vertex ($b_j^1$ or $b_j^3$) with label 1, since for any clause $c_j$ $(1 \le j \le m - 1)$, there is at least one variable which contributes to satisfying $c_j$. Thus, $f$ is a 1-configuration

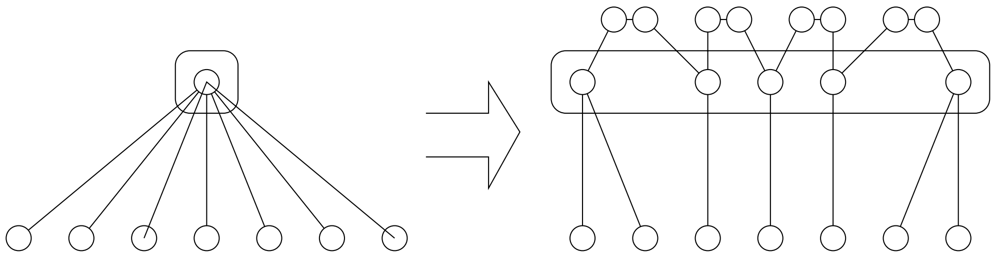


FIG. 4.4. *The vertex split operation used in the transformation.*

of $G$ with three labels.

Next, we prove that if $G$ has a 1-configuration with three labels, then the given formula is satisfiable. Fix a 1-configuration $f$ of $G$ with three labels $\{1, 2, 3\}$. Without loss of generality, we assume $f(s) = 1$. Since $|C_f| = 3$, we have $f(a_i^1) \neq 1$ and $f(a_i^2) \neq 1$ for all $i$. Hence for all $i$, there must exist at least one vertex ($\neq a_i^1$) with label 1 adjacent to vertex $a_i^2$. Since $f(b_j^1) \neq f(b_j^3)$ must hold (otherwise, there would be a label such that $b_j^2$ had no neighbor with that label), by assigning value 1 (respectively, 0) to variables $u_j$ such that $f(b_j^1) = 1$ (respectively, $f(b_j^3) = 1$), we can find an assignment satisfying the given formula. Thus, 3SAT has been transformed to the problem of testing if $D_1(G) \geq 3$ for graphs with minimum degree $\geq 2$.

Then we proceed to the second transformation: we transform the problem of testing if $D_1(G) \geq 3$ for graphs with degree $\geq 2$ to the one for cubic graphs. As explained, the transformation consists of two subtransformations. We first transform graphs with degree $\geq 2$ to graphs with degrees either 2 or 3.

Let $G$ be a given graph with degree $\geq 2$. First, we (repeatedly) replace each vertex of degree $d$ ($\geq 4$) with a path with $3d - 8$ vertices, as shown in Figure 4.4. Let $G'$ be the resulting graph. It is easy to check that the degrees of $V(G')$ are either 2 or 3. As shown below, $G'$ has a 1-configuration with three labels if and only if $G$ has a 1-configuration with three labels. First, if $G$ has a 1-configuration with three labels, obviously, so does $G'$. Suppose now that $G'$ has a 1-configuration with three labels. We fix such a 1-configuration $f'$ of $G'$. Then the $(3i + 1)$st ($i = 0, 1, \ldots, d - 3$) vertex on the path (i.e., the vertices enclosed in the rectangle in the right-hand side of the figure) must have the same label since, otherwise, the two vertices between the $(3i + 1)$st and $(3i + 4)$th vertices cannot be dominated by all three labels. Hence, we can construct a 1-configuration $f$ of $G$ with three labels from $f'$ as follows: If a vertex $v$ in $G$ has degree $d \leq 3$, then label it with the label assigned to the corresponding vertex in $G'$; otherwise (i.e., if $d \geq 4$), then label it with the label assigned to the $(3i + 1)$st ($i = 0, 1, \ldots, d - 3$) vertex in the corresponding path.

Finally, we further transform graphs with degrees either 2 or 3 to cubic graphs. Figure 4.5 illustrates the adaptor used in the transformation, and its three-labeling. We use the following property, which can be proved by straightforward (but tedious), exhaustive checking.

PROPERTY 2. *In any* 1*-configuration of the adaptor in Figure* 4.5 *with three labels,*

1. *the two vertices $A'$ and $B'$ have the same label, and*
2. *the two vertices $A$ and $B$ are not assigned the same label as $A'$ and $B'$.*

Let $G'$ be a given graph with degrees either 2 or 3. To construct a cubic graph $G''$ from $G'$, we replace each degree-2 vertex $u$ with the adaptor by splitting $u$ into two

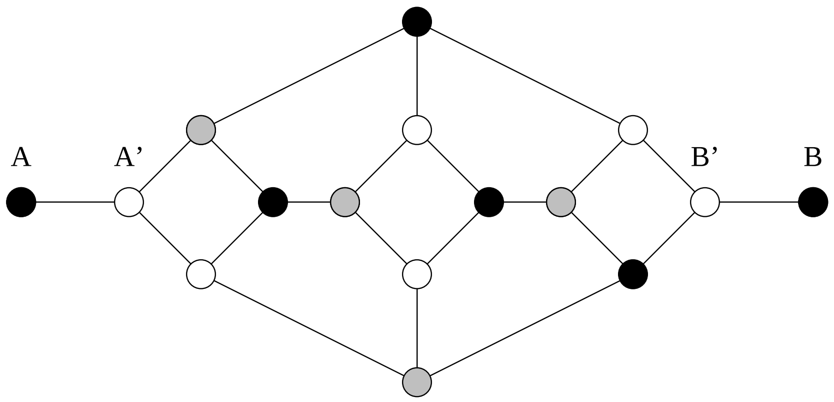


FIG. 4.5. *The adaptor used in the transformation from $G'$ to $G''$ and its three-labeling.*

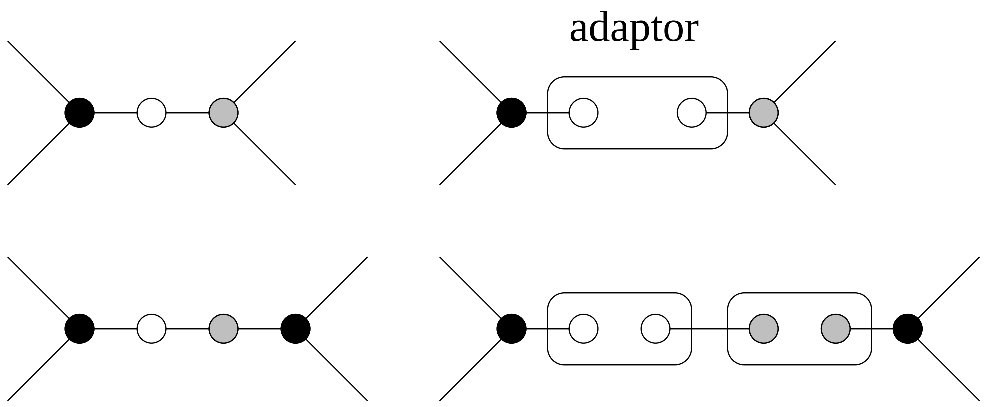


FIG. 4.6. *Transformation from $G'$ to $G''$.*

vertices $u_A$ and $u_B$ and identifying $u_A$ with $A'$ and $u_B$ with $B'$. See Figure 4.6 for an illustration. It is easy to show that $G''$ is cubic.

By using Property 2, we can easily show that $G'$ has a 1-configuration with three labels if and only if $G''$ has a 1-configuration with three labels.

Now we conclude that 3SAT is polynomially transformable to the problem of testing if $D_1(G) \geq 3$ for cubic graphs $G$. □

THEOREM 4.4. *The problem of testing if $D_1(G) \geq 3$ for cubic graphs $G$ is NP-complete, or equivalently, the problem of testing if $D_1(G) = 2$ for cubic graphs $G$, is co-NP-complete.*

**5. 2-configurations for cubic graphs.** In the previous section, we proved that testing if $D_1(G) = 2$ is co-NP-complete, which implies that there are infinitely many cubic graphs with $D_1(G) = 2$. The objective of this section is to show that $D_2(G) \geq 5$ holds for any cubic graph $G$. We start with the investigation of 2-configurations for cycles and proceed to that for cubic graphs.

**5.1. Labeling cycles.** This subsection considers 2-configurations for cycles and shows that every cycle $C_n$ of $n$ vertices, except for $n = 4$ or 7, has a 2-configuration with five labels. In what follows, we assume that the vertices in cycle $C_n$ are arranged

TABLE 1
*Case $n = 3m$.*

| Vertex | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1st label | a | d | c | a | d | c |
| 2nd label | b | e |  | b | e |  |

TABLE 2
*Case $n = 3m + 1$.*

| Vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1st label | a | d | c | a | a | c | b | a | d | b | a | d | c |
| 2nd label | b | e | d | b | e | d | c | e | e | c | b | e |  |

TABLE 3
*Case $n = 3m + 2$.*

| Vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1st label | a | d | a | b | c | a | d | c |
| 2nd label | b | e | c | d | e | b | e |  |

in the order $0, 1, \ldots, n-1$, and vertices $n-1$ and 0 are connected by an edge. The five labels we use are represented by $\{a, b, c, d, e\}$.

THEOREM 5.1. *A cycle $C_n$ of order $n$ has a 2-configuration with five labels if and only if $n \neq 4, 7$.*

*Proof.* We consider the following three cases separately.

*Case* 1 [$n = 3m$]. Since $D_1(C_{3m}) = 3$ is obvious, we have $D_2(C_{3m}) \geq 6$ by Proposition 1.1. Table 1 shows an example of a 2-configuration for a cycle of length 6 with five labels. For $m \geq 1$, in general, the first three columns can be repeated $m/3$ times.

*Case* 2 [$n = 3m + 1$]. $C_4$ $(m = 1)$ has no 2-configuration with five labels. To see this, observe that four vertices can accommodate eight labels altogether, but the size of any dominating set for $C_4$ is at least 2, which implies that every label must appear in at least two vertices. Hence, $C_4$ can have a 2-configuration with at most $\lfloor 8/2 \rfloor = 4 < 5$ labels. Similarly, we can show that $C_7$ $(m = 2)$ has no 2-configuration with five labels. To see this, note that seven vertices can accommodate 14 labels altogether, but the size of any dominating set for $C_7$ is at least 3, which implies that every label must appear in at least three vertices. So, $C_7$ can have a 2-configuration with at most $\lfloor 14/3 \rfloor = 4 < 5$ labels.

For $m \geq 3$, repeating the last three columns of Table 2 $(m-3)/3$ times produces a 2-configuration for $C_{3m+1}$ with five labels.

*Case* 3 [$n = 3m + 2$]. For $m \geq 1$, repeating the last three columns of Table 3 $(m-1)/3$ times produces a 2-configuration for $C_{3m+2}$ with five labels. □

In view of the above theorem, we call $C_4$ and $C_7$ *special cycles.* Note that, for all larger cycles, except for $C_{10}$, we can use a repetitive labeling pattern, $\{a, b\}$, $\{d, e\}$, $\{c\}$, to realize a 2-configuration with five labels. This fact will be used in the proof of Lemma 5.3 below.

**5.2. Labeling cubic graphs.** In this subsection, we show $D_2(G) \geq 5$ for any cubic graph $G$. Without loss of generality, we may assume that the given cubic graph $G$ is connected and has more than four vertices, because the only cubic graph with four vertices, $K_4$, clearly has a 2-configuration with eight labels.

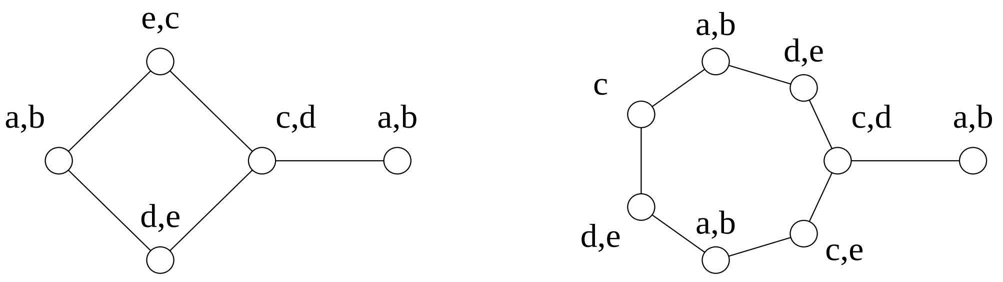


FIG. 5.1. *Five-labelings for $C_4$ and $C_7$ in graph $G'$.*

First we consider the case where $G$ has no bridges.[3] It is well known that any cubic graph has an even number of vertices, and any bridgeless cubic graph $G$ is decomposable into a 2-factor and a 1-factor [16]. Fix an arbitrary factorization (into a 1-factor and a 2-factor) of $G$. If the 2-factor consists of a single Hamiltonian cycle $C_n$, then $G$ has a 2-configuration with five labels, since (1) $n \neq 7$ ($n$ is even), (2) $K_4$ is the only cubic graph of order 4, and (3) each nonspecial cycle $C_n$ is five-labelable by Theorem 5.1.

If the 2-factor consists of more than one cycle, consider a minimal (in terms of the number of edges) connected subgraph $G'$ of $G$ constructed from $G$ by repeatedly removing a matching edge (i.e., an edge forming the 1-factor) as long as the resulting graph does not become disconnected. We call each edge in the 1-factor (of $G$) remaining in $G'$ a *supporting edge.* In the following, we construct a 2-configuration for $G'$ with five labels. Since $G'$ is a subgraph of $G$, the labeling is also a 2-configuration for $G$ with five labels.

A vertex $u$ in a graph $G$ with a five-labeling function $f$ is said to be *unsaturated*, if

$$|\cup_{w \in N_G[u]} f(w)| \leq 4,$$

where $N_G[u]$ was defined in subsection 3.1. If $w \in N_G[u]$, we say that each *label* in $f(w)$ dominates the vertex $u$.

By Theorem 5.1, any nonspecial cycle $C_m$ has a 2-configuration with five labels. Further, for any given vertex $v$ and two labels $a, b$, clearly, we can label $C_m$ with five labels in such a way that $a$ and $b$ are assigned to $v$. Unfortunately, special cycles, $C_4$ and $C_7$, have no 2-configuration with five labels. However, we can label them in such a way that all but one vertex is unsaturated. For example, if we ignore the "handles" of the two graphs in Figure 5.1, they represent five-labelings for the special cycles, $C_4$ and $C_7$, respectively. Note that only the vertex labeled by "c,d" is unsaturated in each cycle. Also, three (out of five) labels dominate these vertices (if the handles are ignored).

Consider a graph consisting of two special cycles connected by a bridge. It is easy to see that any such graph has a 2-configuration with five labels. See Figure 5.2 for an example. In Figure 5.2, the two end vertices of the bridge "borrow" missing labels from each other. In general, consider a 2-configuration with five labels, $C$, for a graph consisting of two components connected by a bridge, such that, if the bridge is removed, at least one of the end vertices of the bridge becomes unsaturated within the component to which it belongs. In this case, we say that the two components

[3]A bridge is an edge whose removal disconnects the graph.

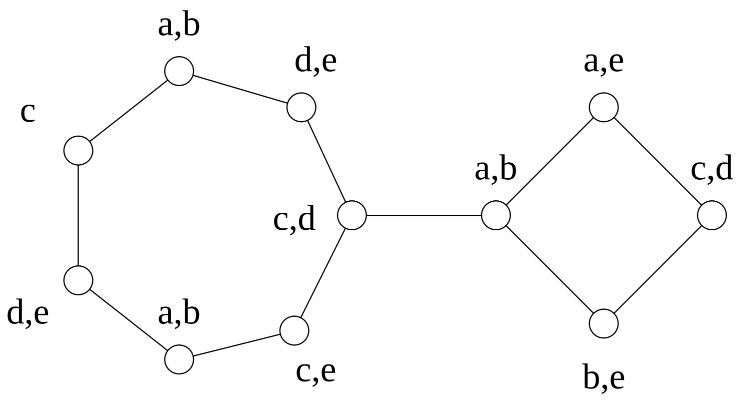


FIG. 5.2. *Complementary labeling for two special cycles connected by a bridge.*

have *complementary labeling* in $C$. We will make use of complementary labeling in the following algorithm.

ALGORITHM LABEL.
Input: Bridgeless cubic graph $G$.
Output: A 2-configuration for $G$ with five labels.

1. Factor $G$ into a 2-factor and a 1-factor.
2. Remove an edge belonging to the 1-factor if it does not disconnect the graph. Repeat this operation as long as it is possible, and name the resulting graph $G'$.
3. Coalesce all vertices in each cycle of $G'$ into a single vertex, and call the resulting tree $T$. Fix a vertex in $T$ as its root.
4. Label the cycle $C_m$ in $G'$ corresponding to the root of $T$ as follows: Let $u$ be any vertex in $C_m$ incident to a supporting edge $\{u, v\}$. If $C_m$ is not special, label it using an arbitrary 2-configuration with five labels. If $C_m$ is special, label it in such a way that $u$ becomes the only unsaturated vertex. In either case, let $a$ and $b$ be the labels that are assigned to $u$ and let the labels that do not dominate $u$, if any, belong to $\{d, e\}$. We now consider that the root of $T$ has been "processed."
5. While there is a vertex of $T$ which has not been processed do
   (a) Pick an unprocessed vertex of $T$ whose parent has been processed, and let $C_m$ be the cycle in $G'$ that corresponds to this vertex. Let $\{u, v\}$ be the supporting edge such that $u$ and $v$ belong to the cycle corresponding to the parent and $C_m$, respectively.
   (b) Let $a$ and $b$ be the labels that have been assigned to $u$ and let the labels that do not dominate $u$, if any, belong to $\{d, e\}$. If $C_m$ is special, label it in such a way that $v$ becomes the unsaturated vertex labeled by $d$ and $e$ and not dominated by $a$ and $b$. Otherwise, use a 2-configuration for $C_m$ with five labels such that $v$ is labeled by $d$ and $e$. The vertex of $T$ representing $C_m$ has now been processed.

Since each step of LABEL can be implemented in polynomial time, we have proved the following lemma.

LEMMA 5.2. *Given any bridgeless cubic graph $G$, Algorithm* LABEL *finds a* 2*-configuration for $G$ with five labels in polynomial time.*

Next we consider the general case where the given cubic graph may contain

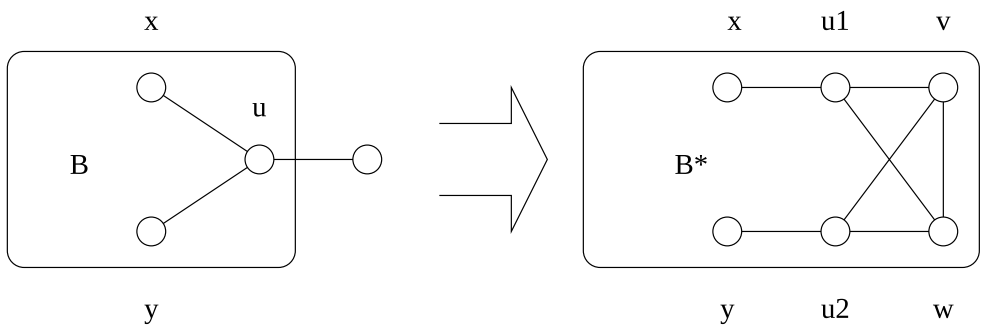


FIG. 5.3. *Construction of $B^*$ from $B$.*

bridges. The proof of the following lemmas presented below.

LEMMA 5.3. *Let $B$ be any maximal biconnected component of a cubic graph $G$. We can five-label $B$ in such a way that*

(a) *every unsaturated vertex $u$ in it is incident to a bridge in $G$, and*

(b) *at most two (out of five) labels do not dominate $u$.*

For the time being, we assume that the above lemma is correct and present an algorithm for finding a 2-configuration with five labels, which is a modification of LABEL. We use maximal biconnected components and bridges instead of cycles and supporting edges, respectively. We construct a tree $T$ from $G$ by coalescing all vertices in each maximal biconnected component into a single vertex. We pick an arbitrary vertex of $T$ as its root and label the corresponding biconnected component as stated in Lemma 5.3.

Suppose now that we are labeling a maximal biconnected component $B$ represented by a vertex $v_B$ in $T$, such that the biconnected component $A$ represented by $v_B$'s parent, $v_A$, has already been labeled. Let $\{u, v\}$ be the bridge connecting $A$ and $B$ such that $u$ and $v$ belong to $A$ and $B$, respectively. Let $a$ and $b$ be the labels assigned to $u$, and, if $u$ is unsaturated, the labels not dominating $u$ belong to the set $\{d, e\}$. (If $u$ is saturated, we take two arbitrary labels as $d$ and $e$.) We label $B$ in such a way that (1) $c$ and $d$ are assigned to $v$, (2) every unsaturated vertex is incident to a bridge, and (3) if $v$ is unsaturated, the labels not dominating $v$ belong to the set $\{a, b\}$. This is always possible according to Lemma 5.3. It is easy to see that, once all vertices are labeled as above, the resulting labeling is a 2-configuration for $G$ with five labels.

*Proof of Lemma* 5.3. We modify $B$ to create a cubic graph $B^*$. Let $k$ be the number of bridges to which $B$ is incident. First consider the case where $k \geq 2$. We prepare two copies of $B$ and connect them by introducing a set of edges, each connecting two copies of the same degree-2 vertex in $B$. The resulting graph $B^*$ is clearly cubic and bridgeless, so, by Lemma 5.2, it has a 2-configuration $f$ with five labels. It is not difficult to see that $f$ restricted to a copy of $B$ can be made to satisfy the conditions of Lemma 5.3.

Next, consider the case $k = 1$. Let $u$ be the unique degree-2 vertex of $B$, and $x$ and $y$ be the two vertices adjacent to $u$. (See the graph on the left in Figure 5.3.) From $B$, we construct a new cubic graph $B^*$ by splitting $u$ into two vertices, $u_1$ and $u_2$, and by introducing two vertices, $v$ and $w$, (and a few edges) as illustrated in Figure 5.3.

Graph $B^*$ is clearly cubic and bridgeless. Therefore, we can decompose $B^*$ into

TABLE 4
*A five-labeling for $B^*$.*

| Vertex | x | $u_1$ | w | $u_2$ | v |
|---|---|---|---|---|---|
| 1st label | c | a | b | c | a |
| 2nd label | d | b | e | d | e |

TABLE 5
*A five-labeling for $B^*$.*

| Vertex | x | $u_1$ | w | v | $u_2$ | y | z |
|---|---|---|---|---|---|---|---|
| 1st label | a | a | c | d | a | b | d |
| 2nd label | c | b |  | e | b | c | e |

a 2-factor and a 1-factor and apply LABEL. We consider two cases.

*Case* 1. Edge $e_1 = \{u_1, x\}$ belongs to the 1-factor. In this case $e_2 = \{u_2, y\}$ and $\{v, w\}$ also belong to the 1-factor. In step 2 of LABEL, if we remove $e_2$ and $\{v, w\}$, $e_1$ becomes a supporting edge, and the two remaining edges incident to $y$ belong to the 2-factor. Based on the above decomposition, we can find a 2-configuration for $B^*$ with five labels. See Table 4 for an example, which was obtained after making the cycle $(u_1, w, u_2, v)$ the root of $T$ in step 3 of LABEL.

$x$ may be unsaturated, since $e_1$ is a supporting edge, but $y$ is saturated, since it is not incident to a supporting edge. (Label assignment to $y$ is not shown in Table 4.) We can now find a labeling for $B$ satisfying the conditions of Lemma 5.3 as follows: Label every vertex in $B$, except $u$, with exactly the same labels as the corresponding vertex in $B^*$, and label $u$ with the same two labels as assigned to $u_1$. In the resulting labeling for $B$, $u$ is unsaturated, being not dominated by $e$.

*Case* 2. Edge $e_1$ belongs to the 2-factor. In this case the cycle, $C$, of the 2-factor containing $e_1$ also contains the edges $(u_1, w)$, $(w, v)$, $(v, u_2)$, and $e_2$. In the case where $C$ is the special cycle of length 7, Table 5 shows a possible 2-configuration for $B^*$ with five labels that assigns the same two labels, $a$ and $b$, to both $u_1$ and $u_2$.

Note that labels $\{d, e\}$ do not dominate $u_1$. We can now find a labeling for $B$ satisfying the conditions of Lemma 5.3 as follows: Label every vertex in $B$, except $u$, with exactly the same labels as the corresponding vertex in $B^*$, and label $u$ with the same two labels as assigned to $u_1$.

If $C$ is a larger cycle, then we can always assign $\{a, b\}$ to $u_1$ and $u_2$, $\{d, e\}$ to $w$, and $\{c\}$ to $v$, since the pattern of $\{a, b\}$,$\{d, e\}$,$\{c\}$ is repeated in Tables 1, 2, and 3. If $C$ is of length 10, then $\{a, b\}$,$\{d, e\}$,$\{c, d\}$ can be assigned instead to $u_1$ (and $u_2$), $w$, and $v$, respectively (see columns 0–3 of Table 2). This completes the proof. $\square$

Now, we can state the following theorem.

THEOREM 5.4. *Any cubic graph has a 2-configuration with five labels, and such a labeling can be found in polynomial time.*

COROLLARY 5.5. *Any cubic graph has an $r$-configuration with $\lfloor 5r/2 \rfloor$ labels, and such a labeling can be found in polynomial time.*

*Proof.* By Proposition 2.4, if $r$ is odd,

$$D_r(G) \geq \left(\frac{r-1}{2}\right) D_2(G) + D_1(G) \;=\; (5/2)(r-1) + 2 \;=\; (5/2)r - 1/2,$$

and if $r$ is even,

$$D_r(G) \;\geq\; \frac{r}{2} D_2(G) \;=\; (5/2)r. \qquad \square$$

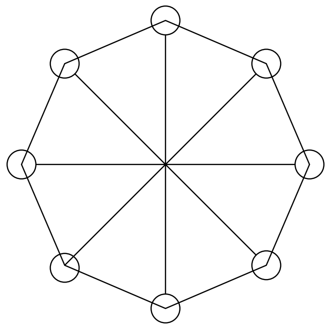


FIG. 6.1. *A cubic graph that has no* 2*-configuration with six labels.*

**6. Concluding remarks.** A problem of interest still open is determining the complexity of finding a 2-configuration for a given cubic graph with six or seven labels. One promising approach to the six-labeling problem would be to extend the argument in section 5 in such a way that we take into account labelings created from $T$ whose vertices correspond to cycles with chords. We conjecture that the approach will work well for sufficiently large cubic graphs. More specifically, we conjecture that any connected cubic graph, except for some small special graphs, has a 2-configuration with six labels. Figure 6.1 illustrates an exception we have found so far. To see that the graph has no 2-configuration with six labels, observe that there are at most 16 places to label, but the size of any dominating set for the graph is at least 3.

Another open problem is examining the effect of increasing $r$ more generally, e.g., to give a nontrivial lower bound on the number of labels which is assignable within the constraint of $r$-configuration for $r \geq 2$.

**Appendix A. Proof of Lemma 3.3.** We construct a set $M$ of 3-sets so as to satisfy the conditions of the lemma. Set $M$ is partitioned into three subsets $M_1, M_2$, and $M_3$ according to their intended functions. An element $a$ in a 3-set in $M_i$ (for $i = 1, 2$, or 3) is said to be *internal* with respect to $M_i$ if $M_j$ $(j \neq i)$ contains no 3-set containing $a$; otherwise, it is said to be *external.* In our construction, we use the following eight disjoint sets of external elements:

$$\begin{array}{rclrcl}
X &=& \bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\{x_i[j]\}, & \tilde{X} &=& \bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\{\tilde{x}_i[j]\},\\
Y &=& \bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\{y_i[j]\}, & \tilde{Y} &=& \bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\{\tilde{y}_i[j]\},\\
X' &=& \bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\{x'_i[j]\}, & \tilde{X}' &=& \bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\{\tilde{x}'_i[j]\},\\
Y' &=& \bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\{y'_i[j]\}, & \tilde{Y}' &=& \bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\{\tilde{y}'_i[j]\}.
\end{array}$$

Note that each set is of size $mn$.

Each 3-set in $M_1$ corresponds to a variable in $U = \{u_1, \dots, u_n\}$ and consists of one external and two internal elements. For $k = 1,2,3$, and 4, let $A^k = \{a_i^k[j] \mid 1 \leq i \leq n, 1 \leq j \leq m\}$ be a set of internal elements with respect to $M_1$. Note that $|A^k| = mn$ for each $k$. Set $M_1$ is then defined as follows:

$$M_1 \;=\; P^t \cup P^f \cup Q^t \cup Q^f,$$

where

$$\begin{array}{rcl} P^t & = & \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} \left\{ \{\tilde{x}_i[j], a_i^1[j], a_i^2[j]\} \right\}, \\ P^f & = & \bigcup_{i=1}^{n} \bigcup_{j=1}^{m-1} \left\{ \{x_i[j], a_i^1[j+1], a_i^2[j]\} \right\} \cup \left\{ \{x_i[m], a_i^1[1], a_i^2[m]\} \right\}, \\ Q^t & = & \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} \left\{ \{\tilde{y}_i[j], a_i^3[j], a_i^4[j]\} \right\}, \\ Q^f & = & \bigcup_{i=1}^{n} \bigcup_{j=1}^{m-1} \left\{ \{y_i[j], a_i^3[j+1], a_i^4[j]\} \right\} \cup \left\{ \{y_i[m], a_i^3[1], a_i^4[m]\} \right\}. \end{array}$$

Since $A' = A^1 \cup A^2$ is also a set of internal elements, $A'$ is exactly covered by a matching $M'$ ($\subseteq M$) only if, for any $1 \le i \le n$, $M'$ contains all 3-sets intersecting with $\{x_i[j] \mid 1 \le j \le m\}$ or $M'$ contains all 3-sets intersecting with $\{\tilde{x}_i[j] \mid 1 \le j \le m\}$. A similar statement holds for $A^3 \cup A^4$. (It implies that an exact cover corresponds to a consistent assignment to variables, i.e., all occurrences of a variable should take the same value.)

Each 3-set in $M_2$ corresponds to a single clause $c_j \in C$. We associate the external element $*_i[j]$ with variable $u_i$ and introduce new internal elements $b^1[j]$ and $b^2[j]$ for $1 \le j \le m$. Let $B^k = \{b^k[j] \mid 1 \le j \le m\}$ for $k = 1, 2$. Note that $|B^k| = m$ for each $k$. Subset $M_2$ is defined as follows:

$$M_2 = R^t \cup R^f,$$

where

$$\begin{array}{rcl} R^t & = & \left\{ \{x_i[j], b^1[j], x_i'[j]\}, \{y_i[j], b^2[j], y_i'[j]\} \mid u_i \in c_j \right\}, \\ R^f & = & \left\{ \{\tilde{x}_i[j], b^1[j], \tilde{x}_i'[j]\}, \{\tilde{y}_i[j], b^2[j], \tilde{y}_i'[j]\} \mid \tilde{u}_i \in c_j \right\}. \end{array}$$

Note that the set $B^1 \cup B^2$ is exactly covered by some $M'$ ($\subseteq M$) only if all clauses in $C$ are satisfiable.

Set $M_3$ is used to associate a satisfiable assignment of the given formula with an *exact* cover of the underlying set $\bigcup_{\{a,b,c\} \in M} \{a, b, c\}$. $M_3$ is defined as

$$M_3 = O^1 \cup O^2,$$

where

$$\begin{array}{rcl} O^1 & = & \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} \bigcup_{\ell=1}^{mn-m} \left\{ \{x_i[j], t^1[\ell], y_i[j]\}, \{\tilde{x}_i[j], t^1[\ell], \tilde{y}_i[j]\} \right\}, \\ O^2 & = & \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} \bigcup_{\ell=1}^{2mn-m} \left\{ \{x_i'[j], t^2[\ell], y_i'[j]\}, \{\tilde{x}_i'[j], t^2[\ell], \tilde{y}_i'[j]\} \right\}. \end{array}$$

Let $T^1 = \{t^1[\ell] \mid 1 \le \ell \le mn - m\}$ and $T^2 = \{t^2[\ell] \mid 1 \le \ell \le 2mn - m\}$. $T_1 \cup T_2$ is a set of internal elements with respect to $M_3$. Note that $|T^1| = mn - m$ and $|T^2| = 2mn - m$.

By the above construction, it is easy to see that $M = M_1 \cup M_2 \cup M_3$ is a set of 3-sets such that (1) there is a subset $M'$ of $M$ which exactly covers $\bigcup_{\{a,b,c\} \in M} \{a, b, c\}$ if and only if $C$ is satisfiable and (2) for any $\{a, b, c\}, \{a, b, d\} \in M$, if $\{x, y, c\} \in M$ for some $x, y$, then $\{x, y, d\}$ is also in $M$.

Finally, let us show that the underlying set $\bigcup_{\{a,b,c\} \in M} \{a, b, c\}$ is partitioned into three subsets $S_1, S_2$, and $S_3$ of equal size, such that $M \subseteq S_1 \times S_2 \times S_3$. Let

$$\begin{array}{l} S_1 = X \cup \tilde{X} \cup Y' \cup \tilde{Y}' \cup A^4, \\ S_2 = T^1 \cup T^2 \cup A^1 \cup A^3 \cup B^1 \cup B^2, \\ S_3 = X' \cup \tilde{X}' \cup Y \cup \tilde{Y} \cup A^2. \end{array}$$

Clearly, $\bigcup_{\{a,b,c\}\in M}\{a,b,c\} = S_1 \cup S_2 \cup S_3$ and $M \subseteq S_1 \times S_2 \times S_3$. Since each set of external elements has size $mn$ and $|A^k| = mn$ for each $k$, $|S_1| = |S_2| = 4mn + mn = 5mn$. Further, since $|T^1| = mn - m$, $|T^2| = 2mn - m$, and $|B^k| = m$ for each $k$, we have $|S_3| = (3mn - 2m) + 2mn + 2m = 5mn$. Hence the lemma follows.

**Appendix B. Proof of Lemma 3.4.** We construct a graph $G = (V, E)$ so as to satisfy the conditions of the lemma. Without loss of generality, let $X = Y = Z = \{1, 2, \ldots, m\}$. Let $U = X \times Y \times Z$. In what follows, we denote an element of $U$ as $(x, y, z)$, where $x \in X, y \in Y$, and $z \in Z$. Let $U_X, U_Y$, and $U_Z$ be three copies of $U$, and $X', Y'$, and $Z'$ be copies of $X, Y$, and $Z$, respectively. Define the set of vertices as follows:

$$V \quad = \quad X \cup Y \cup Z \cup X' \cup Y' \cup Z' \cup U_X \cup U_Y \cup U_Z.$$

We connect vertices in $V$ as follows:

- {Connections among vertices in $X \cup Y \cup Z \cup X' \cup Y' \cup Z'$.}
  Any two vertices in $X \cup X'$ (respectively, $Y \cup Y', Z \cup Z'$) are connected by an edge.
- {Connections among vertices in $U_X \cup U_Y \cup U_Z$.}
  - Any $(i, j, k)$ and $(i', j', k)$ in $U_Z$ are "not" connected by an edge if and only if both $\{i, j, k\}$ and $\{i', j', k\}$ are in $M$, and $i' = i$ or $j' = j$.
  - Any $(i, j, k)$ and $(i, j', k')$ in $U_X$ are "not" connected by an edge if and only if both $\{i, j, k\}$ and $\{i, j', k'\}$ are in $M$, and $j' = j$ or $k' = k$.
  - Any $(i, j, k)$ and $(i', j, k')$ in $U_Y$ are "not" connected by an edge if and only if both $\{i, j, k\}$ and $\{i', j, k'\}$ are in $M$, and $k' = k$ or $i' = i$.
  - Any $(i, j, k) \in U_Z$ and $(i, j', k) \in U_X$ are connected by an edge if and only if $j' \neq j$ and both $\{i, j, k\}$ and $\{i, j', k\}$ are in $M$.
  - Any $(i, j, k) \in U_X$ and $(i, j, k') \in U_Y$ are connected by an edge if and only if $k' \neq k$ and both $\{i, j, k\}$ and $\{i, j, k'\}$ are in $M$.
  - Any $(i, j, k) \in U_Y$ and $(i', j, k) \in U_Z$ are connected by an edge if and only if $i' \neq i$ and both $\{i, j, k\}$ and $\{i', j, k\}$ are in $M$.

  See Figure B.1 for illustration.
- {Other connections.} For any $\{i, j, k\} \in M$, let $\hat{X}_{*,j,k}$ denote the set of all vertices $i' \in X$ such that $\{i', j, k\} \in M$, and define $\hat{Y}_{i,*,k}, \hat{Z}_{i,j,*}, \hat{X}'_{*,j,k}, \hat{Y}'_{i,*,k}$, and $\hat{Z}'_{i,j,*}$, in a similar way.
  - Let $(i, j, k)$ be a vertex in $U_Z$. If $\{i, j, k\} \in M$, then connect it with all vertices in
  $$\hat{X}_{*,j,k} \cup (X' \setminus \hat{X}'_{*,j,k}) \cup \hat{Y}'_{i,*,k} \cup (Y \setminus \hat{Y}_{i,*,k}).$$
  If $\{i, j, k\} \notin M$, then connect it with all vertices in $Y \cup Z'$.
  - Let $(i, j, k)$ be a vertex in $U_X$. If $\{i, j, k\} \in M$, then connect it with all vertices in
  $$\hat{Y}_{i,*,k} \cup (Y' \setminus \hat{Y}'_{i,*,k}) \cup \hat{Z}'_{i,j,*} \cup (Z \setminus \hat{Z}_{i,j,*}).$$
  If $\{i, j, k\} \notin M$, then connect it with all vertices in $Z \cup X'$.
  - Let $(i, j, k)$ be a vertex in $U_Y$. If $\{i, j, k\} \in M$, then connect it with all vertices in
  $$\hat{Z}_{i,j,*} \cup (Z' \setminus \hat{Z}'_{i,j,*}) \cup \hat{X}'_{*,j,k} \cup (X \setminus \hat{X}_{*,j,k}).$$
  If $\{i, j, k\} \notin M$, then connect it with all vertices in $X \cup Y'$.

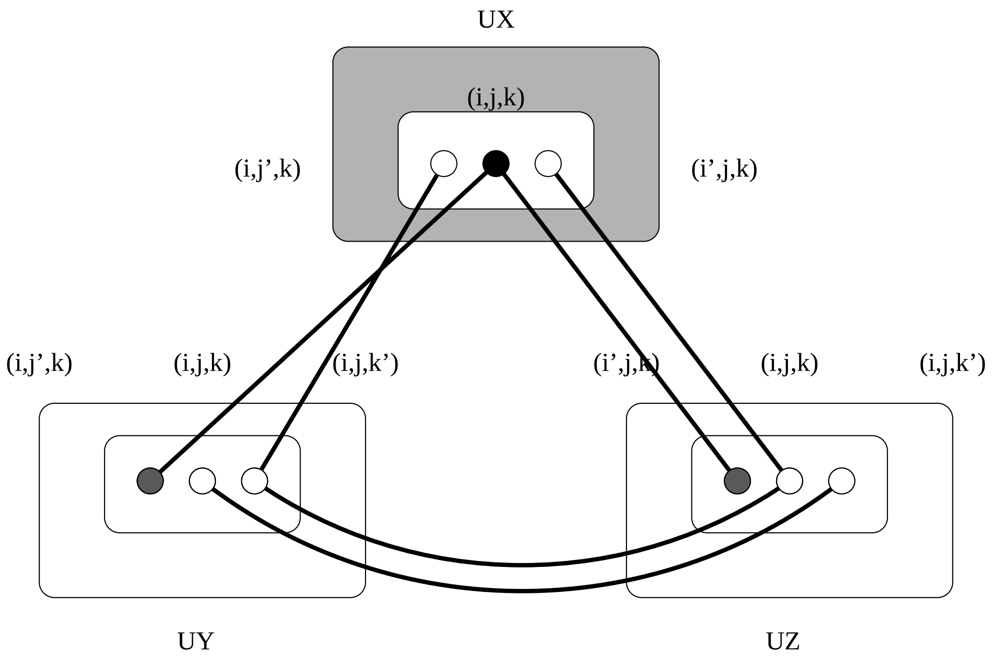


FIG. B.1. *The connections among vertices in $U_Z \cup U_X \cup U_Y$.*

See Figures B.2 and B.3 for illustration.

First, we claim that graph $G$ is regular.

LEMMA B.1. *Graph $G$ is a $(|V|/3-1)$-regular graph with $|V|$ vertices.*

*Proof.* Any vertex $i$ in $X$ connects with $m-1$ vertices in $X$ and $m$ vertices in $X'$. Further, it connects with $m^3-K$ vertices in $U_Y$ and $K$ vertices in $U_Z$, where $K$ is the number of vertices $(i',j',k') \in U_Y$ such that $i \in \hat{X}_{*,j',k'}$. Hence, each vertex in $X$ is connected with $m^3+2m-1 = |V|/3-1$ vertices. A similar argument holds for the vertices in $Y \cup Z \cup X' \cup Y' \cup Z'$.

Let $(i,j,k)$ be a vertex in $U_Z$. If $\{i,j,k\} \notin M$, then it connects with $m^3-1$ vertices in $U_Z$ and all (i.e., $2m$) vertices in $Y \cup Z'$. Hence, in total, it connects with $m^3+2m-1 = |V|/3-1$ vertices. If $\{i,j,k\} \in M$, then it connects with $m^3 - |\hat{X}_{*,j,k}| - |\hat{Y}_{i,*,k}| + 1$ vertices in $U_Z$, $|\hat{Y}_{i,*,k}|-1$ vertices in $U_X$, and $|\hat{X}_{*,j,k}|-1$ vertices in $U_Y$; i.e., it connects with $m^3-1$ vertices in $U_X \cup U_Y \cup U_Z$. In addition, it connects with $|\hat{X}_{*,j,k}|$ vertices in $X$, $m - |\hat{X}'_{*,j,k}| = m - |\hat{X}_{*,j,k}|$ vertices in $X'$, $|\hat{Y}'_{i,*,k}| = |\hat{Y}_{i,*,k}|$ vertices in $Y'$, and $m - |\hat{Y}_{i,*,k}|$ vertices in $Y$. Hence, in total, it connects with $(m^3-1)+2m = |V|/3-1$ vertices. A similar argument holds for the vertices in $U_X \cup U_Y$. ∎

LEMMA B.2. *For any $\{i,j,k\} \notin M$, the three copies of $(i,j,k)$ in $U_X, U_Y$, and $U_Z$ form a dominating set for graph $G$.*

*Proof.* The copy of $(i,j,k)$ in $U_Y$ dominates $U_Y \cup X \cup Y'$, the copy of $(i,j,k)$ in $U_Z$ dominates $U_Z \cup Y \cup Z'$, and the copy of $(i,j,k)$ in $U_X$ dominates $U_X \cup Z \cup X'$. Hence the lemma follows. ∎

LEMMA B.3. *For any $\{i,j,k\} \in M$, the three copies of $(i,j,k)$ in $U_Z, U_X$, and $U_Y$ form a dominating set for graph $G$.*

*Proof.* Recall that for any $\{a,b,c\}, \{a,b,d\} \in M$, if $\{x,y,c\} \in M$ for some $x, y$,

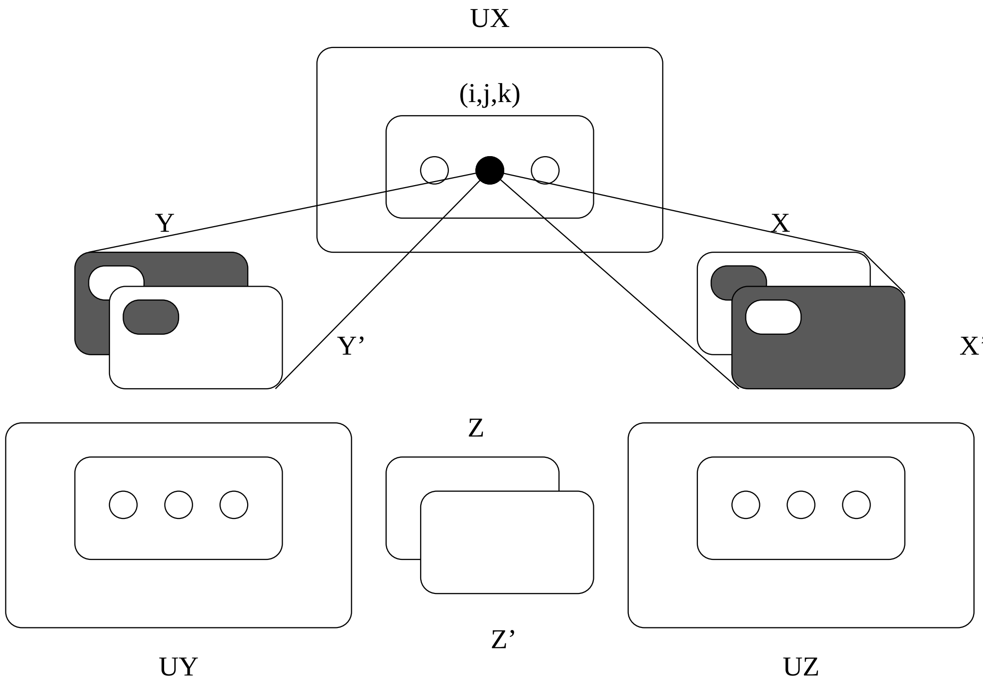


FIG. B.2. *Other connections in the case where $\{i, j, k\} \in M$.*

then $\{x, y, d\} \in M$. Thus, for any $i'' \in \hat{X}_{*,j,k}$, $j'' \in \hat{Y}_{i,*,k}$, and $k'' \in \hat{Z}_{i,j,*}$, the 3-set $\{i'', j'', k''\}$ is an element of $M$. Hence, the copy of $(i, j, k)$ in $U_Z$ dominates

$$\{\hat{X}_{*,j,k} \cup (Y \setminus \hat{Y}_{i,*,k})\} \cup \{\hat{Y}'_{i,*,k} \cup (X' \setminus \hat{X}'_{*,j,k})\},$$

the copy of $(i, j, k)$ in $U_X$ dominates

$$\{\hat{Y}_{i,*,k} \cup (Z \setminus \hat{Z}_{i,j,*})\} \cup \{\hat{Z}'_{i,j,*} \cup (Y' \setminus \hat{Y}'_{i,*,k})\},$$

and the copy of $(i, j, k)$ in $U_Y$ dominates

$$\{\hat{Z}_{i,j,*} \cup (X \setminus \hat{X}_{*,j,k})\} \cup \{\hat{X}'_{*,j,k} \cup (Z' \setminus \hat{Z}'_{i,j,*})\};$$

i.e., the 3-set consisting of the three copies of $(i, j, k)$ is a dominating set for $X \cup Y \cup Z \cup X' \cup Y' \cup Z'$. The 3-set is also a dominating set for $U_X \cup U_Y \cup U_Z$, since any vertex $(i', j, k)$ (respectively, $(i, j', k)$) in $U_Z$ which is not dominated by the copy of $(i, j, k)$ in $U_Z$ is dominated by the copy of $(i, j, k)$ in $U_Y$ (respectively, $U_X$), and a similar argument holds for vertices in $U_X \cup U_Y$. ▯

By the above two lemmas, $U_X \cup U_Y \cup U_Z$ can be partitioned into $m^3$ $(= |U_X|)$ dominating 3-sets, regardless of the existence of an exact cover $M' \subset M$ of $M$.

LEMMA B.4. *Let $\{i, j, k\}$ be a subset of $X \cup Y \cup Z \cup X' \cup Y' \cup Z'$. If $M \subset U$ and $\{i, j, k\}$ is a dominating set for $G$, then $\{i, j, k\} \subset X \cup Y \cup Z$ or $\{i, j, k\} \subset X' \cup Y' \cup Z'$.*

*Proof.* Let $\{i, j, k\}$ be any dominating set for $G$. By Lemma B.1, any dominating 3-set for $G$ is perfect. We therefore can assume, without loss of generality, that $i \in X \cup X'$, $j \in Y \cup Y'$, and $k \in Z \cup Z'$.

Suppose that $\{i, j, k\} \not\subseteq X \cup Y \cup Z$ and $\{i, j, k\} \not\subseteq X' \cup Y' \cup Z'$. If $i \in X$, then $j$ must be in $Y$, since any element $(i, j, k) \in U_Y$ such that $\{i, j, k\} \notin M$ is connected

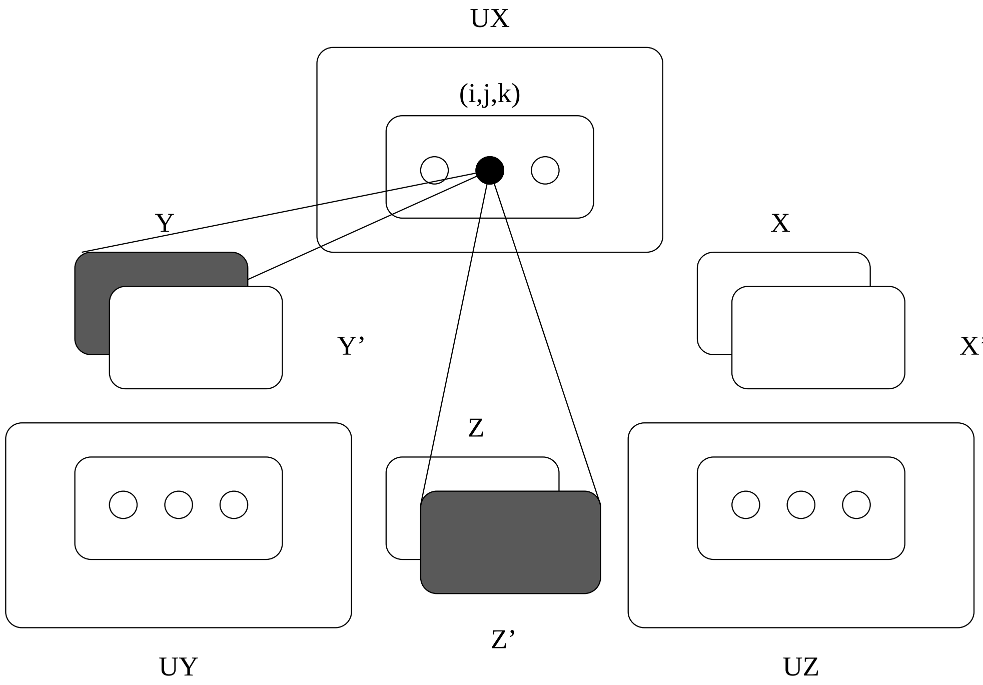


FIG. B.3. *Other connections in the case where $\{i,j,k\} \notin M$.*

with all vertices in $X \cup Y'$. If $j \in Y$, then since any element $(i,j,k) \in U_Z$ such that $\{i,j,k\} \notin M$ is connected with all vertices in $Y \cup Z'$, $z$ must be in $Z$, which is a contradiction. Similarly, if $i \in X'$, then $k$ must be in $Z'$, and if $k \in Z'$, then $j$ must be in $Y'$, again, which is a contradiction. $\square$

LEMMA B.5. *A 3-set $\{i,j,k\} \subset X \cup Y \cup Z$ is a dominating set for $G$ if and only if $\{i,j,k\} \in M$.*

*Proof.* If $M = U$ $(= X \times Y \times Z)$, then the lemma is obviously correct. So we assume $M \subset U$.

Suppose that $\{i,j,k\} \in M$. Clearly, the 3-set $\{i,j,k\} \subset X \cup Y \cup Z$ dominates $X \cup Y \cup Z$. Let $\hat{U} = \{(x,y,z) \in U \mid x \in \hat{X}_{*,j,k}, y \in \hat{Y}_{i,*,k}, z \in \hat{Z}_{i,j,*}\}$. By definition, the copy of $\hat{U}$ in $U_Y$ is dominated by vertex $k \in Z$ and the copy of $U \setminus \hat{U}$ in $U_Y$ is dominated by vertex $i \in X$. Similarly, the copy of $\hat{U}$ in $U_Z$ (respectively, $U_X$) is dominated by vertex $i \in X$ (respectively, $j \in Y$) and the copy of $U \setminus \hat{U}$ in $U_Y$ (respectively, $U_X$) is dominated by vertex $j \in Y$ (respectively, $k \in Z$). Hence, $\{i,j,k\} \subset X \cup Y \cup Z$ is a dominating set for $G$.

Suppose that there is a dominating set $\{i,j,k\}$ for $G$ such that $\{i,j,k\} \notin M$. By Lemma B.4, without loss of generality, we may assume that $\{i,j,k\} \subset X \cup Y \cup Z$.

Let $\{i,j',k'\}$ be an element of $M$ containing $i$. If $\{i,j',k\}$ is not an element of $M$, then vertex $(i,j',k') \in U_Y$ is not dominated by any vertex in $\{i,j,k\}$. Hence $\{i,j',k\}$ must be an element of $M$. If $\{i,j,k'\}$ is not an element of $M$, then vertex $(i,j',k') \in U_X$ is not dominated by any vertex in $\{i,j,k\}$. Hence $\{i,j,k'\}$ must be an element of $M$. Since $\{i,j',k'\}, \{i,j',k\}$, and $\{i,j,k'\}$ are elements in $M$, by assumption, $\{i,j,k\}$ must be an element of $M$, which is a contradiction. $\square$

By the above lemmas, we can conclude that $D_1(G) = m^3 + 2m$ if and only if there is an exact cover $M' \subset M$. Hence the lemma follows.

## REFERENCES

[1] M. J. Atallah, G. K. Manacher, and J. Urrutia, *Finding a minimum independent dominating set in a permutation graph*, Discrete Appl. Math., 21 (1988), pp. 177–183.

[2] D. W. Bange, A. E. Barkauskas, and P. J. Slater, *Efficient dominating sets in graphs*, in Applications of Discrete Mathematics, Proceedings of the Third Conference on Discrete Mathematics, Clemson, SC, May 14–16, 1986, R. D. Ringeisen and F. S. Roberts, eds., SIAM, Philadelphia, PA, 1988, pp. 189–199.

[3] A. A. Bertossi, *On the domatic number of interval graphs*, Inform. Process. Lett., 28 (1988), pp. 275–280.

[4] N. Biggs, *Perfect codes in graphs*, J. Combin. Theory Ser. B, 15 (1973), pp. 289–296.

[5] G. J. Chang, C. P. Rangan, and S. R. Coorg, *Weighted Independent Perfect Domination on Cocomparability Graphs*, Tech. report 93-24, DIMACS, Rutgers University, Piscataway, NJ, 1993.

[6] E. J. Cockayne and S. T. Hedetniemi, *Optimal domination in graphs*, IEEE Trans. Circuits Systems, CAS-22 (1975), pp. 855–857.

[7] M. Farber, *Domination, independent domination, and duality in strongly chordal graphs*, Discrete Appl. Math., 7 (1984), pp. 115–130.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, 1979.

[9] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Fundamentals of Domination in Graphs*, Marcel Dekker, New York, 1998.

[10] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Domination in Graphs: Advanced Topics*, Marcel Dekker, New York, 1998.

[11] I. Holyer, *The NP-completeness of some edge-partition problems*, SIAM J. Comput., 10 (1981), pp. 713–717.

[12] R. W. Irving, *On approximating the minimum independent dominating set*, Inform. Process. Lett., 37 (1991), pp. 197–200.

[13] M. Livingston and Q. F. Stout, *Perfect dominating sets*, Congress. Numer., 79 (1990), pp. 187–203.

[14] T.-L. Lu, P.-H. Ho, and G. J. Chang, *The domatic number problem in interval graphs*, SIAM J. Discrete Math., 3 (1990), pp. 531–536.

[15] L. R. Matheson and R. E. Tarjan, *Dominating Sets in Planar Graphs*, Tech. report TR-461-94, Dept. of Computer Science, Princeton University, Princeton, NJ, 1994.

[16] J. Petersen, *Die Theorie der regulären Graphen*, Acta Math., 15 (1891), pp. 193–220.

[17] A. S. Rao and C. P. Rangan, *Linear algorithm for domatic number problem on interval graphs*, Inform. Process. Lett., 33 (1989), pp. 29–33.

[18] C. C. Yen and R.C.T. Lee, *The weighted perfect domination problem*, Inform. Process. Lett., 35 (1990), pp. 295–299.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 2, pp. 255–267

# APPROXIMATING MINIMUM SUBSET FEEDBACK SETS IN UNDIRECTED GRAPHS WITH APPLICATIONS[*]

GUY EVEN[†], JOSEPH (SEFFI) NAOR[‡], BARUCH SCHIEBER[§], AND LEONID ZOSIN[¶]

**Abstract.** Let $G = (V, E)$ be a weighted undirected graph where all weights are at least one. We consider the following generalization of feedback set problems. Let $S \subset V$ be a subset of the vertices. A cycle is called *interesting* if it intersects the set $S$. A subset feedback edge (vertex) set is a subset of the edges (vertices) that intersects all interesting cycles. In minimum subset feedback problems the goal is to find such sets of minimum weight. This problem has a variety of applications, among them genetic linkage analysis and circuit testing. The case in which $S$ consists of a single vertex is equivalent to the *multiway cut* problem, in which the goal is to separate a given set of terminals. Hence, the subset feedback problem is NP-complete and also generalizes the multiway cut problem. We provide a polynomial time algorithm for approximating the subset feedback edge set problem that achieves an approximation factor of two. This implies a $\Delta$-approximation algorithm for the subset feedback vertex set problem, where $\Delta$ is the maximum degree in $G$. We also consider the multicut problem and show how to achieve an $O(\log \tau^*)$ approximation factor for this problem, where $\tau^*$ is the value of the optimal fractional solution. To achieve the $O(\log \tau^*)$ factor we employ a bootstrapping technique.

**Key words.** approximation algorithms, feedback vertex set, feedback edge set, subset feedback set, combinatorial optimization, multicut

**AMS subject classifications.** 68Q25, 68R10, 05C85, 68W25

**PII.** S0895480195291874

## 1. Introduction.

### 1.1. Problem definition.

Let $G = (V, E)$ be an undirected graph, and suppose that there is a weight function $w$ associated with either the vertices or edges of $G$. A *feedback vertex set* (FVS) is defined to be a set of vertices that intersects all cycles in the graph. A *feedback edge set* (FES) is similarly defined. The weight of a feedback set is defined to be equal to the sum of the weights of the elements in the set.

We define a natural generalization of the FVS and FES problems by restricting the set of cycles that the feedback set should intersect. These cycles are called *interesting* cycles. Let $S \subseteq V$ be a set of *special* vertices. In this paper, a cycle is considered to be interesting if it contains a special vertex. A feedback set in this case is called a SUBSET-FVS or SUBSET-FES, depending on the version to be considered. We are interested in computing a minimum weight subset feedback set.

[*]Received by the editors September 13, 1995; accepted for publication (in revised form) November 22, 1999; published electronically April 6, 2000. A preliminary version of this paper appeared in the *Proceedings of the Fourth Israeli Symposium on Theory of Computing and Systems (ISTCS)*, Jerusalem, Israel, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 78–88.
http://www.siam.org/journals/sidma/13-2/29187.html

[†]Department of Electrical Engineering – Systems, Tel-Aviv University, Ramat-Aviv, Tel-Aviv 69978, Israel (guy@eng.tau.ac.il). Part of this work was done while the author was supported by Graduiertenkolleg "Effizienz und Komplexität von Algorithmen und Rechenanlagen," Universität des Saarlandes, Saarbrücken, Germany.

[‡]Computer Science Department, Technion, Haifa 32000, Israel (naor@cs.technion.ac.il). The research of this author was supported in part by grant 92-00225 from the United States–Israel Binational Science Foundation (BSF), Jerusalem, Israel and by Technion V.P.R. Fund 120-905–promotion of sponsored research.

[§]IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (sbar@watson.ibm.com).

[¶]Computer Science Department, Technion, Haifa 32000, Israel (leon@cs.technion.ac.il). Current address: NEC Research Institute, 4 Independence Way, Princeton, NJ 08540.

**1.2. Motivation.** The problem of finding feedback sets in undirected graphs arises in a variety of applications in genetics, circuit testing, artificial intelligence, deadlock resolution, and analysis of manufacturing processes.

The motivation for our generalization to SUBSET-FES and SUBSET-FVS problems is twofold. First, in some of the applications we may be interested only in cycles that intersect a subset of the vertices and edges. Second, from the theoretical standpoint, the subset feedback problem defines a full spectrum of feedback problems which are categorized according to the cardinality of the set $S$. The two extreme points of this spectrum are the feedback set problem, i.e., where $S = V$, and the case where $|S| = 1$ which turns out to be equivalent to the multiway cut problem, as demonstrated in section 1.3. This paper provides a 2-approximation for the SUBSET-FES problem, and thus, we show that the approximation factor does not depend on the number of special vertices.

We now describe the applicability of the subset feedback set problem to genetic linkage.

*Genetic linkage analysis.* Genetic linkage is the phenomenon where alleles of different genes appear to be genetically coupled. The goal of genetic linkage is to quantify this linkage by estimating the recombination fraction which denotes the probability of a crossover occurring in a region of a specific chromosome of interest. The essence of linkage analysis, which is routinely performed by geneticists, is to use family pedigrees in order to estimate parameters that determine whether a set of genes resides on the same chromosome, what is the relative order of them on the chromosome, and what is the approximate distance between them. This is important for, e.g., genetic counseling and estimating changes in population genetics. An excellent treatment of this subject can be found in [20].

Analysis of genetic linkage is an area where *Bayesian networks* can be naturally applied [20]. Bayesian networks are useful knowledge representation schemes which allow a wide spectrum of independence assumptions to be considered by a model builder so that a practical balance can be established between computational needs and adequacy of conclusions. For a complete exploration of this subject see [21].

Suppose that a pedigree (of one or more persons) is given. A pedigree can be represented by a Bayesian network where each vertex is a random variable representing the genes under study. The problem of estimating the recombination fraction translates to an updating problem in the Bayesian network representation of the pedigree [21] which is essentially the computation of certain conditional probabilities. In general, such computations are inefficient both in time and space, and therefore one must resort to heuristics for the updating problem.

The connection between algorithms for the updating problem in Bayesian networks having specific topologies (e.g., when the number of cycles is small) and feedback sets in undirected graphs was established by Bar-Yehuda et al. [1]. They show that the running time of the conditioning algorithm [21] for the updating problem depends exponentially on the weight of the feedback set computed in an undirected graph that is derived from the Bayesian network. (See also Becker, Geiger, and Schäffer [4] who discuss the usage of feedback vertex sets for efficient genetic linkage analysis.)

When considering Bayesian networks that are derived from pedigrees, the influence of distant ancestors on the genotype of a newborn is negligible. This means that Pearl's conditioning algorithm [21] can deal with cycles induced by vertices corresponding to distant ancestors by arbitrarily assigning them a genotype, until all such cycles are eliminated. Thus, the cost of eliminating such cycles is very cheap since there is no longer a need to perform an exhaustive search. We note that the arbitrariness of the selected genotype has little effect on the likelihood computations. Thus, the complexity of estimating the recombination fraction is dominated by the weight of the subset feedback set of the cycles that contain a vertex corresponding to one of the recent generations. Such cycles are defined to be the interesting cycles in the undirected graph derived from the Bayesian network. An example of a genetic linkage analysis where only a subset of the cycles could be covered (but not a subset feedback vertex set) appears in [5].

**1.3. Hardness.** Computing a minimum weight FVS is a classical NP-complete problem, as shown by Karp in his seminal paper [17]. (See also [12].) This motivated the search for polynomial time approximation algorithms. Bar-Yehuda et al. [1] showed how to approximate the FVS problem by a factor of four in the unweighted case; this bound was later extended to the weighted case and improved to two in [3, 2]; see also [7, 16]. In contrast, the FES problem can easily be solved in polynomial time by observing that a minimum weight FES is the complement of a maximum weight spanning tree in the graph.

The subset feedback set problem generalizes the *multiway cut* problem, defined as follows. Let $T$ be a set of terminals in a (weighted) graph. A multiway cut is a set of edges (or vertices) whose removal disconnects every pair of terminals in $T$. Finding a minimum weight multiway cut problem is known to be NP-complete [8]. The multiway cut problem is equivalent to the subset feedback set problem with a single special vertex. This equivalence is obtained by connecting all terminals in $T$ to a special vertex $s$. (In the vertex version $s$ has infinite weight, and in the edge version the edges adjacent to $s$ have infinite weight.) Any cycle passing through $s$ corresponds to a path connecting two terminals in $T$, and thus, a subset feedback set with respect to $s$ is also a multiway cut of $T$ (and vice versa). This establishes that the subset feedback set problem, in both edge and vertex versions, is NP-complete, even for the case where $|S| = 1$.

**1.4. Our results.** Having established that the subset feedback set problem is NP-complete, even for the case of a single special vertex, we consider approximation algorithms.

In section 2, we present our main result: For any weight function on the edges, the SUBSET-FES problem can be approximated in polynomial time by a factor of two. The algorithm itself is very simple and consists of successive computations of minimum cuts. On the other hand, the proof that this algorithm indeed achieves a 2-approximation factor requires a delicate analysis. In the analysis, capacities of minimum cuts are represented by flow paths. When new minimum cuts are computed, previous flow paths are updated according to the decomposition of the graph induced by an optimal solution. Finally, a charging scheme relates the flow along the updated flow paths to the capacity of an optimal solution.

In section 2.3, we show that applying the SUBSET-FES algorithm to the SUBSET-FVS problem yields an approximation factor of $\Delta$, where $\Delta$ denotes the maximum degree in the graph. Subsequent to our paper, Even, Naor, and Zosin [11] gave an 8-approximation algorithm for the SUBSET-FVS problem. Thus, the $\Delta$-approximation factor is of interest only for graphs for which $\Delta \leq 8$. However, we note that the algorithm of [11] is computationally expensive, and therefore our result for the SUBSET-FVS problem might be of use to low degree graphs in general.

In section 3, we present a new technique called *bootstrapping* for enhancing the approximation factor of the undirected multicut problem. Given $k$ pairs of sources and sinks in an edge (vertex) weighted graph, a *multicut* is defined to be a set of edges (vertices) that separates all source-sink pairs. The best approximation factor known for multicuts is $O(\log k)$ [13]. Assuming that all weights are at least one, the bootstrapping technique improves the approximation factor for multicuts to $O(\min\{\log k, \log \tau\})$, where $\tau^*$ denotes the value of an optimal fractional multicut. This technique is also applicable to the directed subset feedback set problem.

Feedback set problems fall into the category of *covering problems* and can be cast in a "natural" way as an integer program, where there is an indicator variable for each vertex or edge, and the requirement is that all interesting cycles are covered. In section 4 we provide a simple example that shows that the integrality gap of this integer program, i.e., the ratio between the integral and fractional optima for the SUBSET-FES and SUBSET-FVS problems, can be as high as $\Omega(\log |S|)$. This implies that any technique that is based on rounding an optimal fractional solution for this "natural" formulation of the problem to an integral one is doomed to yield a non-constant approximation factor.

Note that the example presented in section 4 shows that the integrality gap for the FVS and FES problems is $\Omega(\log |V|)$. (Recall that the approximation factor achievable for this problem is 2 [3, 2].) Interestingly, for the FVS problem, Chudak et al. [7] (see also [16]) presented an alternative formulation of an integer program, for which the integrality gap is at most two. The problem of finding an alternative integer program for the SUBSET-FES and SUBSET-FVS problems for which the integrality gap is constant is very challenging.

**1.5. Prior work.** Minimum weight feedback sets in *directed* graphs were considered in several papers [19, 23, 18, 9, 10]. Note that in the directed case, the vertex and edge versions are equivalent. Seymour [23] showed that the integrality gap between the optimal fractional FVS and the integral optimal FVS in directed graphs is $O(\log \tau^* \log\log \tau^*)$ and can be $\Omega(\log \tau^*)$. In [9, 10] algorithms for approximating subset feedback sets in directed graphs that are based on Seymour's techniques were presented, achieving an $O(\min\{\log \tau^* \log\log \tau^*, \log |S| \log\log |S|\})$ approximation factor. Recently, Goemans and Williamson [15] considered a generalization of the SUBSET-FVS problem in undirected planar graphs. In particular, they presented a 9/4-approximation algorithm for the SUBSET-FVS problem in undirected planar graphs and for the FVS problem in directed planar graphs.

The multiway cut problem (in both edge and vertex versions) can be approximated in polynomial time by a factor of $2 - 1/|T|$ [8, 14]. The approximation algorithms for the multiway cut problem do not translate into good approximation algorithms for the subset feedback problems. The only vertex-multiway cut algorithm [14] is based on the half-integrality of an optimal solution to a linear program that formulates the multiway cut problem as a covering problem. However, as discussed above, the integrality gap of this linear programs for SUBSET-FES and SUBSET-FVS is $\Omega(\log |S|)$. Hence, a linear programming "rounding" scheme cannot yield a constant approximation factor.

The edge-multiway cut approximation algorithm presented by Dahlhaus et al. [8] finds a multiway cut by computing a minimum cut from every terminal to the rest of the terminals. The solution consists of all the cuts except for the "heaviest" cut. An analogous algorithm for the SUBSET-FES problem in which an (almost) optimal feedback edge set is found independently for each special vertex yields an $|S| - o(1)$ approximation. For example, consider the "flower" graph depicted in Figure 1. Suppose that the weight of each $(v_i, v_{i+1})$ edge is $\epsilon$ and that the weight of each $(s_i, v_j)$ edge is one. A minimum weight feedback edge set with respect to $s_i$ contains a single

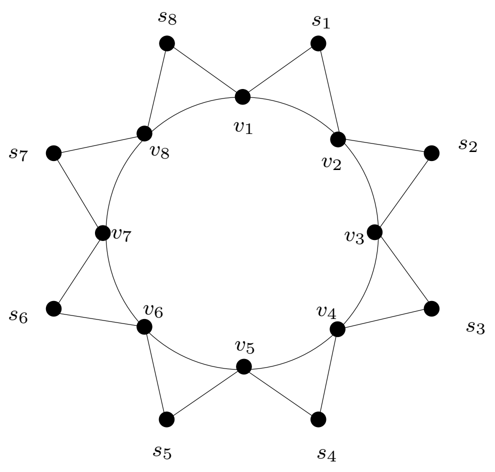


FIG. 1. *A flower graph.*

edge, either $(v_i, s_i)$ or $(s_i, v_{i+1})$, and hence, the total weight of such feedback edge sets equals eight (note that this is a minimal solution). On the other hand, a minimum weight subset feedback edge set contains only one edge of weight one and seven edges of weight $\epsilon$.

**2. Approximating the subset feedback set problem.** In this section we present an approximation algorithm for the SUBSET-FES problem. In section 2.3 we apply this algorithm to the SUBSET-FVS problem and obtain improved results for some special cases.

Let $G = (V, E)$ be an undirected graph with a weight function $w : E \to \mathbb{R}$. For a set of edges $E'$, let $w(E')$ denote the total weight of the edges in $E'$. Let the set of special vertices be $S = \{s_1, \ldots, s_k\}$. Define $V_i \triangleq V - \{s_i, \ldots, s_k\}$. (Note that $V_1$ is the set of nonspecial vertices, $V_i \subset V_{i+1}$, and that $V_{k+1} = V$.)

We start with some simplifying assumptions. Without loss of generality, assume that $G$ is connected and that for each special vertex, the following holds: (i) It has degree two; (ii) its two neighbors are not special vertices; (iii) its two adjacent edges have infinite weight. We justify these assumptions by the following reduction applied whenever a special vertex $s$ does not satisfy these requirements. For each edge $e$ adjacent to $s$, split $e$ into three new edges $e_1$, $e_2$, and $e_3$ and insert two new vertices of degree two each: vertex $s'$ between $e_1$ and $e_2$ and vertex $s''$ between $e_2$ and $e_3$. Let $e_1$ denote the edge that is adjacent to $s$. Edges $e_1$ and $e_2$ are assigned infinite weight whereas edge $e_3$ is assigned the same weight as $e$. Add the new vertex $s'$ to the set of special vertices $S$ and delete $s$ from $S$. It is not difficult to see that when this reduction terminates, the assumptions hold. Clearly, there is a one-to-one weight preserving correspondence between feedback sets before and after the reduction, and hence, approximating the optimal solution before the reduction is equivalent to approximating the optimal solution after reduction.

**2.1. The algorithm.** The algorithm consists of $k$ iterations. In the $i$th iteration, the algorithm finds a feedback set that covers all the interesting cycles that were not covered in previous iterations and in which the highest index of a special vertex is $i$. Let the two neighbors of special vertex $s_i$ be denoted by $x_i$ and $y_i$. In the first iteration, the algorithm finds a minimum cut, $M_1$, between vertices $x_1$ and $y_1$ in the graph induced by $V_1$. In the $i$th iteration, $(1 < i \leq k)$, it finds a minimum cut, $M_i$, between $x_i$ and $y_i$ in the graph $G_i$ induced by $V_i$, and the edge set $E - M_1 \cup \cdots \cup M_{i-1}$. The algorithm returns as a SUBSET-FES the edges of $M_1 \cup \cdots \cup M_k$. A formal description of the algorithm follows.

ALGORITHM SUBSET-FES.

```
*Input:* Graph $G = (V, E)$ with special vertices $s_1, \ldots, s_k$;
*Output:* SUBSET-FES $M$ of $G$;

*Notation:* $V_i \triangleq V - \{s_i, \ldots, s_k\}$;
for $i = 1$ to $k$ do
    $M_i \leftarrow$ minimum cut between $x_i$ and $y_i$ in the graph $G_i = (V_i, E - \cup_{j=1}^{i-1} M_j)$.
$M \leftarrow M_1 \cup \cdots \cup M_k$.
```

It is easy to see that the algorithm finds a feasible solution to the SUBSET-FES problem. We now show that the weight of its solution approximates the optimal solution by a factor of two.

**2.2. The analysis.** Let OPT denote a minimum weight solution to the SUBSET-FES problem. Define the graph $H = (V, E - \text{OPT})$ to be the graph induced by the vertex set $V$ and the edge set $E - \text{OPT}$. Clearly, $H$ does not contain any interesting cycles. We can assume without loss of generality that $G$ is connected, and therefore $H$ is also connected. For $i = 1, \ldots, k+1$, let $H_i$ be the subgraph of $H$ induced by $V_i$. In particular, $H_1$ is the subgraph of $H$ induced by the set of nonspecial vertices. Note that $H_1$ consists of several connected components, since for each special vertex $s_i$, its neighbors $x_i$ and $y_i$ cannot belong to the same connected component of $H_1$, as this would imply the existence of an interesting cycle in $H$.

First, we claim that the number of components in $H_1$ is $k+1$. To see this, consider the auxiliary graph obtained from $H$ by contracting each connected component of $H_1$ into a vertex. Namely, the vertex set of the auxiliary graph is the set of special vertices together with one vertex for each connected component of $H_1$. The degree of each special vertex $s_i$ in the auxiliary graph is two: one edge connects $s_i$ to the vertex corresponding to the component containing $x_i$, and the other edge connects $s_i$ to the vertex corresponding to the component containing $y_i$. Since $H$ does not contain any interesting cycles, the auxiliary graph must be acyclic. Since $H$ is connected, the auxiliary graph is a tree. The number of edges in the auxiliary graph is precisely $2k$, yielding that the number of vertices must be $2k+1$. Since the auxiliary graph contains $k$ special vertices, the number of connected components in $H_1$ is $k+1$.

Denote the connected components of $H_1$ by $C_1, \ldots, C_{k+1}$. Let $\text{OPT}_i$ denote the edges in OPT with one endpoint in $C_i$. Since each edge in OPT touches two connected components we get that $\sum_{i=1}^{k+1} w(\text{OPT}_i) = 2w(\text{OPT})$. To show that the approximation factor of our algorithm is two, we present an injective mapping from the set of $k$ minimum cuts computed by the algorithm, $M_1, \ldots, M_k$, to the set of $k+1$ components of $H_1$. This mapping has the property that if minimum cut $M_i$ is mapped to component $C_j$, then $w(M_i) \leq w(\text{OPT}_j)$. Since the mapping is injective (i.e., each component may be the image of at most one cut), we have $\sum_{i=1}^{k} w(M_i) \leq 2w(\text{OPT})$.

We present the injective mapping in two stages. First, for each minimum cut $M_i$ we identify two components $C_{\ell(i)}$ and $C_{r(i)}$ such that

$$w(M_i) \leq \min\{w(\text{OPT}_{\ell(i)}), w(\text{OPT}_{r(i)})\}.$$

Then, we show how to map each $M_i$ to one of these two components so that the mapping is injective.

*Identifying the two components.* Consider a simple path $p$ from $x_i$ to $y_i$ in $G_i$. Discard from $p$ all nonspecial vertices except for $x_i$ and $y_i$. The remaining vertices, in the order of their occurrence in $p$, are called the *backbone* of $p$.

Follow the backbone of $p$, starting from $x_i$ until the first vertex that is not connected in $H_i$ to its subsequent vertex in the backbone. Such a vertex always exists since the path $p$ is disconnected by OPT. If this vertex is $x_i$, then we set $C_{\ell(i)}$ to be the component of $H_1$ containing $x_i$. Otherwise, this vertex is a special vertex $s_j$ for some $j < i$. In this case $C_{\ell(i)}$ is set to be the component of $H_1$ containing the vertex following $s_j$ in the path $p$.

The component $C_{r(i)}$ is defined similarly by considering $p$ in reverse order (from $y_i$ to $x_i$).

First, we prove that the backbones of all simple paths from $x_i$ to $y_i$ in $G_i$ are identical. This implies that the selection of $C_{\ell(i)}$ and $C_{r(i)}$ is independent of the choice of the path $p$. Then, we show that the weight of the minimum cut $M_i$ is indeed bounded by both $w(\mathrm{OPT}_{\ell(i)})$ and $w(\mathrm{OPT}_{r(i)})$.

CLAIM 2.1. *All simple paths connecting $x_i$ with $y_i$ in $G_i$ have the same backbone.*

*Proof.* The claim is based on the fact that for every simple path $p$ from $x_i$ to $y_i$ the sequence of special vertices along $p$ is identical. Moreover, if a special vertex $s_j$ is on such a path $p$, then every such path must traverse $s_j$ in the same direction (i.e., either from $x_j$ to $s_j$ and then to $y_j$ or from $y_j$ to $s_j$ and then to $x_j$).

Consider two such paths $p$ and $q$ and suppose that they have different sequences of special vertices. In this case, the union of $p$ and $q$ must contain a nontrivial interesting cycle. Let $j < i$ be the maximum index of a special vertex in this cycle. Note that the path connecting $x_j$ and $y_j$, obtained by removing $s_j$ from the cycle, is in $G_j$. However, this implies that $M_j$ must intersect this path, and thus it cannot belong to $G_i$, yielding a contradiction. A similar argument shows that the special vertices must be traversed in the same direction as well. ∎

CLAIM 2.2. *For $i = 1, \dots k$, $w(M_i) \le \min\{w(\mathrm{OPT}_{\ell(i)}), w(\mathrm{OPT}_{r(i)})\}$.*

*Proof.* We prove that $w(M_i) \le w(\mathrm{OPT}_{\ell(i)})$. The proof that $w(M_i) \le w(\mathrm{OPT}_{r(i)})$ is analogous. By Claim 2.1, all the paths from $x_i$ to $y_i$ in $G_i$ traverse a vertex in $C_{\ell(i)}$. Moreover, by the definition of $C_{\ell(i)}$ all these paths emanate from $C_{\ell(i)}$ using an edge from $\mathrm{OPT}_{\ell(i)}$. Hence, $\mathrm{OPT}_{\ell(i)}$ cuts all these paths. Since $M_i$ is such a cut of minimum weight, the inequality follows. ∎

*The mapping.* We show how to map each $M_i$ to either $C_{\ell(i)}$ or $C_{r(i)}$, guaranteeing that the mapping is injective. For this, consider yet another auxiliary graph. The vertex set of this auxiliary graph consists of the set of special vertices and one vertex for each connected component of $H_1$. For each special vertex $s_i$, the graph has two edges, one connecting $s_i$ to the vertex corresponding to $C_{\ell(i)}$ and the other to the vertex corresponding to $C_{r(i)}$. Note that each edge in the auxiliary graph corresponds to a path in $H$. Hence, every path in the auxiliary graph corresponds to a path in $H$. Moreover, a cycle in the auxiliary graph would correspond to an interesting cycle in $H$. Since $H$ does not contain any interesting cycle, the auxiliary graph is acyclic. The auxiliary graph has $2k$ edges and $2k + 1$ vertices and therefore must be a tree. Now, root this tree at an arbitrary vertex that corresponds to a connected component. In this rooted tree, each special vertex $s_i$ has exactly one child which corresponds to either $C_{\ell(i)}$ or $C_{r(i)}$. The cut $M_i$ is mapped to this component. Clearly, this mapping is injective.

Our discussion proves that the algorithm finds a feasible solution to the SUBSET-FES problem whose weight is at most twice the weight of an optimal solution and thus proves the following theorem.

THEOREM 2.3. *Algorithm* SUBSET-FES *computes a feasible solution whose weight is at most twice the weight of an optimal solution.*

We can easily construct an example that shows that Theorem 2.3 is tight. Consider a clique on $n$ vertices where each edge has unit weight. Connect a new special vertex $s$ to all the vertices of the clique, where each edge adjacent to $s$ has weight $n - \varepsilon$, for some fixed $\varepsilon > 1$. The optimal solution contains all the clique edges, and hence its weight is $n \cdot (n-1)/2$. Algorithm SUBSET-FES chooses all the edges adjacent to $s$, except for one edge. Hence, the approximation factor in this graph is $2 - 2\varepsilon/n$.

**2.3. The SUBSET-FVS problem.** In this section we note that the 2-approximation algorithm for the SUBSET-FES problem yields a $\Delta$-approximation algorithm for the SUBSET-FVS problem, where $\Delta$ denotes the maximum vertex degree in the given graph.

We first note that, similarly to the edge case, we can assume without loss of generality that for each special vertex the following hold: (i) it has infinite weight; (ii) it has precisely two neighbors which are not special vertices; (iii) its two neighbors have infinite weight.

The algorithm for the SUBSET-FVS problem is the same as the SUBSET-FES algorithm except that at each step $i$, $1 \leq i \leq k$, $M_i$ is a minimum weight vertex cut between $x_i$ and $y_i$. The subset feedback vertex set produced by the algorithm is the union of the minimum cuts $\{M_i\}_{i=1}^{k}$. The graph $G_i$ is defined to be the graph induced by $V_i - \cup_{j=1}^{i-1} M_j$.

Let OPT denote a minimum weight solution to the SUBSET-FVS problem. Similar to the edge case define the graph $H = (V - \text{OPT}, E)$ to be the graph induced by the vertex set $V - \text{OPT}$. Clearly, $H$ does not contain any interesting cycles. Let $H_1$ be the subgraph of $H$ induced by $V_1$. Again, $H_1$ consists of several connected components, and in this case it can be shown that the number of these components is at least $k+1$. (However, here $H_1$ may be a forest rather than a tree.)

Denote the connected components of $H_1$ by $C_1, \ldots, C_{k+1}, \ldots$. Let $\text{OPT}_i$ denote the vertices in OPT that are the neighbors of $C_i$. Since every vertex of OPT may be the neighbor of at most $\Delta$ components we get that $\sum_{i=1}^{k+1} w(\text{OPT}_i) \leq \Delta w(\text{OPT})$. To show that the approximation factor of our algorithm is $\Delta$, we show an injective mapping from the set of $k$ minimum cuts computed by the algorithm to the set of components of $H_1$. This mapping has the property that for each minimum cut $M_i$ mapped to component $C_j$, $w(M_i) \leq w(\text{OPT}_j)$. Since the mapping is injective, we have $\sum_{i=1}^{k} w(M_i) \leq \Delta w(\text{OPT})$.

Similar to the edge case we show the injective mapping in two stages. First, for each minimum cut $M_i$ we identify two components $C_{\ell(i)}$ and $C_{r(i)}$ such that $w(M_i) \leq w(\text{OPT}_{\ell(i)})$ and $w(M_i) \leq w(\text{OPT}_{r(i)})$. Then, we show how to map each $M_i$ to one of these two components so that the mapping is injective.

The component $C_{\ell(i)}$ ($C_{r(i)}$) is given by following a path $p$ connecting $x_i$ and $y_i$ in $G_i$, starting at $x_i$ ($y_i$), until the first vertex that belongs to both $p$ and OPT. The component $C_{\ell(i)}$ ($C_{r(i)}$) is the component containing the vertex preceding this first vertex. Again, it can be shown that these two components are independent of the choice of $p$. The injective mapping is computed using an auxiliary forest similar to the auxiliary tree used in the edge case.

## 3. Bootstrapping for approximating multicuts.

Suppose that we have a SUBSET-FVS approximation algorithm where the approximation factor depends on the number of special vertices. Consider the following bootstrapping technique applied to such a SUBSET-FVS approximation algorithm. Given a subset feedback vertex set $F$, redefine the set of special vertices to be $F$, and now apply the SUBSET-FVS approximation algorithm to obtain a new subset feedback vertex set $F'$ with respect to $F$. Observe that $F'$ intersects all the interesting cycles with respect to the original set of special vertices. We can continue this process and generate new subset feedback vertex sets by using the current subset feedback vertex set as the set of special vertices for the next iteration. Does this process provide improved approximation factors? Not for the SUBSET-FVS problem, since this problem can be approximated within a constant factor [11]. However, this technique can be used to enhance the approximation factor for the undirected multicut problem and for the directed feedback set problem. Below, we describe the application of the bootstrapping technique to the multicut problem.

The undirected multicut problem is defined as follows. The input for the "vertex" version consists of a graph $G = (V, E)$ with vertex weights $w : V \rightarrow \mathbb{R}^+ \cup \{\infty\}$ and $k$ terminal pairs $\{s_i, t_i\}_{i=1,\ldots,k}$. A (vertex) multicut is a subset of the vertices that separates all terminal pairs. The goal is to find a vertex multicut of minimum weight. In the "edge" version weights are assigned to edges, and the goal is to find a subset of the edges of minimum weight that separates all terminal pairs. The edge version can be reduced to a vertex by splitting each edge $e$ into two edges with a new vertex $v_e$ in between, giving each new vertex $v_e$ the weight of the original edge $e$ and assigning infinite weight to the original vertices. Thus, from now on we consider only the vertex version.

Garg, Vazirani, and Yannakakis [13] presented an $O(\log k)$ approximation algorithm for undirected multicuts. For their algorithm they defined fractional multicuts.

DEFINITION 3.1. *A* fractional multicut *is an assignment of (nonnegative) length $d(v)$ to each vertex $v \in V$ such that under this assignment the distance between any terminal pair is at least one. In other words, a* fractional multicut *is a function $d : V \rightarrow \mathbb{R}^+$ such that $\sum_{v \in p} d(v) \geq 1$ for every path $p$ connecting a terminal pair $\{s_i, t_i\}_{i=1,\ldots,k}$.*

Note that if we restrict $d(v)$ to be either 0 or 1, the vertices assigned length one constitute a multicut.

DEFINITION 3.2. *An* optimal *fractional multicut is a multicut that minimizes $\sum_{v \in V} w(v)d(v)$. Define $\tau$ to be the weight of an optimal fractional multicut.*

The algorithm of Garg, Vazirani, and Yannakakis [13] proceeds as follows. First, an optimal fractional multicut is computed by solving the corresponding linear program. This fractional multicut is then rounded to an integral multicut using a region growing procedure. Garg, Vazirani, and Yannakakis [13] prove that the sum of the weights of the vertices in the multicut is bounded by $4\tau \cdot \ln(k + 1)$. The algorithm of [13] has the property that, after computing the fractional multicut, it no longer considers the $t_i$-s. We reformulate the region growing procedure of [13] as a procedure that finds a subset of the vertices that intersects every path of length at least one that contains a vertex of the set $R_0 = \{s_i\}_{i=1}^{k}$. Note that such a subset of vertices constitutes a multicut since the distance between every pair of terminals is at least one. The total weight of these vertices is bounded by $4\tau \cdot \ln(k + 1)$.

We can now describe the bootstrapping technique. We start with the set $R_0$ containing all the terminals $s_i$, for $1 \leq i \leq k$, and find a subset of vertices that intersects every path of length at least one that contains a vertex of $R_0$. Denote this subset of vertices by $R_1$. Now, we find a subset of vertices that intersects every path of length at least one that goes through $R_1$. Denote this subset of vertices by $R_2$. We continue in this manner, i.e., at step $i$ we find a subset $R_{i+1}$ of the vertices that intersects every path of length at least one that goes through $R_i$. From Garg, Vazirani, and Yannakakis [13] it follows that the total weight of the vertices in $R_{i+1}$ is bounded by $4\tau \cdot \ln(|R_i|+1)$.

First, we show that each of the subsets $R_i$, for $i \geq 1$, is indeed a multicut.

CLAIM 3.3. *Each of the subsets $R_i$, for $i \geq 1$, is a multicut.*

*Proof.* The proof is by induction on $i$. From the discussion above it follows that $R_1$ is a multicut. Suppose that $R_i$ is a multicut. This implies that $R_i$ intersects every path between $s_j$ and $t_j$, $1 \leq j \leq k$. Since $R_{i+1}$ intersects every path of length at least one that goes through a vertex in $R_i$, and since the length of every path from $s_j$ to $t_j$ is at least one, we get that $R_{i+1}$ intersects every path connecting $s_j$ to $t_j$, $1 \leq j \leq k$. $\square$

We now prove that the bootstrapping technique enhances the approximation factor from $O(\log k)$ to $O(\log \tau)$. Since $\tau$ is a lower bound on the weight of an optimal multicut, bootstrapping improves the approximation factor in cases where the optimal multicut is significantly lighter than $k$.

THEOREM 3.4. *Suppose that $w(v) \geq 1$ for all $v \in V$. If we apply the bootstrapping technique for $O(\log^* k)$ iterations, then we find a multicut whose weight is $O(\tau \cdot \log \tau)$.*

*Proof.* Define

$$f(i) \triangleq \begin{cases} \ln(k+1) & \text{if } i = 1, \\ \ln(2 \cdot f(i-1)) & \text{if } i > 1 \text{ and } f(i-1) > 0, \\ 0 & \text{if } i > 1 \text{ and } f(i-1) \leq 0, \end{cases}$$

and

$$t_0 \triangleq \min\{i : f(i) \leq 1 + \ln 4 + \ln \tau\}.^1$$

The following lemma bounds the cardinality of $R_i$ as a function of $i$.

LEMMA 3.5. *If $1 \leq i \leq t_0$, then $1 + |R_i| \leq 4 \cdot \tau \cdot (1 + \ln 4 + \ln \tau + f(i))$.*

*Proof.* The proof is by induction on $i$. Since all vertex weights are at least one, then $|R_i| \leq w(R_i)$. The algorithm of Garg, Vazirani, and Yannakakis guarantees that $w(R_{i+1}) \leq 4 \cdot \tau \cdot \ln(|R_i|+1)$. Hence, $|R_1| \leq 4 \cdot \tau \cdot \ln(k+1)$, and the induction basis follows.

The induction step is proven as follows:

$$\begin{aligned} 1 + |R_{i+1}| &\leq 1 + w(R_{i+1}) \\ &\leq 1 + 4 \cdot \tau \cdot \ln(1 + |R_i|) \\ &\leq 1 + 4 \cdot \tau \cdot \ln[4 \cdot \tau \cdot (1 + \ln 4 + \ln \tau + f(i))], \end{aligned}$$

where the last inequality follows from the induction hypothesis. Since $i < t_0$, it follows that $1 + \ln 4 + \ln \tau < f(i)$; hence,

$$\begin{aligned} 1 + |R_{i+1}| &\leq 1 + 4 \cdot \tau \cdot (\ln 4 + \ln \tau + \ln(2 \cdot f(i))) \\ &\leq 4 \cdot \tau \cdot (1 + \ln 4 + \ln \tau + f(i+1)). \quad \square \end{aligned}$$

[1]We use the notation "ln" to denote the natural basis logarithm and "log" to denote the base 2 logarithm.

It is not difficult to verify the following claim.

CLAIM 3.6. *If $i \geq 2 \cdot \log^* k + 3$, then $f(i) \leq 2$.*

By the definition of $t_0$ and from Claim 3.6 it follows that $t_0 \leq 2 \cdot \log^* k + 3$. From Lemma 3.5 it follows that

$$1 + |R_{t_0}| \leq 4 \cdot \tau \cdot 2(1 + \ln 4 + \ln \tau) = O(\tau \cdot \log \tau).$$

Since $w(R_{t_0+1}) \leq 4 \cdot \tau \cdot \ln(|R_{t_0}| + 1)$, it follows that $w(R_{t_0+1}) = O(\tau \cdot \log \tau)$, and the theorem follows. ☐

## 4. Linear programming formulation and integrality gap.

In this section we cast the FVS, FES, SUBSET-FVS, and SUBSET-FES problems as integer programs. The formulation chosen is the "natural" one, i.e., as covering problems. The *integrality gap* of an integer program is defined to be the ratio between the integer and fractional optima. We show that the integrality gap in the FVS and FES problems can be as big as $\Omega(\log |V|)$. This is used to further show that the integrality gap in the SUBSET-FVS and SUBSET-FES problems can be as big as $\Omega(\log |S|)$.

We note that in spite of this gap, for the FES problem in undirected graphs we can find an exact solution in polynomial time (by computing a maximum weight spanning tree), and for the FVS problem in undirected graphs, we can find a constant factor approximation in polynomial time [1, 2, 3]. Interestingly, Chudak et al. [7] recently gave an alternative integer programming formulation for the FVS problem for which the gap is two; see also [16].

Let $G = (V, E)$ be an undirected graph. In this section we assume that all weights in $G$ are unit weights. Let $|V| = n$, and let the set of special vertices be $S = \{s_1, \ldots, s_k\}$. We denote by $\mathcal{C}$ the set of cycles in $G$ and by $\mathcal{C}_v$ the set of cycles passing through a particular vertex $v$.

### 4.1. Integer programming formulation.

Let $t : V \to [0, 1]$ be an indicator variable for membership in a feedback set of $G$. The following is a linear programming formulation of the FVS problem. (The formulation of the FES problem is the same except that $t$ is an indicator variable for the edge set.)

$$\begin{array}{rl} \text{minimize} & \sum_{v \in V} w(v) \cdot t(v) \\ \text{such that} & \displaystyle\sum_{v \in C} t(v) \;\geq\; 1 \quad \text{for every} \quad C \in \mathcal{C}\ . \end{array}$$

The formulation of the SUBSET-FES and SUBSET-FVS problems remains the same, except that the constraints are only with respect to cycles $C$ belonging to $\cup_{s \in S} \mathcal{C}_s$.

Note that the above linear program may contain, in general, an exponential number of constraints. However, since a violated constraint of a given nonfeasible solution can be found in polynomial time, an optimal solution can be either computed by the ellipsoid algorithm or approximated (very closely) by the techniques presented in [22].

### 4.2. Integrality gap.

We use the following construction for showing the integrality gap. Let $H_n = (V, E)$ be a connected, 3-regular graph, where $|V(H_n)| = n$, and the girth of $H_n$ is $\lceil \log n \rceil$. Such graphs exist and can be constructed explicitly. (See, e.g., Bollobás [6, pp. 108–110].)

CLAIM 4.1. *The value of the optimal fractional* FVS *solution in $H_n$ is at most $n/\lceil \log n \rceil$. The value of the optimal fractional* FES *solution in $H_n$ is at most $1.5n/\lceil \log n \rceil$.*

*Proof.* The girth of $H_n$ is $\lceil \log n \rceil$. Thus, to get a feasible fractional solution for the FVS problem, assign each vertex the value $1/\lceil \log n \rceil$. To get a feasible solution for the fractional FES problem, assign each of the $1.5n$ edges the value $1/\lceil \log n \rceil$. ∎

Clearly, an optimal integral solution for the FES problem is the complement of any spanning tree of $H_n$ whose cost is $1.5n - (n-1) = n/2 - 1$. This establishes the gap for the FES problem. To establish the gap for the FVS problem we need the following claim.

CLAIM 4.2. *The value of an optimal integral solution for the* FVS *problem in $H_n$ is $\Omega(n)$.*

*Proof.* We construct an instance of the FES problem on $H_n$ by assigning each edge unit weight. Clearly, an optimal solution to the FES problem costs at most $\Delta = 3$ times the cost of an optimal solution to the FVS problem. Since the cost of this FES problem is $n/2 - 1$, this yields that the cost of the FVS problem is at least $n/6 - 1/3 = \Omega(n)$. ∎

We show an $\Omega(\log k)$ integrality gap for the SUBSET-FVS and SUBSET-FES problems in the following graph $F_n$, which is the union of two graphs: $H_k = (S, E)$ on $k$ *special* vertices, as defined above, and a clique on the remaining $n - k$ vertices. To make $F_n$ connected we connect $H_k$ to the clique by a *single* edge. Note that no cycle that goes through a vertex in $S$ intersects the clique. This fact, together with the integrality gap shown above, implies the $\Omega(\log k)$ integrality gap for the SUBSET-FVS and SUBSET-FES problems.

**Acknowledgments.** We are very grateful to Dan Geiger and Alejandro Schäffer for explaining to us the application of the subset feedback set problem to genetic linkage analysis. The bootstrapping technique was developed jointly with Madhu Sudan. We thank Madhu Sudan for his permission to include it here. We would like to thank Naveen Garg for useful discussions and Satish Rao for pointing out a missing explanation in section 3. We thank the referees for helpful comments and especially for simplifying the analysis of the algorithm.

## REFERENCES

[1] R. BAR-YEHUDA, D. GEIGER, J.S. NAOR, AND R.M. ROTH, *Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference*, SIAM J. Comput., 27 (1998), pp. 942–959.

[2] V. BAFNA, P. BERMAN, AND T. FUJITO, *A 2-approximation algorithm for the undirected feedback vertex set problem*, SIAM J. Discrete Math., 12 (1999), pp. 289–297.

[3] A. BECKER AND D. GEIGER, *Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem*, Artificial Intelligence, 83 (1996), pp. 167–188.

[4] A. BECKER, D. GEIGER, AND A.A. SCHÄFFER, *Automatic selection of loop breakers for genetic linkage analysis*, Human Heredity, 48 (1998), pp. 49–60.

[5] A. BOLINO, V. BRANCOLINI, F. BONO, A. BRUNI, A. GAMBARDELLA, G. ROMEO, A. QUATTRONE, AND M. DEVOTO, *Localization of a gene responsible for autosomal recessive demyelinating neuropathy with focally folded myelin sheaths to chromosome 11q23 by homozygosity mapping and haplotype sharing*, Human Molecular Genetics, 5 (1996), pp. 1051–1054.

[6] B. BOLLOBÁS, *Extremal Graph Theory*, Academic Press, New York, 1978.

[7] F.A. CHUDAK, M. X. GOEMANS, D. S. HOCHBAUM, AND D. P. WILLIAMSON, *A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs*, Oper. Res. Lett., (22) 1998, pp. 111–118.

[8] E. DAHLHAUS, D.S. JOHNSON, C.H. PAPADIMITRIOU, P.D. SEYMOUR, AND M. YANNAKAKIS, *The complexity of multiterminal cuts*, SIAM J. Comput., 23 (1994), pp. 864–894.

[9] G. EVEN, J. NAOR, B. SCHIEBER, AND M. SUDAN, *Approximating minimum feedback sets and multi-cuts in directed graphs*, Algorithmica, 20 (1998), pp. 151–174.

[10] G. Even, J. Naor, S. Rao, and B. Schieber, *Divide-and-conquer approximation algorithms via spreading metrics*, in Proceedings of the 36th Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, Los Alamitos, CA, 1995, pp. 62–71; J. ACM, to appear.

[11] G. Even, J. Naor, and L. Zosin, *An 8-approximation algorithm for the subset feedback vertex set problem*, in Proceedings of the 36th Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, Los Alamitos, CA, 1996, pp. 310–319; SIAM J. Comput., to appear.

[12] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman, San Francisco, CA, 1979.

[13] N. Garg, V.V. Vazirani, and M. Yannakakis, *Approximate max-flow min-(multi)cut theorems and their applications*, SIAM J. Comput., 25 (1996), pp. 235–251.

[14] N. Garg, V.V. Vazirani, and M. Yannakakis, *Multiway cuts in directed and node weighted graphs*, in Proceedings of the 21st International Colloquium on Automata, Languages and Programming (ICALP), Lecture Notes in Comput. Sci. 820, Springer-Verlag, New York, 1994, pp. 487–498.

[15] M.X. Goemans and D.P. Williamson, *Primal-dual approximation algorithms for feedback problems in planar graphs*, Combinatorica, 18 (1998), pp. 37–59.

[16] D.S. Hochbaum, *Chapter* 9: *Good, better, best, and better than best approximation algorithms*, in Approximation Algorithms for NP-Hard Problems, D.S. Hochbaum, ed., PWS Publishing Company, Boston, MA, 1996.

[17] R.M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., Plenum Press, New York, 1972, pp. 85–104.

[18] P.N. Klein, S.A. Plotkin, S. Rao, and É. Tardos, *Approximation algorithms for Steiner and directed multicuts*, J. Algorithms, 22 (1997), pp. 241–269.

[19] T. Leighton and S. Rao, *An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms*, in Proceedings of the 29th Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, Los Alamitos, CA, 1988, pp. 422–431; J. ACM, to appear.

[20] J. Ott, *Analysis of Human Genetic Linkage*, revised ed., The John Hopkins University Press, Baltimore and London, 1991.

[21] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA, 1988.

[22] S. Plotkin, É. Tardos, and D. Shmoys, *Fast approximation algorithms for fractional packing and covering problems*, Math. Oper. Res., 20 (1995), pp. 257–301.

[23] P.D. Seymour, *Packing directed circuits fractionally*, Combinatorica, 15 (1995), pp. 281–288.

SIAM J. Discrete Math.
Vol. 13, No. 2, pp. 268–279

# DIRECTIONAL ROUTING VIA GENERALIZED *st*-NUMBERINGS*

FRED S. ANNEXSTEIN† AND KENNETH A. BERMAN†

**Abstract.** We present a mathematical model for network routing based on generating paths in a consistent direction. Our development is based on an algebraic and geometric framework for defining a directional coordinate system for real vector spaces. Our model, which generalizes graph *st*-numberings, is based on mapping the nodes of a network to points in multidimensional space and ensures that the paths generated in different directions from the same source are node-disjoint. Such directional embeddings encode the global disjoint path structure with very simple local information. We prove that all 3-connected graphs have 3-directional embeddings in the plane so that each node outside a set of extreme nodes has a neighbor in each of the three directional regions defined in the plane. We conjecture that the result generalizes to $k$-connected graphs. We also show that a directed acyclic graph (dag) that is $k$-connected to a set of sinks has a $k$-directional embedding in $(k-1)$-space with the sink set as the extreme nodes.

**Key words.** graph connectivity, network routing, *st*-numbering, matchings

**AMS subject classifications.** 68R10, 05C40, 68R10

**PII.** S0895480198333290

## 1. Introduction.

A fundamental problem in network routing is the generation of communication paths from a set of source nodes $Y$ to a set of sink nodes $X$. Routing schemes for communication networks are often implemented using local tables that logically associate each destination address $x$ with a parent link on which to forward messages to $x$ (see [3, 6]). The set of parent links forms a directed *sink tree* to the sink node $x$. More generally, given a set of sink nodes $X$, we define a *sink tree to* $X$ as a set of parent links, with one parent for each $u \notin X$, forming a union of directed in-trees, where each in-tree has a single sink node from $X$. When used in routing applications, the tables associated with such sink trees suffer from the fact that they do not provide a full representation of the entire network and therefore are vulnerable to faults and other dynamic network changes. One approach to the problem of fault-tolerant routing is the study of collections of *independent* sink trees and closely related graph *st*-numberings [10, 2]. A pair of sink trees to $X$ is *independent* if it has the property that for each node $u \notin X$, the pair of tree-paths from $u$ to $X$ is internally node-disjoint. A standard way to obtain a pair of independent sink trees is via an *st*-numbering of a biconnected graph. Let $G = (V, E)$ be a graph with distinguished vertices $s, t$. An *st-numbering* of $G$ is a mapping $f$ of the vertices into the real line, $f : V \to R$, such that for all $v \in V - \{s, t\}$, there are neighbors $u, w \in N(v)$ such that $f(u) < f(v) < f(w)$. A 2-connected graph has an *st*-numbering for any vertex pair $s, t$ [11, 8]. In general, collections of $k > 2$ sink trees are independent if they are pairwise independent. There is a long-standing open conjecture of Frank (see [13]) that states that all $k$-connected graphs have, for each vertex $v$, $k$-independent sink trees to $v$. One goal of this work is to provide a generalized graph *st*-numbering in an attempt to characterize graphs having $k$-independent sink trees.

*Received by the editors January 29, 1998; accepted for publication (in revised form) September 22, 1999; published electronically April 6, 2000.
http://www.siam.org/journals/sidma/13-2/33329.html

†Department of ECE & Computer Science, University of Cincinnati, Cincinnati, OH 45221 (fred.annexstein@uc.edu, ken.berman@uc.edu). This research was supported in part by NSF grant CCR-9877139 and by an OBR investment grant.

An $st$-numbering provides a stronger model for routing applications than that which is provided by the (induced) independent sink trees in the following sense: For each pair of vertices $u, v \notin X = \{s, t\}$, there exists a pair of tree-paths from distinct sink trees, one from $u$ to $X$ and the other from $v$ to $X$, that is internally node-disjoint. We say that a pair of sink trees with this property is *strongly independent*. A collection of $k > 2$ sink trees is *strongly independent* if for each set $Y$ of $k$ vertices $\{y_1, \ldots, y_k : y_i \notin X\}$ there exists a matching $M$ of $Y$ to the $k$ trees such that the collection of induced tree-paths (each routing a distinct $y_i$ to $X$) is internally node-disjoint. We strengthen Frank's conjecture by conjecturing that every $k$-connected graph has $k$ strongly independent sink trees, and we prove our conjecture for $k = 3$ using a generalized $st$-numbering that yields the matching $M$ of $Y$ to the three sink trees.

Our generalization of graph $st$-numbering is based on an algebraic and geometric framework for defining directions in real vector spaces. Using this framework we propose a mathematical model for network routing based on *directional graph embeddings with compass*. We show that such embeddings naturally induce collections of strongly independent sink trees and thus encode the global disjoint path structure with very simple local information. A $k$-directional embedding of a graph $G$ involves mapping the vertices $V(G)$ to $(k-1)$-space so that each vertex, outside a set of extreme vertices, has a neighbor in each of $k$ distinct directions. Routing in the directional embedding model is based on generating routing paths that are *monodirectional*, i.e., the paths follow a consistent direction. The mechanisms of the model ensure that the monodirectional paths generated in different directions from the same source are node-disjoint.

The compass associated with our embeddings yields directions (or equivalently, directional regions relative to any fixed location) that possess a fundamental matching property of combinatorial interest. Namely, any set of $k$ distinct points in $(k-1)$-dimensional space can be matched to the set of $k$ directions defined by the compass. The $k$ points are matched in the sense that there exists a fixed location point $s$ such that each of the original $k$ points lies in a distinct directional region with respect to $s$. Furthermore, we characterize the set of all such points that satisfy this property, and we provide the conditions under which the associated matching is unique.

In this paper we apply these developments to prove that all 3-connected graphs have 3-directional embeddings in the plane. We conjecture that the result generalizes to the case of $k$-connected graphs, for each $k > 3$. Supporting this conjecture, we prove that a directed acyclic graph (dag) that is $k$-connected to a set of sinks has a $k$-directional embedding in $(k-1)$-space. A proof of our conjecture would affirm the conjecture of Frank.

**Comparisons with previous work.** The concept of graph $st$-numberings has been generalized to geometric embedding of graphs in Euclidean space in Linial, Lovász, and Widgerson [12]. Cheriyan and Reif [5] later extended the results to digraphs. This generalization of $st$-numberings originated by appealing to potential functions in physics (e.g., rubber bands or electricity), leading to the notion of a *convex embedding in general position*. A graph has a convex $X$-embedding $f$ if $f$ is a mapping of the vertices to points in real space $f : V \to R^{|X|-1}$ such that, for each $v \in V - X, f(v)$ is in the convex hull of $f(N(v))$. A convex embedding is in general position if $f(V)$ is in general position. Convex embeddings in general position characterize $k$-connected graphs, analogous to the $st$-numbering characterization of 2-connected graphs. Unfortunately, from the point of view of path generation, convex

$X$-embeddings for $|X| \geq 3$ neither seem to provide a model from which to extract structural information concerning disjoint paths to $X$ nor seem to provide a means to generate independent sink trees.

Our conjecture that $k$-connected graphs have $k$-directional embeddings in $R^{k-1}$ is equivalent to showing that convex embeddings in general position can be transformed to nondegenerate directional embeddings (see section 4). Our result that 3-connected graphs have 3-directional embeddings implies that 3-connected graphs have 3-independent spanning trees, which was previously shown by Zehavi and Itai [15] and Cheriyan and Maheshwari [4].

**2. A directional compass.** The basic framework for defining directions associated with a directional embedding is a natural one, obtained by defining a *$k$-directional compass* on the geometry of real vector spaces. A compass is given by a *direction function* $d : R^{k-1} \times R^{k-1} \to D$, where $D = \{1, 2, \ldots, k\}$ is a set of $k$ directions, and has the following two properties: (1) A compass is *translatable* from the origin $\bar{\mathbf{0}}$, i.e., the direction of vector $y$ relative to a reference vector $x$ is $d(x, y) = d(\bar{\mathbf{0}}, y - x)$, for every pair of points $x, y$; and (2) a compass has *directional transitivity*, i.e., if $d(x, y) = d(y, z)$, then $d(x, y) = d(x, z)$, for every set of points $x, y, z$. From these two properties it follows immediately that for graphs embedded in $k$-space, monodirectional paths, i.e., paths that consistently follow the same direction, originating from the same vertex and following different directions, are internally node-disjoint. As a simple illustrative example, consider a graph embedded on the real line along with a 2-directional compass (on the real line) given by the positive and negative directions. It is clear that positive directed paths are disjoint from negative directed paths, when originating from the same node. It is this notion of directions in graphs that we wish to generalize to higher dimensions.

The generalized compass that we will apply enjoys a useful directional splitting property that is also enjoyed by the 2-directional compass on the real line. This splitting property of a compass is particularly useful for algorithmic constructions of directional embeddings.

Definition. *A compass with $k$ directions has the* $k$-directional splitting property *if, given any set of $k$ distinct points $Y = \{y_1, y_2, \ldots, y_k\}$, there exists a reference point $s = s_Y$, called a* $k$-directional splitter point*, and an associated* directional permutation *$\pi_Y$, so that the point $y_{\pi_Y(i)}$ is in the ith direction relative to $s$, i.e., $d(s, y_{\pi_Y(i)}) = i$. A splitter point $s$ is called a* strict-splitter *if each of the $k$ points lies strictly in the interior of a directional region with reference to $s$, i.e., no point lies on a boundary shared by more than one region.*

The compass for the plane $R^2$ using the four natural directions $N,S,E,W$ does not satisfy the 4-directional splitting property, as seen in Figure 1. We solve the problem of directional splitting by using a compass with three directions in $R^2$ and show that this extends to $R^k$ using a compass with $k + 1$ directions.

We define a compass so that directional regions are associated with the space spanned by a given fixed set of vectors. To begin, let $L$ be a fixed set of lines in $R^k$ through the origin. Divide each line $l_i \in L$ at the origin into two half-lines and arbitrarily label one half-line as *ray-$i$* and the other half-line as *antiray-$i$*. We say that a particular labeling is a *convex bipartition labeling* of $L$ if the set of chosen rays are *convex-spanning*, i.e., the entire space $R^k$ is generated by positive (convex) linear combinations of vectors that lie on the rays. It follows that if the set of rays are convex-spanning, then the set of antirays are convex-spanning too. Clearly, in $k$-space, at least $k+1$ lines are required for there to exist a set of rays that is convex-spanning.

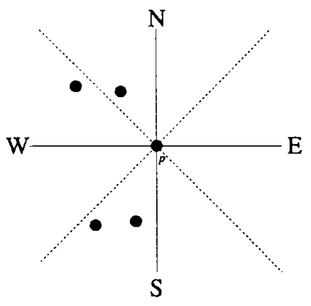


FIG. 1. *Four points that have no* 4*-directional splitter point. With respect to the reference point p shown, only three of four directional regions are covered. The dashed lines depict the boundaries of the directional regions. Further, no matter where p is positioned, the four fixed points cannot be located in four distinct directional regions relative to p.*

A set of points $X$ in $k$-space is in *general position* if no $d$-element subset of $X$ lies on a hyperplane of dimension $d-2$, for any $d \leq k+1$. We say that a set of lines $L$ in $k$-space is in *general position* if the lines intersect at the origin and any set $X$ of nonorigin points, chosen one per line, is in general position. The following shows that a set of $k+1$ lines in $k$-space that is in general position has a convex bipartition labeling, and this partition labeling is unique up to the naming of rays.

PROPOSITION 1. *Let $L$ be a set of $k$ lines in general position in $(k-1)$-space with one half-line from $L$ marked as ray-*1*. Then there is a unique convex bipartition labeling of $L$.*

*Proof.* Arbitrarily order $L$, say, $\{l_1, l_2, \ldots, l_k\}$, where $l_1$ has the marked half-line. Label the marked half-line as ray-1, and label the other half-line as antiray-1. We now show that the remaining lines are uniquely labeled in a convex bipartitioning. Consider $l_2$ and the $(k-2)$-dimensional hyperplane $H$ spanned by the remaining lines $\{l_3, l_4, \ldots, l_k\}$. Since $H$ divides the original $(k-1)$-space into two half-spaces, it follows that a unique half-line of $l_2$ lies on the opposite side of $H$ as the ray-1. Label this particular half-line as ray-2. Continuing in this way, each line $l_i \in L$ uniquely partitions into ray-$i$ and antiray-$i$. The collection of rays must be convex-spanning since the lines are in general position. Any other labeling (in which ray-1 was fixed) could not be convex-spanning; thus the partition labeling is unique. □

DEFINITION. *Let $L$ be a set of $k$ lines in general position equipped with a convex bipartition labeling which marks half-lines as $k$ rays and $k$ antirays. The* compass $\mathcal{C}_k(L)$ *defined by $L$ demarcates the* $k$-directional regions *relative to any fixed reference point. Each region is defined by the convex combinations of $k-1$ antirays chosen from the set of $k$ antirays originating at the fixed reference point.*

The antirays thus mark the "boundary corners" of each region. Each of the $k$-directional regions is named by the index of the unique ray it contains (or equivalently, by the antiray it omits). The direction of a point that lies on the boundary (hyperplane) dividing more than one region is ambiguous, but by convention we consider it to have multiple directions, one for each region associated with the boundary.

There is, of course, a natural dual-compass demarcating *directional antiregions* defined by convex combinations of $k-1$ rays chosen from the set of $k$ rays. Here the rays mark the boundary corners of directional antiregions, and each of the $k$-directional antiregions is named by the index of the unique antiray it contains. Antiregions are useful for identifying those points that have a fixed direction to a fixed reference point

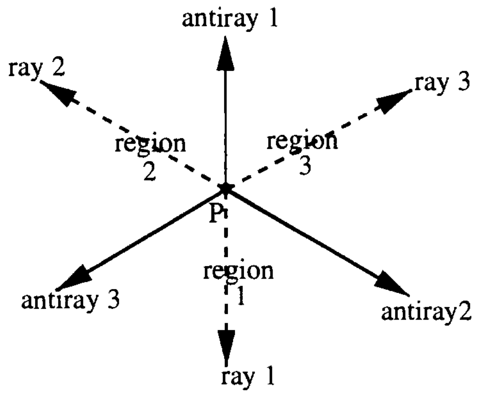


FIG. 2. *Compass with three regions in the plane from reference point p, obtained by a convex bipartitioning of three lines.*

and for helping to locate splitter points.

For example, in Figure 2 we have a compass $\mathcal{C}_3(L)$ in the plane, yielding directional regions that are symmetric. In this figure we see that region 1 is the set of convex combinations of points on antiray-2 and antiray-3. Dually, antiregion-1 (not labeled) is the set of convex combinations of points on ray-2 and ray-3 and defines precisely those points $x$ for which point $P$ lies in region 1, i.e., $d(x, P) = 1$. We can find a splitter point of a set of three points $\{A, B, C\}$, as shown in Figure 3, by considering the antiregions defined at each point. The splitter points are found in an intersection of three differently named antiregions. This intersection region associates each of the points $\{A, B, C\}$ to a different region. Note that in Figure 3 the set of all splitter points forms a convex region, and any interior splitter point divides $\{A, B, C\}$ into directional regions in the same way.

In the next section we show how to formalize this notion of matching points to directional regions, and we show how this extends to higher dimensions. We show that the set of splitter points always forms a convex region $C$ and the set of strict-splitters is associated with the interior of $C$. Further, we show that there is a function that gives a weight to each point-direction pair such that maximum weight matchings correspond to the region $C$. The weighting is obtained by a special set of directional coordinates discussed in the next section.

## 3. A directional coordinate system and splitting lemma.

For a given compass $C_k(L)$ in $(k-1)$-space, we now show that there exists a transformation from Euclidean coordinates to a *directional coordinate system.* This new system uses $k$-tuples in a manner where each fixed directional region (from the origin) is precisely those points whose $i$th component is maximum (dominant). The directional coordinatization is defined as follows. First, fix a collection $\{v_1, v_2, \ldots, v_k\}$ of $k$ nontrivial vectors on the $k$ rays defined by $C_k(L)$, precisely one per ray, such that $\sum v_i = \bar{0}$, the origin. Note that this can always be done since the rays are assumed to be convex-spanning.

To obtain directional coordinates, consider the linear transformation $\mathcal{T} : R^k \to R^{k-1}$ that maps each vector $z$ in $k$-space with Euclidean coordinates $z = (z_1, z_2, \ldots, z_k)$ to the vector $-(z_1v_1 + z_2v_2 + \cdots + z_kv_k)$ in $(k-1)$-space. The nullspace of this linear mapping is the line through the "all-1s" vector $(1, 1, \ldots, 1)$. Thus for any two points

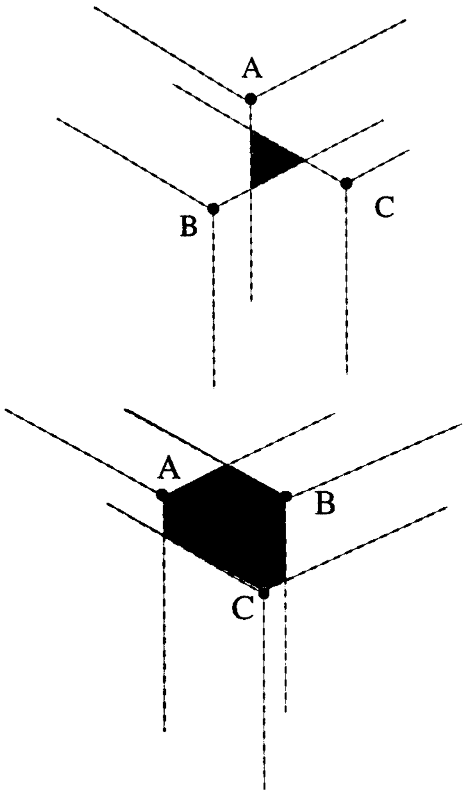


FIG. 3. *Two examples, each showing a set of three points $\{A, B, C\}$, the antiregions from each point, and the set of all splitter points shown as a shaded polytope. The directional (anti-)regions are defined by the compass in Figure* 2. *From each splitter point in the interior of the polytope, the points $\{A, B, C\}$ are matched to (lie in) distinct regions. In the top figure, each interior point matches A to region* 2*, B to region* 3*, and C to region* 1*. In the bottom figure, each interior point matches A to region* 2*, B to region* 1*, and C to region* 3.

$x_1, x_2 \in \mathcal{T}^{-1}(y)$ in the inverse transformation $\mathcal{T}^{-1}(y)$ of a point $y \in R^{k-1}$, we have that the differences between components are identical, and thus the minimum or maximum component of $\mathcal{T}^{-1}(y)$ is well defined. We will use angled brackets $\langle\rangle$ to represent the equivalence class and thereby denote the *directional coordinates.* So for example, we say that $\mathcal{T}^{-1}(y) = \langle y_1, y_2, \ldots, y_k\rangle$ is the set of directional coordinates associated with a point $y \in R^{k-1}$. As the following proposition shows, the maximum directional component of $\mathcal{T}^{-1}(y)$ identifies the directional region in $(k-1)$-space within which the point $y \in R^{k-1}$ lies, with respect to the origin.

First, we illustrate the concept of directional coordinates using the example of the compass from Figure 2. As noted, we first choose three vectors $\{v_1, v_2, v_3\}$ on these rays so that they sum to zero. We give the Euclidean coordinates of these vectors as follows: $v_1 = (0, -2)$ on ray-1, $v_2 = (-\sqrt{3}, 1)$ on ray-2, and $v_3 = (\sqrt{3}, 1)$ on ray-3. Now, say we are given a point $q$ in the plane with Euclidean coordinates $q = (q_1, q_2)$. Under the inverse transformation $\mathcal{T}^{-1}(q)$ we obtain directional coordinates for $q$ given by $q = \langle z_1, z_2, z_3\rangle$. By setting $q = -(z_1 v_1 + z_2 v_2 + z_3 v_3)$ and using the Euclidean coordinates for each $v_i$, we can solve for directional coordinates as follows: $z_1 = 0, z_2 = q_2/2 - q_1/2\sqrt{3}$, and $z_3 = z_2 + q_2$. So, for example, the point $q_0 = (0, 1)$

in Euclidean space has directional coordinates $q_0 = \langle 0, -\sqrt{3}/6, \sqrt{3}/6\rangle$. Since the maximum directional coordinate is the third, we have that $q_0$ lies in region 3.

PROPOSITION 2. *The linear transformation $\mathcal{T} : R^k \to R^{k-1}$ is an onto mapping with the property that the point $x \in R^{k-1}$ lies in the ith directional region $(1 \le i \le k)$ relative to the origin iff the maximum directional-component of $\mathcal{T}^{-1}(x)$ is the ith component.*

*Proof.* The transformation $\mathcal{T}$ is an onto mapping since the collection of vectors $\{v_1, v_2, \ldots, v_k\}$ spans $(k-1)$-space. By definition, a point $x \in R^{k-1}$ is in the $i$th directional region iff it is a convex combination of the antirays different from antiray-$i$, i.e., $x = \sum_{j \ne i} \lambda_j(-v_j)$, where each $\lambda_j \ge 0$ and each $-v_j$ is a vector on antiray-$j$. Hence, the directional coordinatization yields $\mathcal{T}^{-1}(x) = \langle -\lambda_1, -\lambda_2, \ldots, -\lambda_{i-1}, 0, -\lambda_{i+1}, \ldots, -\lambda_k\rangle$, and thus the $i$th component is maximum. ☐

The following proposition is easily verified by applying Proposition 2.

PROPOSITION 3. *The directional compass is translatable and has directional transitivity, as defined in section 2.*

Within this directional coordinate system for $(k-1)$-space we can identify the problem of finding a directional permutation with the problem of finding a maximum perfect matching in an associated weighted bipartite graph or, equivalently, finding a maximum weight permutation. To wit, let $Y = \{y_1, y_2, \ldots, y_k\}$ be a set of $k$ points in Euclidean space $R^{k-1}$, and for each $1 \le i \le k$, let $\langle y_{i,1}, y_{i,2}, \ldots, y_{i,k}\rangle$ denote (a representative of) $\mathcal{T}^{-1}(y_i)$. For $\pi$ a permutation on $A$, we define $wt(\pi)$, the weight of $\pi$, as the sum $\sum y_{\pi(i),i}$.

We now prove the main theorem of this section.

THEOREM 1 (the $k$-direction splitting lemma). *Let $A$ be a set of $k$ points in $R^{k-1}$. The set of all $k$-directional splitter points of $A$ forms a nonempty convex region. Further, a permutation $\pi$ on $A$ is a directional permutation associated with a splitter point iff $\pi$ is a maximum weight permutation. Moreover, $\pi$ is unique iff there exists a strict-splitter point of $A$.*

*Proof.* First, assume that $\pi$ is a maximum weight permutation. We now show that we can find a splitter point $x$ such that $y_{\pi(i)}$ is in the $i$th directional region relative to $x$. From Proposition 2 we know that such a point $x$ must satisfy the following property: The vector $y_{\pi(i)} - x$ when coordinatized as $\mathcal{T}^{-1}(y_\pi(i) - x)$ must have its $i$th component as the maximum. Hence, choosing representatives $\langle x_1, x_2, \ldots, x_k\rangle \in \mathcal{T}^{-1}(x)$ and $\langle y_{\pi(i),1}, y_{\pi(i),2}, \ldots, y_{\pi(i),k}\rangle \in \mathcal{T}^{-1}(y_{\pi(i)})$, we have that the following inequalities must be satisfied (independent of the representative choices): for each $j \ne i$,

$$y_{\pi(i),i} - x_i \ge y_{\pi(i),j} - x_j.$$

These inequalities are simply a set of difference constraints of the form

$$x_i - x_j \le y_{\pi(i),i} - y_{\pi(i),j}.$$

We know from the theory of difference constraints (see [7]) that such a set of constraints has a solution iff the associated constraint graph $G_\pi$ has no negative cycles. The constraint graph $G_\pi$ is a complete weighted directed graph on $k$ vertices $V(G_\pi) = \{1, 2, \ldots, k\}$ with each edge $(i, j)$ assigned the weight $w(i, j) = y_{\pi(i),i} - y_{\pi(i),j}$. By definition, the weight of a cycle $(i_1, i_2, \ldots, i_p, i_1)$ is given by

$$(y_{\pi(i_1),i_1} - y_{\pi(i_1),i_2}) + (y_{\pi(i_2),i_2} - y_{\pi(i_2),i_3}) + \cdots + (y_{\pi(i_p),i_p} - y_{\pi(i_p),i_1}).$$

Collecting terms, this sum is equal to

$$(y_{\pi(i_1),i_1} + y_{\pi(i_2),i_2} + \cdots + y_{\pi(i_p),i_p}) - (y_{\pi(i_1),i_2} + y_{\pi(i_2),i_3} + \cdots + y_{\pi(i_p),i_1}) \ge 0.$$

It follows that this value is nonnegative, since $\pi$ is a maximum permutation (matching). Hence, we have proved that if $\pi$ is a maximum weight permutation, then we can find a vector $x$ that is a splitter point of $A$ with $\pi$ as the associated directional permutation. Note that a solution vector $x$ can be found by applying an algorithm for shortest paths [7].

For the converse, suppose that $\pi$ is a directional permutation for $A$, and $x$ is the associated splitter point. Let $\sigma \neq \pi$ be some other permutation. We have the following inequalities that show that the weight of $\pi$ is at least as large as the weight of $\sigma$:

$$\begin{aligned} 0 &= \sum x_i - \sum x_i = \sum_i (x_i - x_{(\pi\sigma^{-1})(i)}) \leq \sum y_{\pi(i),i} - y_{\pi(i),(\pi\sigma^{-1})(i)} \\ &= \sum y_{\pi(i),i} - \sum y_{\sigma(i),i}. \end{aligned}$$

Finally, it follows from the argument above that a permutation $\pi$ is a unique maximum weight permutation iff there is a strict-splitter point. This can be deduced by replacing the inequalities in the difference constraints with strict inequalities. $\square$

**4. Applications to graph connectivity and routing.** We now show the connection between the $k$-directional compass as defined in section 2 and the generation of disjoint paths in graphs using only local information. This connection is established by embedding the vertices of the graph to points of a real vector space. We define a directional embedding of a graph as follows.

DEFINITION. *A graph $G = (V, E)$ has a* directional $X$-embedding *if there is a mapping $f$ of the vertices to points in real space $f : V \to R^{k-1}$, where $|X| = k$, and there is a $k$-directional compass $\mathcal{C}_k$, such that for each $v \in V - X$, $f(v)$ is a $k$-directional splitter point of a subset of $f(N(v))$, the embedded neighbor set of $v$. A directional $X$-embedding is* nondegenerate *if, for each $v \in V - X$, $f(v)$ is a strict-splitter point of a subset of $f(N(v))$.*

PROPOSITION 4. *Let $G = (V, E)$ be a graph and let $X \subset V$. If $G$ has a nondegenerate directional $X$-embedding, then $G$ contains $|X| = k$ independent sink trees to $X$. Furthermore, these sink trees are strongly independent.*

*Proof.* As noted earlier, the directional transitivity of the compass implies that paths generated in a consistent direction must be (internally) vertex-disjoint from paths generated consistently in a different direction, when the paths originate from the same vertex. Hence, each of the $|X| = k$ directions can be used to define a distinct parent for each vertex $u \in V - X - \{v\}$. Thus, each of the $k$ directions can be associated with a distinct sink tree, and the collection of $k$ sink trees is independent. It is easily verified that these trees are strongly independent, since we can match any set $Y$ of $k$ points to the $k$ sink trees via the directional splitting permutation $\pi_Y$. $\square$

We remark that by viewing a graph as a symmetric digraph we can strengthen Proposition 4 so as to produce $k$ pairwise independent dags; each direction determines a dag with a single sink vertex $v$. A pair of dags on the same vertex set is independent if, given any vertex $u$, all paths in one dag are internally node-disjoint from all paths in the other dag originating at the same vertex.

A long-standing, open conjecture of Frank (see [13]) states that all $k$-connected graphs have, for each vertex $v$, $k$-independent spanning trees rooted at $v$. Our results lead us to conjecture a stronger result.

CONJECTURE 1. *A graph $G$ is $k$-connected iff for every $X \subset V$, with $|X| = k$, $G$ has a nondegenerate directional $X$-embedding.*

A closely related conjecture builds on the characterization of [12].

Conjecture 2. *A graph $G$ has a convex $X$-embedding in general position iff $G$ has a nondegenerate directional $X$-embedding.*

From the results in [12] it follows that the if-part of Conjecture 2 holds. Further, from *st*-numbering we have that the conjecture is true for $|X| = 2$. We prove in section 5 that both conjectures are true for $|X| = 3$. We remark that Conjectures 1 and 2 are not true for directed graphs, since there are $k$-connected digraphs that do not have $k$-independent sink trees, for each $k \geq 3$ [9]. However, we now show the conjecture is true for dags $k$-connected to a sink set. As a corollary of this result we have that dags $k$-connected to a sink vertex $v$ can be decomposed into $k$-independent dags (independently proven by [1] and [9]).

Theorem 2. *A dag $k$-connected to a set $X$ of $k$ sink vertices has a nondegenerate directional $X$-embedding.*

*Proof.* Embed the sink vertices $X$ anywhere in general position in $R^{k-1}$. It follows from $k$-connectivity that all nonsink vertices have outdegree at least $k$. Now topologically sort the remaining vertices. Each vertex, considered in (reverse) topological order, can be positioned as a splitter point of any size $k$ subset of its out-neighborhood. By employing slight adjustments, each point can be made a strict-splitter. Hence, the theorem follows. □

**5. Directional embeddings of 3-connected graphs.** In this section we show that 3-connected graphs have 3-directional embeddings in the plane. To prove this we need a result that is stronger than Theorem 1 (the splitting theorem). Let us assume we are given a 3-directional compass $\mathcal{C}_3(L)$ in the plane equipped with a fixed set of three rays (and the complementary antirays). We now show that it is still possible to "split" a set of three points in the plane into three directions even with the addition of many new directional constraints of a certain type.

Lemma 1. *Given a pair of points $\{a, c\}$ in the plane, let $p$ be the intersection point of ray-$i$ (of $\mathcal{C}_3(L)$) originating at $a$ with an antiray-$j$ originating at $c$. Then all the points on the line segment $[a, p]$ are in the same region as $a$ with respect to the reference point $c$.*

*Proof.* Clearly, ray-$i$ intersects no other antiray originating at $c$ except for antiray-$j$. If any point on the segment $[a, p]$ lies in a region different from that of $a$ with respect to the reference point $c$, then this segment must cross a boundary of directional regions, i.e., it must cross a different antiray from $c$; this is a contradiction. □

Lemma 2 (the 3-direction constrained splitting lemma). *Let $C$ be any nonempty set of points in the plane. Given a pair of points $\{a, b\}$ in the plane, there is a point $c_0 \in C$ and a point $s$ such that $s$ is a 3-directional splitter point of $\{a, b, c_0\}$ satisfying the following constraints: $d(a, s) = d(a, b)$, $d(b, s) = d(b, a)$, and for each $c \in C$, $d(c, s) = d(c, a)$.*

*Proof.* Without loss of generality, we can assume that $d(a, b) = 1$ and $d(b, a) = 2$. Consider the pair of rays $\{r, r'\}$, where $r$ is ray-1 originating from $a$ and $r'$ is ray-2 originating at $b$. Let $p$ denote their point of intersection. Note that the point $p$ is a 3-directional splitter (nonstrict) for $\{a, b, c\}$ (for any $c \in C$) that satisfies the first two directional constraints given in the statement of the lemma, i.e., $d(a, p) = d(a, b)$, $d(b, p) = d(b, a)$. Hence, if no antiray originating at a point of $C$ intersects the ray $r$, then setting $s = p$ satisfies all the constraints of the lemma. Otherwise, consider all the points of intersection of the ray $r$ with the three antirays originating at points of $C$. Let $p_0$ denote the intersection point (of ray $r$ with the antiray originating from a point $c_0 \in C$) that is nearest $a$. Then by Lemma 1, it follows that $s = p_0$ is a 3-

directional splitter (nonstrict) that satisfies all the constraints and the lemma holds. We remark that by perturbing the points the splitter can be made strict. ∎

Graphs that are 3-connected have the following characterization well known as Tutte's wheel theorem. A *wheel* is a graph consisting of a cycle (of at least three vertices) and a distinct hub vertex that is connected to every vertex on the cycle.

THEOREM 3 (Tutte's wheel theorem [14]). *If a graph $G$ is 3-connected, then either $G$ is a wheel or there is an edge of $G$ that can be deleted or contracted while preserving 3-connectivity. Furthermore, no contractible edge is part of a triangle.* ∎

It follows immediately from Tutte's theorem that every 3-connected graph can, through a series of edge contractions and removals of multiedges, be reduced to $K_3$ while preserving 3-connectivity at each intermediate step. We now extend this result to show that these contractions can be chosen to avoid the edges of any given triangle $S \subset G$.

THEOREM 4 (the reduction theorem). *Let $G = (V, E)$ be a 3-connected graph that contains a triangle $S$ with vertices $\{s_1, s_2, s_3\}$. The graph $G$ can be reduced, via edge contraction and removal of multiedges, to the triangle $S$, while preserving 3-connectivity at each intermediate stage, and without contracting an edge of $S$.*

*Proof.* The proof will follow by induction on the number of edge contractions.

First, suppose that a vertex $s_i$ of the triangle $S$ has only a single neighbor $x$ outside $S$. The edge $e = s_i x$ can be contracted without affecting 3-connectivity. The 3-connectivity is preserved since the three disjoint paths that all vertices in $V - S$ have to $S$ before the contraction of $e$ remain unaffected after the contraction.

Hence, we can assume all vertices of the triangle $S$ have at least two neighbors in the set $V - S$. Starting with the graph $G$, consider the sequence of Tutte contractions $\{e_1, e_2, \ldots\}$, and let $e_k$ denote the first edge in the sequence that is not part of the triangle $S$.

If the graph $G_{e_k}$ obtained by contracting the edge $e_k$ in the graph $G$ is 3-connected, then the theorem follows by induction.

Suppose, on the other hand, that $G_{e_k}$ has a 2-vertex cut $C$. The 2-cut $C$ must contain a vertex of $S$, since otherwise $C$ would be a 2-cut in the graph $G' = G_{\{e_1, e_2, \ldots, e_k\}}$ obtained by contracting the sequence of edges $\{e_1, e_2, \ldots, e_k\}$; however, this graph $G'$ is 3-connected by Tutte's theorem. The 2-cut $C$ must also contain a vertex of $S$, since otherwise $C$ would be a 2-cut in the graph $G$, which is assumed to be 3-connected. Hence, the 2-cut $C$ in the graph $G_{e_k}$ contains precisely one vertex of $S$, say, $s_3$, and one vertex of $V - S$, say, $x$. No connected component of $G_{e_k}(V - C)$ can contain vertices in both $S$ and $V - S$, since otherwise this would imply that the graph $G'$ has a 2-cut. It follows that the remaining vertices $\{s_1, s_2\}$ in $S - C$ must have the vertex $x$ as their only neighbor in the graph $G_{e_k}$. Hence, the (original, uncontracted) graph $G$ contains two vertices $x_1, x_2$ in $V - S$ that are neighbors with each of the two vertices $s_1, s_2$ of $S - C$, forming a $K_{2,2}$ subgraph.

We claim that we can contract any edge $e'$ of this $K_{2,2}$ subgraph of $G$ without affecting 3-connectivity. This claim follows from the fact that the three disjoint paths to $S$ that each vertex possesses in the graph $G$ can be chosen so that one path uses edge $s_1 x_1$ and another uses $s_2 x_2$. Hence, contracting the edge $s_1 x_1$ (or edge $s_2 x_2$) will not change the number of disjoint paths to $S$, i.e., the graph $G_{e'}$ is 3-connected. The proof thus follows by induction. ∎

COROLLARY 1. *Let $G = (V, E)$ be a graph 3-connected to three vertices $S = \{s_1, s_2, s_3\} \subset V$. The graph $G$ can be reduced, via edge contraction and removal of multiedges, to the subgraph induced by $S$, while preserving 3-connectivity to $S$ at each* *intermediate stage, and without contracting an induced edge of $S$.*

*Proof.* The proof is an immediate consequence of Theorem 4, since if we complete $S$ to a triangle, the graph becomes 3-connected. $\square$

We apply the reduction theorem to show that we can inductively obtain a 3-directional embedding when a graph is 3-connected to a set of vertices $S$.

THEOREM 5. *Let $G$ be a graph, and suppose $S$ is any set of three vertices. Then $G$ is 3-connected to $S$ iff $G$ has a nondegenerate directional $S$-embedding.*

*Proof.* The proof of the if-part is immediate from Proposition 4. The proof of the converse follows by induction on the number of reduction operations defined by Corollary 1 above.

For the inductive hypothesis we assume we have a nondegenerate directional $S$-embedding of a reduced graph $G'$. For the base of the induction embed the three vertices of $S$ anywhere in general position in $R^2$. Clearly, for an edge addition operation the induction is trivially extended. Edge expansions involving a single vertex of $S$ are also trivial. We need only consider the case of an edge expansion not involving $S$ (i.e., a vertex 3-split operation).

We may assume the vertex $a \notin S$ involved in the edge expansion is a 3-directional splitter of its neighbors represented by $B \cup C$, where $|B| \geq 2 \leq |C|$. Further, we can assume w.l.o.g. that $a$ is a 2-directional splitter of $B$, i.e., there are two points in $B$ that lie in two distinct directions relative to the reference point $a$. The proof of the theorem follows from the following claim.

CLAIM. *There exists in the plane a pair of points $s_b, s_c$ such that $s_b$ is a 3-directional splitter of $B \cup \{s_c\}$, and $s_c$ is a 3-directional splitter of $C \cup \{s_b\}$. Furthermore, all previously defined directional constraints towards $a$ are preserved by the new points, i.e., for each $b \in B$, $d(b,a) = d(b,s_b)$ and for each $c \in C$, $d(c,a) = d(c,s_c)$.*

*Proof of claim.* Set $s_b = a$. A solution for $s_c$ can be found by applying Lemma 2 (the 3-direction constrained splitting lemma), where the point $s_c$ plays the role of $s$ in the statement of Lemma 2. The satisfaction of the first constraint of Lemma 2 ensures that $s_b$ (playing the role of $a$ in the lemma) is a 3-directional splitter of $B \cup \{s_c\}$. The claim follows by the satisfaction of the remaining constraints of Lemma 2. $\square$

**6. Conclusion.** In this paper we proposed a mathematical model for network routing based on generating paths in a consistent direction. Our directional model was developed out of an algebraic and geometric framework. We gave a natural model for defining directions in real spaces and showed that such directions possess a fundamental matching property of combinatorial interest. We defined a generalization of $st$-numberings that was based on embedding the vertices in multidimensional space and applying the defined directions for disjoint path generation. Directional embeddings are motivated by the fact that they encode the disjoint path structure and induce strongly independent sink trees with simple local information. We showed that a dag that is $k$-connected to a set of sinks has a $k$-directional embedding in $(k-1)$-space with the sink set as the extreme vertices. Finally, we proved that all 3-connected graphs have 3-directional embeddings in the plane. The problem of whether directional embedding exist for $k$-connected graphs, for $k > 3$, remains open.

## REFERENCES

[1] F.S. ANNEXSTEIN, K.A. BERMAN, AND R. SWAMINATHAN, *A multi-tree generating routing scheme using acyclic orientations*, in Computing and Combinatorics, Proceedings of the Third Annual International Conference, Lecture Notes in Comput. Sci. 1279, Springer-Verlag, New York, 1997, pp. 18–22.

[2] F. Bao, Y. Igarashi, and S.R. Ohring, *Reliable broadcasting in product networks*, Discrete Appl. Math., 83 (1998), pp. 3–20.

[3] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, New York, 1992.

[4] J. Cheriyan and S.N. Maheshwari, *Finding non-separating induced cycles and independent spanning trees in* 3*-connected graphs*, J. Algorithms, 9 (1988), pp. 507–537.

[5] J. Cheriyan and J. Reif, *Directed s-t numberings, rubber bands, and testing digraph k-vertex connectivity*, Combinatorica, 14 (1994), pp. 435–451.

[6] R. Cohen, B.V. Patel, F. Schaffa, and M. Willebeek-LeMair, *The sink tree paradigm: Connectionless traffic support on ATM LAN's*, IEEE/ACM Trans. on Networking, 4 (1996), pp. 363–374.

[7] T. Corman, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, McGraw-Hill, New York, 1994.

[8] S. Even and R.E. Tarjan, *Computing an st-numbering*, Theoret. Comput. Sci., 2 (1976), pp. 339–344.

[9] A. Huck, *Disproof of a conjecture about independent branchings in k-connected directed graphs*, J. Graph Theory, 20 (1995), pp. 235–239.

[10] A. Itai and M. Rodeh, *The multi-tree approach to reliability in distributed networks*, Inform. and Comput., 79 (1988), pp. 43–59.

[11] A. Lempel, S. Even, and I. Cederbaum, *An algorithm for planarity testing graphs*, in Theory of Graphs, P. Rosentiehl, ed., Gordon and Breach, New York, 1966, pp. 215–232.

[12] N. Linial, L. Lovász, and A. Wigderson, *Rubber bands, convex embeddings, and graph connectivity*, Combinatorica, 8 (1988), pp. 91–102.

[13] A. Schrijver, *Fractional packing and covering*, in Packing and Covering in Combinatorics, A. Schrijver, ed., Mathematical Centre Tracts 106, Mathematisch Centrum, Amsterdam, 1979, pp. 201–274.

[14] W.T. Tutte, *A theory of* 3*-connected graphs*, Indag. Math. (N.S.), 23 (1961), pp. 441–455.

[15] A. Zehavi and A. Itai, *Three tree-paths*, J. Graph Theory, 13 (1989), pp. 175–188.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 3, pp. 281–294

# TWO-DIMENSIONAL GANTT CHARTS AND A SCHEDULING ALGORITHM OF LAWLER*

MICHEL X. GOEMANS† AND DAVID P. WILLIAMSON‡

**Abstract.** In this note we give an alternate proof that a scheduling algorithm of Lawler [E.L. Lawler, *Ann. Discrete Math.*, 2 (1978), pp. 75–90, E.L. Lawler and J.K. Lenstra, in *Ordered Sets*, I. Rival, ed., D. Reidel, 1982, pp. 655–675] finds the optimal solution for the scheduling problem $1|prec|\sum_j w_jC_j$ when the precedence constraints are series-parallel. We do this by using a linear programming formulation of $1|prec|\sum_j w_jC_j$ introduced by Queyranne and Wang [*Math. Oper. Res.*, 16 (1991), pp. 1–20]. Queyranne and Wang proved that their formulation completely describes the scheduling polyhedron in the case of series-parallel constraints; a by-product of our proof of correctness of Lawler's algorithm is an alternate proof of this fact. In the course of our proof it is helpful to use what might be called *two-dimensional* (2D) *Gantt charts*. We think these may find independent use, and to illustrate this we show that some recent work in the area becomes transparent using 2D Gantt charts.

**Key words.** scheduling, series-parallel, linear programming

**AMS subject classifications.** 90B35, 90C27, 68Q25, 90C05

**PII.** S0895480197330254

## 1. Introduction.

We consider the problem of scheduling $n$ jobs on a single machine. Each job $j$ must be scheduled for $p_j$ units of time, and only one job can be scheduled at any point in time. We only consider *nonpreemptive* schedules, in which all $p_j$ units of job $j$ must be scheduled consecutively. Furthermore, the schedule must obey specified *precedence constraints*. Precedence constraints are given by a partial order on the jobs; if $i$ precedes $j$ in the partial order (denoted $i \rightarrow j$), then $i$ must be completely processed before $j$ can begin its processing. A (possibly negative) weight $w_j$ is associated with job $j$, and our goal is to find a schedule which minimizes the sum $\sum_j w_jC_j$, where $C_j$ is the time at which job $j$ completes in the given schedule (the *completion time* of $j$). This scheduling problem is denoted $1|prec|\sum_j w_jC_j$ in the notation of Lawler et al. [6].

The general problem $1|prec|\sum_j w_jC_j$ was shown to be NP-complete by Lenstra and Rinnooy Kan [7] and Lawler [4]. Nevertheless, special cases are known to be polynomial-time solvable. In 1978, Lawler [4] gave an $O(n \log n)$ time algorithm for solving $1|prec|\sum_j w_jC_j$ when the given precedence constraints are *series-parallel*. Series-parallel precedence constraints can be defined inductively; the base elements are individual jobs. Given series-parallel constraints on sets of jobs $S_1$ and $S_2$, $S_1 \cap S_2 = \emptyset$, the *parallel composition* of $S_1$ and $S_2$ gives a partial order on $S_1 \cup S_2$ that maintains the orders on $S_1$ and $S_2$, and if $i \in S_1$ and $j \in S_2$, then $i$ and $j$ are unordered. The *series composition* of $S_1$ and $S_2$ gives a partial order on $S_1 \cup S_2$ that maintains the

*Received by the editors November 19, 1997; accepted for publication (in revised form) November 18, 1999; published electronically May 15, 2000.
http://www.siam.org/journals/sidma/13-3/33025.html

†Department of Mathematics, MIT, Room 2-351, 77 Massachusetts Avenue, Cambridge, MA 02139 (goemans@math.mit.edu). Part of this work was done while the author was at CORE, Louvain-La-Neuve, Belgium. Research supported in part by NSF contract 9623859-CCR, ARPA Contract N00014-95-1-1246, and a Sloan fellowship.

‡IBM T.J. Watson Research Center, Room 33-219, P.O. Box 218, Yorktown Heights, NY 10598 (dpw@watson.ibm.com).

orders on $S_1$ and $S_2$, and if $i \in S_1$ and $j \in S_2$, then $i$ precedes $j$ in the partial order. Often series-parallel constraints are given in terms of a binary *structure tree*, with the leaves denoting jobs and the internal nodes denoting either a parallel or series composition of the two corresponding subtrees.

In this note we give an alternate proof of the correctness of Lawler's algorithm. We do this by using a linear programming (LP) formulation of $1|prec|\sum_j w_j C_j$ introduced by Queyranne and Wang [11]. Queyranne and Wang proved that their formulation completely describes the scheduling polyhedron in the case of series-parallel constraints; a by-product of our proof of correctness of Lawler's algorithm is an alternate proof of this fact. Other proofs of the correctness of Lawler's algorithm have been given (see [4, 10, 9], for example), but to the best of our knowledge, ours is the first duality-based proof.

In the course of our proof it is helpful to use what might be called *two-dimensional* (2D) *Gantt charts*. Although 2D Gantt charts were introduced in a 1964 paper by Eastman, Even, and Isaacs [2], they seem to have become buried in the literature. We use these 2D Gantt charts to obtain geometric intuition into the dual of Queyranne and Wang's linear programming formulation. We think 2D Gantt charts may find further uses, and to illustrate this we show that a recent observation of Hall and Chudak [3] and a recent theorem of Margot, Queyranne, and Wang [8] (discovered independently by Chekuri and Motwani [1]) are transparent using 2D Gantt charts.

The remainder of the note is structured as follows. In section 2 we introduce 2D Gantt charts and illustrate their usefulness. In section 3 we review the LP formulation for $1|prec|\sum_j w_j C_j$. Finally, in section 4 we turn to the proof of correctness of Lawler's algorithm.

## 2. 2D Gantt charts.

We first need the following notation. Let $N = \{1, \ldots, n\}$ denote the set of all jobs. For any set $S$ of jobs, let $w(S) = \sum_{j \in S} w_j$ and $p(S) = \sum_{j \in S} p_j$.

A traditional Gantt chart has a single dimension of processing time, in which jobs are represented as blocks of length $p_j$ (see Figure 1).

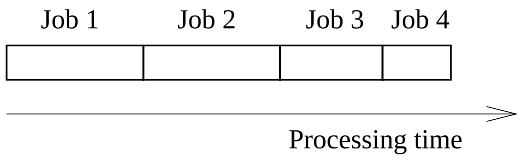


FIG. 1. *A Gantt chart.*

In a 2D Gantt chart, we introduce a second axis of weight. The chart starts at the point $(0, w(N))$ and ends at $(p(N), 0)$. Each job, say $j$, is represented by a rectangle of length $p_j$ and height $w_j$ whose position is defined by a startpoint and an endpoint. The startpoint $(t, w)$ of a job is the endpoint of the previous job (or $(0, w(N))$ for the first job) while its endpoint is $(t + p_j, w - w_j)$. See Figure 2 for an illustration in which job 3 has a negative weight. The total amount of weight that has not been completed yet at any point in time—*the uncompleted work*—can be easily read off from the 2D Gantt chart: this corresponds to the bold line in the figure, and it is composed of the upper side of each rectangle for the jobs of positive weight and of the lower side of each rectangle for the jobs of nonpositive weight. The value $\sum_j w_j C_j$ of a schedule can then be easily inferred from the 2D Gantt chart; it is simply the area below the uncompleted work line, as shaded in Figure 3. This is true even if there are

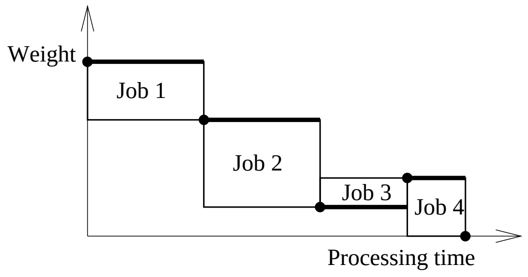


FIG. 2. *A* 2D *Gantt chart. Job* 3 *has a negative weight.*

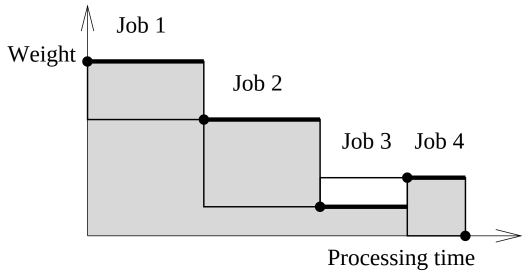


FIG. 3. *The area of a schedule.*

negative weight jobs since $w_1p_1 + w_2(p_1 + p_2) + \cdots + w_n(p_1 + p_2 + \cdots + p_n)$ can also be written as $p_1(w_1 + w_2 + \cdots + w_n) + p_2(w_2 + w_3 + \cdots + w_n) + \cdots + p_nw_n$. We will refer to this shaded area as the *area* of the schedule. Thus minimizing $\sum_j w_jC_j$ over nonpreemptive schedules is equivalent to arranging the jobs as shown in the 2D Gantt chart so as to minimize its area. If we have negative weights and the chart of a given schedule goes below the $x$ axis, then this means that, by postponing the processing of the jobs following that point, we can decrease the weighted sum of completion times arbitrarily and the problem is unbounded. We will see that this is the only possible situation to make the objective function unbounded; in particular, if the problem is bounded, then there will not be any idle time in an optimum schedule. As a result, we will only consider schedules without idle time.

For convenience, we will often draw a line from the startpoint to the endpoint of the rectangle representing a job. We will denote minus the slope of this line segment as $\rho(j) = w_j/p_j$, and we will sometimes refer to this line segment as the *slope* of job $j$. See Figure 4. We will let $\rho(S)$ denote the slope of a set $S$ of jobs, so that $\rho(S) = w(S)/p(S)$, where $w(S) = \sum_{j\in S} w_j$ and $p(S) = \sum_{j\in S} p_j$. The dotted line in the figure represents $\rho(S)$ for $S = \{1, 2, 3\}$. Observe that the area of a schedule is equal to the area below its slopes plus $\sum_{j\in N} \frac{1}{2}w_jp_j$. The area below the slopes thus represents $\sum_{j\in N} w_j(C_j - \frac{1}{2}p_j) = \sum_{j\in N} w_jM_j$, where $M_j$ denotes the mean busy time of job $j$, that is, the midpoint between its start and its completion.

To show the usefulness of this concept, we illustrate two recently discovered facts about $1|prec|\sum_j w_jC_j$ using 2D Gantt charts. First, Hall and Chudak [3] made the following observation (see Von Arnim, Faigle, and Schrader [14] for the case $p_j = 1$). Given an instance $(p, w, \rightarrow)$ of $1|prec|\sum_j w_jC_j$, where $\rightarrow$ is the precedence relation, if one creates a new instance $(p', w', \rightarrow')$ by setting $p'_j = w_j$ and $w'_j = p_j$ for all $j$, and $j \rightarrow' i$ iff $i \rightarrow j$, then an optimal schedule for $(p', w', \rightarrow')$ is in the opposite order of

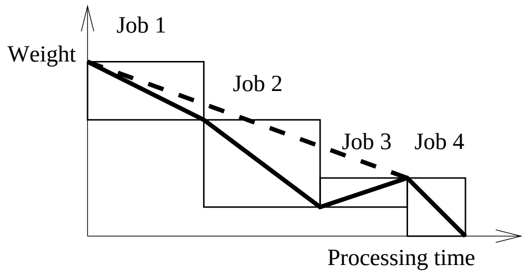


FIG. 4. *The slope of jobs and of a set of jobs.*

an optimal schedule for $(p, w, \rightarrow)$ and has the same value. This follows quite simply by displaying any solution for $(p, w, \rightarrow)$ on a 2D Gantt chart and flipping the axes to obtain a solution of the same value for $(p', w', \rightarrow')$. See Figure 5 for an illustration.

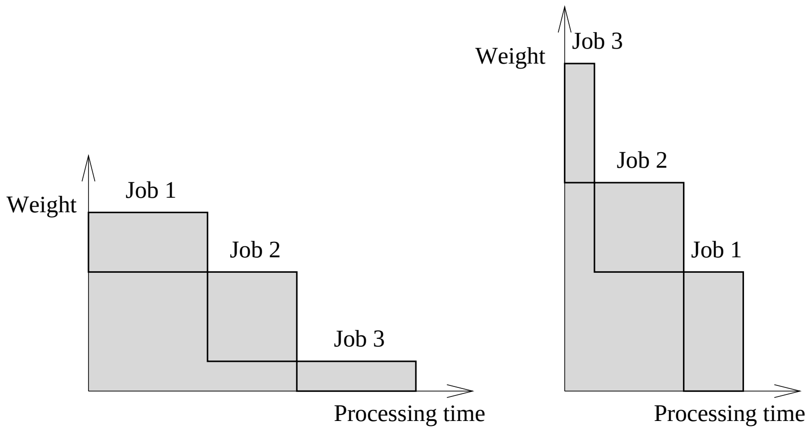


FIG. 5. *Illustration of the Hall–Chudak observation.*

Also, Margot, Queyranne, and Wang [8] recently showed the following (also discovered independently by Chekuri and Motwani [1]): suppose we have an instance of $1|prec|\sum_j w_j C_j$ such that all weights are nonnegative and for any *initial* set of jobs $S$ (i.e., there exists a valid schedule in which all the jobs in $S$ are scheduled before all jobs in $N - S$), $\rho(S) \le \rho(N)$. Then any valid schedule comes within a factor of 2 of optimal. Given the condition $\rho(S) \le \rho(N)$ for any initial set $S$, using a 2D Gantt chart it is easy to see that the slopes of the jobs will always remain above the slope of the entire set of jobs. Thus for any schedule $\sum_j w_j C_j \ge \frac{1}{2} w(N)p(N)$, since its area will always be at least $\frac{1}{2} w(N)p(N)$. See Figure 6 for an illustration. But certainly $\sum_j w_j C_j \le w(N)p(N)$, so that any schedule is no more than twice the optimal value.

This observation, together with a result of Sidney [12], leads to a 2-approximation algorithm for general instances of $1|prec|\sum_j w_j C_j$ as observed by Chekuri and Motwani [1]. Sidney [12] shows that if we let $S$ be an initial set of jobs of maximum $\rho$ value, then there exists an optimum solution to $1|prec|\sum_j w_j C_j$ for which the jobs in $S$ are processed before any job in $N \setminus S$. This result can be shown using 2D Gantt charts, and this is left as a (nontrivial) exercise to the reader. Chekuri and Motwani [1] propose to find such a set $S$ (which can be done using parametric maximum flow techniques), process these jobs first in any valid ordering, and repeat the procedure

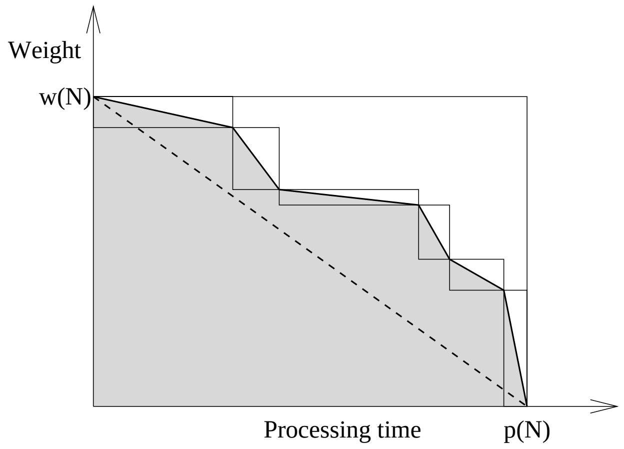


FIG. 6. *Illustration of the Margot, Queyranne, and Wang result.*

with the remaining jobs in $N \setminus S$. Their previous observation together with the result of Sidney guarantees that the solution they construct is within a factor of 2 of the optimum.

**3. A linear program for $\mathbf{1}|\boldsymbol{prec}|\sum_j \boldsymbol{w_j C_j}$ and some preliminaries.** We now turn to proving that a scheduling algorithm of Lawler for $1|prec|\sum_j w_j C_j$ is optimal when the precedence constraints are series-parallel. The proof uses the following linear programming formulation of the problem

$$\text{Min} \quad \sum_j w_j (M_j + p_j/2)$$

subject to

$$\sum_{j \in S} p_j M_j \geq \frac{1}{2} p(S)^2, \quad S \subseteq N, \tag{1}$$

$$\frac{1}{p(B)} \sum_{j \in B} p_j M_j - \frac{1}{p(A)} \sum_{j \in A} p_j M_j \geq \frac{1}{2}(p(A) + p(B)), \tag{2}$$
$$A \subseteq N,\ B \subseteq N - A,\ A \to B,$$

where $A \to B$ means that the precedence constraints enforce that each job in $A$ must be scheduled before each job in $B$. In this formulation, $M_j$ represents the mean busy time of job $j$. By adopting the convention that $\emptyset \to S$ for any $S$, we could simply get rid of constraint (1). Queyranne and Wang [11] have shown that the completion time reformulations of constraints (1) and (2) completely describe the scheduling polyhedron when the precedence constraints are series-parallel (also shown by Von Arnim, Faigle, and Schrader [14] in the case $p_j = 1$ for all jobs). A by-product of our proof of correctness of Lawler's algorithm is an alternate proof of this fact.

To see that constraint (1) is valid, choose any $S \subseteq N$; suppose $S = \{1, \dots, k\}$. If the jobs $1, \dots, k$ are the first jobs scheduled, then simple algebra shows that $\sum_{j \in S} p_j M_j = \frac{1}{2} p(S)^2$ (notice that this does not depend on the ordering of the jobs). In any valid schedule the sum is at least as large, and hence inequality (1) is valid. More generally, if the jobs in $S$ are continuously scheduled between $s$ and $t = s + p(S)$ then $\sum_{j \in S} p_j M_j = \left(s + \frac{1}{2} p(S)\right) p(S) = \left(t - \frac{1}{2} p(S)\right) p(S)$. Hence in any valid schedule in which the jobs in $S$ are scheduled (not necessarily continuously) between $s$ and $t$, we have

$$\left(s + \frac{1}{2}p(S)\right) p(S) \le \sum_{j \in S} p_j M_j \le \left(t - \frac{1}{2}p(S)\right) p(S). \tag{3}$$

To see that constraint (2) is valid, choose any $A$ and $B$ obeying the conditions. From (3), we derive that $\frac{1}{p(B)} \sum_{j \in B} p_j M_j \ge \tau + \frac{1}{2}p(B)$ and $\frac{1}{p(A)} \sum_{j \in A} p_j M_j \le \tau - \frac{1}{2}p(A)$, where $\tau$ is any time between the completion of $A$ and $B$. Subtracting these two inequalities gives (2). Note that an inequality of the form (2) is tight exactly when all jobs in $A$ are processed from time $\tau - p(A)$ to $\tau$ (for some $\tau$), followed by jobs in $B$ processed from time $\tau$ to time $\tau + p(B)$.

To prove the optimality of Lawler's algorithm, we will construct a feasible solution to the dual of the linear program above:

$$\text{Max} \quad \frac{1}{2}\sum_S p(S)^2 y_S + \frac{1}{2}\sum_{A,B}(p(A) + p(B)) z_{A,B} + \frac{1}{2}\sum_j w_j p_j$$

subject to

$$\begin{aligned} &\sum_{S: j \in S} y_S + \sum_{A,B: j \in B} \frac{1}{p(B)} z_{A,B} - \sum_{A,B: j \in A} \frac{1}{p(A)} z_{A,B} = \rho(j) \quad \forall j, \\ &y_S \ge 0 \quad \forall S \subseteq N, \\ &z_{A,B} \ge 0 \quad \forall A \subseteq N, B \subseteq N - A, A \to B. \end{aligned}$$

We will show that our dual solution obeys the complementary slackness conditions with respect to the $M_j$ constructed by Lawler's algorithm. Assuming that all $p_j > 0$, notice that if we have a solution $M$ and $(y, z)$ that obey the complementary slackness relations, then by the observations above, if $y_S > 0$, then the jobs in $S$ must all appear at the beginning of the schedule, and if $z_{A,B} > 0$, then the jobs in $A$ and $B$ are all scheduled together, with the jobs in $A$ appearing immediately before those in $B$.

As a warm-up exercise, suppose that $\rho(1) \ge \rho(2) \ge \cdots \ge \rho(n) \ge 0$, and the schedule $1, \ldots, n$ is compatible with the precedence constraints. Notice that we require that the minimum slope $\rho(n)$ be nonnegative; otherwise the problem is unbounded since we can postpone the processing of this last job arbitrarily. Then the dual solution

$$\begin{aligned} y_{\{1\}} &= \rho(1) - \rho(2) \\ y_{\{1,2\}} &= \rho(2) - \rho(3) \\ &\vdots \\ y_{\{1,\ldots,n-1\}} &= \rho(n-1) - \rho(n) \\ y_{\{1,\ldots,n\}} &= \rho(n) \end{aligned}$$

and all other variables set to zero are feasible and obey the complementary slackness conditions by the discussion above. Hence the schedule $1, \ldots, n$ is optimal. Notice that this gives an alternate proof of Smith's rule [13], which states that scheduling jobs in order of nonincreasing $\rho$ value gives the optimal schedule for problem instances without precedence constraints.

We now consider this simple case from the perspective of 2D Gantt charts. We observe that in this case, the diagonals of the jobs in a 2D Gantt chart representation

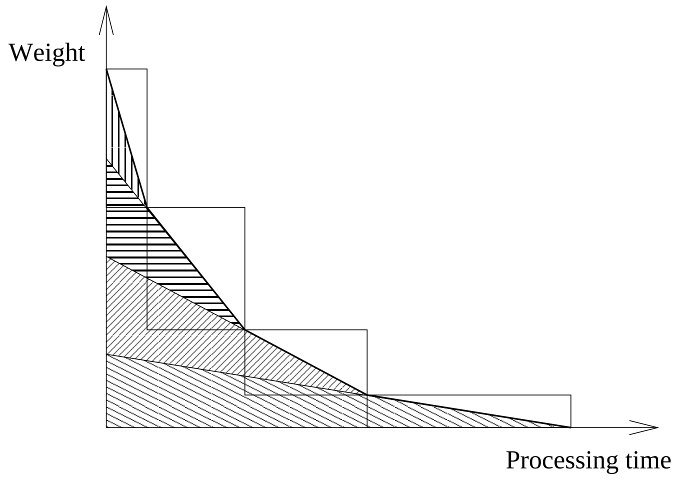


FIG. 7. *Illustration of Smith's rule. Shaded areas indicate the $n$ triangles that account for the area of the schedule (after $\frac{1}{2}\sum_j w_j p_j$ is included).*

of the sequence $1, 2, \ldots, n$ form a piecewise-linear convex function (see Figure 7). Note also that the area of the schedule can be expressed as $\sum_{j\in N} \frac{1}{2} w_j p_j$ plus the area below the slopes of the jobs. This area can be decomposed into $n$ triangles, shown as shaded regions in Figure 7. These triangles are formed by extending the slope of each job to the $y$ axis. We associate with job $j$ the triangle formed between the line from the slope of job $j$ and that of the slope of job $j + 1$ (where we use the $y$ axis of slope $\rho(n + 1) = 0$ for job $n + 1$). Then the "base" of the $j$th triangle on the $y$ axis is $p(\{1, \ldots, j\})(\rho(j) - \rho(j + 1))$, and its height is $p(\{1, \ldots, j\})$, so that its area is exactly $\frac{1}{2}p(\{1, \ldots, j\})^2(\rho(j) - \rho(j + 1)) = \frac{1}{2}p(\{1, \ldots, j\})^2 y_{\{1,\ldots,j\}}$, using the dual solution of the preceding paragraph. Then the total area of the schedule is $\frac{1}{2}\sum_S p(S)^2 y_S + \frac{1}{2}\sum_j w_j p_j$, or exactly the value of the dual objective function.

## 4. Lawler's algorithm.

We now turn to Lawler's algorithm. Lawler's algorithm works bottom-up on the series-parallel structure tree of the precedence constraints. We give the algorithm below, but first we will try to give some intuition on how the algorithm and our dual construction works. One perspective is that as much as possible it tries to follow Smith's rule and makes sure that the jobs in the current subtree can be maintained in order of nonincreasing $\rho$, so that the dual solution given above proves the optimality of the schedule. This may not always be possible, so in some cases two or more jobs are replaced by a *composite* job. The composite jobs may be composed themselves of composite jobs. The weight of a composite job of a set $S$ of jobs is $w(S)$, and its processing time is $p(S)$, so that $\rho(S) = w(S)/p(S)$. A sequence of the jobs forming the composite job is given, so that the effect of scheduling the $p(S)$ time units of the composite job is that the jobs in $S$ are scheduled together in the given sequence.

To be more specific, suppose we have a parallel composition of two sets of jobs $S_1$ and $S_2$, such that the $\rho$ values of the jobs in $S_i$ can be scheduled in nonincreasing order. Then the jobs in $S = S_1 \cup S_2$ can be scheduled in nonincreasing $\rho$ order. (We should stress that, when scheduling in nonincreasing order of $\rho$ values, the slopes of the jobs form a convex function which starts to increase once we start processing the

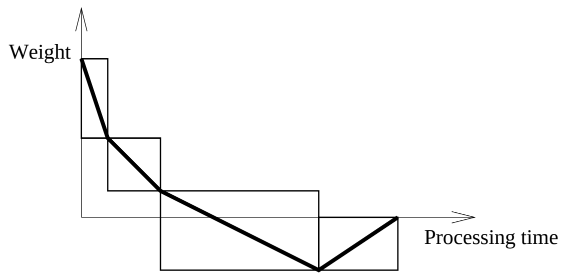


FIG. 8. *Slopes in each $S_i$ are nonincreasing before performing a series or parallel composition.*

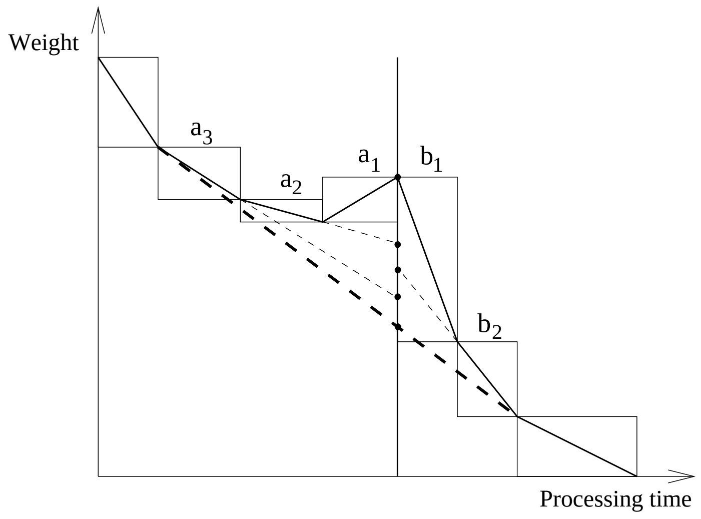


FIG. 9. *An illustration of a composite job $c$ resulting from a series composition of two sets of jobs. The slope $\rho(c)$ is shown as the bold dashed line.*

jobs with negative $\rho$ value; see Figure 8 for an illustration.) However, if we have a series composition of two sets of jobs $S_1$ and $S_2$, then it is possible that some job in $S_2$ has $\rho$ value greater than some job in $S_1$. In this case, we create a composite job $c$ formed of a certain number of jobs of $S_1$ with lowest $\rho$ values and a certain number of jobs of $S_2$ with greatest $\rho$ value. The crucial property to achieve is that the slope of the composite job $c$ is no greater than the $\rho$ value of any remaining job in $S_1$ and no less than the $\rho$ value of any remaining job in $S_2$. The scheduling of job $c$ is then understood to mean that the jobs taken from $S_1$ are scheduled in order of nonincreasing $\rho$, followed immediately by the jobs taken from $S_2$ in order of nonincreasing $\rho$.

There is a unique way of performing this aggregation of jobs, as one can easily see from the 2D Gantt chart. Consider, for example, the situation in Figure 9. In this case, there is the series composition of four jobs in $S_1$ (the four jobs to the left of the vertical line, which we will call the *dividing line*) and three jobs in $S_2$. Note that both $S_1$ and $S_2$ can be scheduled according to Smith's rule, so that the slopes of the jobs in each form a piecewise-linear convex function. A composite job $c$ is formed from $a_3$, $a_2$, $a_1$, $b_1$, and $b_2$. In general, the composite job is formed by the jobs whose slopes do not participate in the lower envelope of the slopes in $S_1$ and in $S_2$.

In Lawler's algorithm, for a series composition of $S_1$ and $S_2$, the composite job was obtained by repeatedly combining two jobs at a time. More precisely, let $j_1$ be the job in $S_1$ that minimizes $\rho(j)$, and let $j_2$ be the job in $S_2$ that maximizes $\rho(j)$. If $\rho(j_1) \geq \rho(j_2)$, then $S = S_1 \cup S_2$ can be scheduled in nonincreasing $\rho$ order, and we stop. Otherwise, we remove $j_1$ from $S_1$ and $j_2$ from $S_2$ and form a composite job $c = (j_1, j_2)$. As long as there is a job $j$ in $S_1$ whose $\rho$ value is lower than $\rho(c)$ or a job $j$ in $S_2$ whose $\rho$ value is greater than $\rho(c)$, we remove $j$ from $S_1$ or $S_2$ and add it to the composite job $c$. When this terminates, $\rho(j) \geq \rho(c)$ for all jobs remaining in $S_1$ (if any), and $\rho(c) \geq \rho(j)$ for any job $j$ remaining in $S_2$ (if any).

When performing a series composition of $S_1$ and $S_2$ which concatenates jobs $a_k, a_{k-1}, \ldots, a_1 \in S_1$ (where $\rho(a_i) \geq \rho(a_{i-1})$) and jobs $b_1, b_2, \ldots, b_\ell \in S_2$ (where $\rho(b_i) \geq \rho(b_{i+1})$) into a single job $c$, we define the $z$ variables in such a way that the $\rho$ value of $a_i$ effectively increases to $\rho(c)$ and the $\rho$ value of $b_j$ effectively decreases to $\rho(c)$. More formally, we construct dual variables $z_{A,B} \geq 0$ such that $z_{A,B} > 0$ only if $A \subseteq S_1$, $B \subseteq S_2$, the jobs in $A$ appear immediately before $B$ in the schedule, and such that for all $a_i$,

$$\rho(c) = \rho(a_i) + \sum_{A,B: a_i \in A} \frac{1}{p(A)} z_{A,B},$$

and for all jobs $b_j$,

$$\rho(c) = \rho(b_j) - \sum_{A,B: b_j \in B} \frac{1}{p(B)} z_{A,B}.$$

All the jobs composing $c$ now look identical in terms of $\rho$ values. Later on, if we set a variable $z_{A,B}$, or even $y_S$, involving the composite job $c$, we simply need to replace $c$ by the set of jobs (or by the set of composite jobs and proceed recursively) to get the appropriate dual solution. In particular, if an $a_i$ or $b_j$ is itself a composite job formed in a previous series composition, applying the argument recursively ensures that all the (original) jobs that compose $c$ look identical in terms of $\rho$ values. In the process of defining $z_{A,B}$, we need to ensure that the contribution of these newly defined variables to the dual objective function corresponds exactly to the area between the slope of the composite job $c$ and the slopes of the jobs composing $c$ (the area shaded on Figure 12); this is the area of the schedule which is lost by replacing the jobs composing $c$ into a single composite job. If we can show that these newly defined variables account for this lost area, then by induction it suffices to show that, at the end, after processing the root of the series-parallel structure tree, we can account for the area below the resulting schedule, and this will follow from our discussion of Smith's rule from section 3.

The $z$ variables are defined as follows. Let $A_i = \{a_1, \ldots, a_i\}$ for $i \leq k$ and $B_j = \{b_1, \ldots, b_j\}$ for $j \leq \ell$. We set $z = 0$ and give the procedure shown in Figure 10 for computing $z_{i,j} \equiv z_{A_i,B_j}$, where we use the convention that $A_{k+1} \equiv A_k$, $B_{\ell+1} \equiv B_\ell$, $\rho(a_{k+1}) \equiv \rho(b_{\ell+1}) \equiv \rho(A_k \cup B_\ell)$. As stated above, we need to show that $z \geq 0$, that for job $a_i$

$$\rho(A_k \cup B_\ell) - \sum_{r,s: a_i \in A_r} \frac{1}{p(A_r)} z_{r,s} = \rho(a_i),$$

```
1.    for $i \leftarrow 1$ to $k$
2.      $\alpha_i \leftarrow \rho(a_{i+1})p(A_{i+1}) - w(A_{i+1})$
3.    for $j \leftarrow 1$ to $\ell$
4.      $\beta_j \leftarrow w(B_{j+1}) - \rho(b_{j+1})p(B_{j+1})$
5.    $i \leftarrow 1$
6.    $j \leftarrow 1$
7.    $x \leftarrow 0$
8.    while $i \leq k$ and $j \leq \ell$
9.      if $\alpha_i < \beta_j$
10.       $z_{i,j} \leftarrow \alpha_i - x$
11.       $x \leftarrow \alpha_i$
12.       $i \leftarrow i + 1$
13.     else
14.       $z_{i,j} \leftarrow \beta_j - x$
15.       $x \leftarrow \beta_j$
16.       $j \leftarrow j + 1$
```

FIG. 10. *Procedure for computing $z_{i,j}$.*

and that for job $b_j$

$$\rho(A_k \cup B_\ell) + \sum_{r,s: b_j \in B_s} \frac{1}{p(B_s)} z_{r,s} = \rho(b_j).$$

Before we formally prove the correctness of these values for $z_{i,j}$, we use 2D Gantt charts to give some intuition of where these values come from. In the situation depicted in Figure 9, we extend the slopes of $a_3$, $a_2$, and $b_2$ to intersect the dividing line: these extensions are shown as dashed lines. If we think of the origin of the dividing line as the point at which $a_1$ and $b_1$ touch, then $\alpha_i$ is the point on the dividing line at which the slope extended from job $a_{i+1}$ touches the dividing line. Similarly, $\beta_i$ is the point on the dividing line at which the slope extended from job $b_{i+1}$ touches the dividing line. See Figure 11 for a blowup of the dividing line. Then the $z_{i,j}$ are computed by walking down the dividing line from the origin to the next point on the line: $z_{i,j}$ is the difference between the current point and the last one, and $i$ gets incremented if an $\alpha$ is encountered, whereas $j$ gets incremented if a $\beta$ is encountered. Now observe that $\frac{1}{2}\sum_{i,j}(p(A_i) + p(B_j))z_{i,j}$ expresses exactly the area above the slope of composite job $c$ but underneath the slopes of the jobs $a_i$ and $b_i$, as needed. To see this, observe that the shaded area in Figure 11 is the area of two triangles, each with base $z_{2,1} = \beta_1 - \alpha_1$, one of height $p(A_2)$ and one of height $p(B_1)$, for a total area of $\frac{1}{2}z_{2,1}(p(A_2) + p(B_1))$. The area between the lower envelope and the slopes of the jobs can be expressed as the sum of pairs of triangles like this pair, as shown in Figure 12.

We now turn to the formal proof of the correctness of the dual variables. To show that $z \geq 0$, note that by construction $z_{i,j}$ is either $\alpha_i - \alpha_{i-1}$, $\beta_j - \beta_{j-1}$, $\alpha_i - \beta_{j-1}$, or $\beta_j - \alpha_{i-1}$. In the last two cases, it follows immediately that $z_{i,j} \geq 0$: in the previous iteration of the program (when $x$ was set to $\beta_{j-1}$ (resp., $\alpha_{i-1}$)), it was the case that $\alpha_i \geq \beta_{j-1}$ (resp., $\beta_j > \alpha_{i-1}$). In the first case,

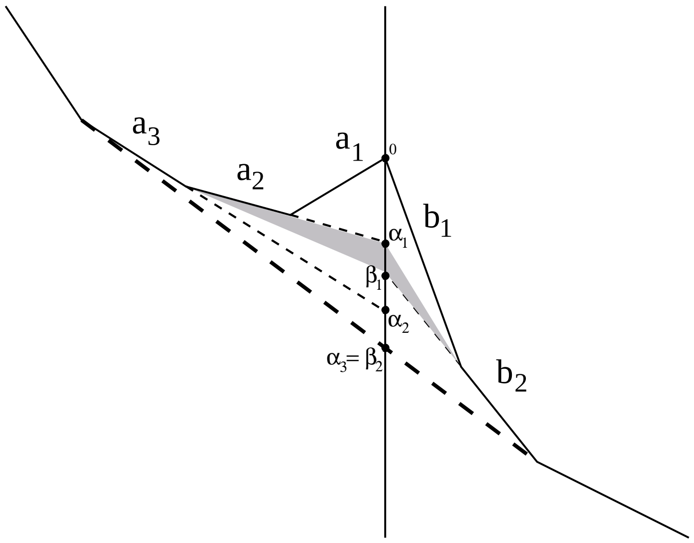


FIG. 11. *A blowup of the dividing line, and intersections of it with extensions of slopes of the $a_i$ and $b_i$, resulting in the $\alpha_{i+1}$ and $\beta_{i+1}$. By construction $z_{2,1} = \beta_1 - \alpha_1$ and the gray shaded area is the sum of two triangles of base $z_{2,1}$ and heights $p(A_2)$ and $p(B_1)$, resp.*

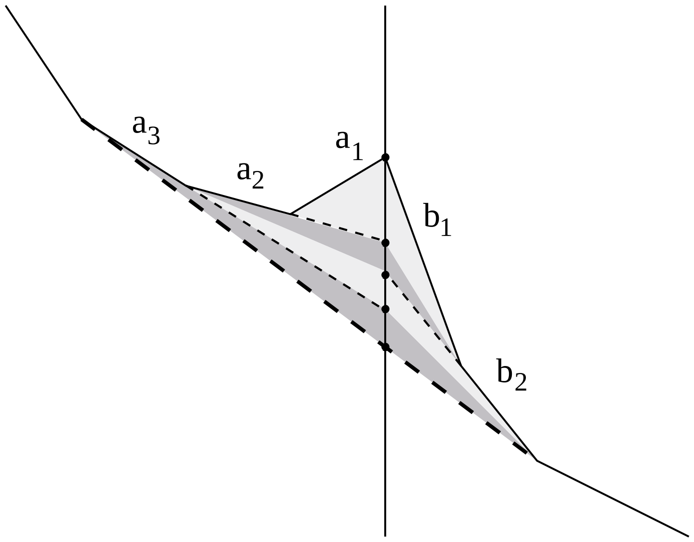


FIG. 12. *An illustration showing that the area between the slope of a composite job and its constituent jobs can be expressed as the sum of the area of pairs of triangles.*

$$\begin{aligned}\alpha_i - \alpha_{i-1} &= \rho(a_{i+1})p(A_{i+1}) - \rho(a_i)p(A_i) + w(A_i) - w(A_{i+1})\\ &= p(A_i)(\rho(a_{i+1}) - \rho(a_i)) + \rho(a_{i+1})p(a_{i+1}) - w(a_{i+1})\\ &= p(A_i)(\rho(a_{i+1}) - \rho(a_i))\\ &\geq 0.\end{aligned}$$

In the second case,

$$\begin{aligned}\beta_j - \beta_{j-1} &= w(B_{j+1}) - w(B_j) + \rho(b_j)p(B_j) - \rho(b_{j+1})p(B_{j+1}) \\ &= p(B_j)(\rho(b_j) - \rho(b_{j+1})) - \rho(b_{j+1})p(b_{j+1}) + w(b_{j+1}) \\ &= p(B_j)(\rho(b_j) - \rho(b_{j+1})) \\ &\geq 0.\end{aligned}$$

To see that

$$\rho(A_k \cup B_\ell) - \sum_{r,s: a_i \in A_r} \frac{1}{p(A_r)} z_{r,s} = \rho(a_i)$$

for job $a_i$, first observe that by construction

$$\sum_{r,s: a_i \in A_r} \frac{1}{p(A_r)} z_{r,s} = \sum_{r: a_i \in A_r} \frac{1}{p(A_r)} \sum_s z_{r,s} = \sum_{r=i}^{k} \frac{1}{p(A_r)} (\alpha_r - \alpha_{r-1}).$$

Thus we have

$$\begin{aligned}\rho(A_k \cup B_\ell) - \sum_{r,s: a_i \in A_p} \frac{1}{p(A_r)} z_{r,s} &= \rho(A_k \cup B_\ell) - \sum_{r=i}^{k} \frac{1}{p(A_r)} (\alpha_r - \alpha_{r-1}) \\ &= \rho(A_k \cup B_\ell) - \sum_{r=i}^{k} \frac{1}{p(A_r)} (p(A_r)(\rho(a_{r+1}) - \rho(a_r))) \\ &= \rho(A_k \cup B_\ell) - (\rho(A_k \cup B_\ell) - \rho(a_i)) \\ &= \rho(a_i).\end{aligned}$$

Showing that

$$\rho(A_k \cup B_\ell) + \sum_{r,s: b_j \in B_s} \frac{1}{p(B_s)} z_{r,s} = \rho(b_j)$$

for job $b_j$ is similar. First we observe that

$$\sum_{r,s: b_j \in B_s} \frac{1}{p(B_s)} z_{r,s} = \sum_{s=j}^{\ell} \frac{1}{p(B_s)} (\beta_s - \beta_{s-1}),$$

so that

$$\begin{aligned}\rho(A_k \cup B_\ell) + \sum_{r,s: b_j \in B_s} \frac{1}{p(B_s)} z_{r,s} &= \rho(A_k \cup B_\ell) + \sum_{s=j}^{\ell} \frac{1}{p(B_s)} (\beta_s - \beta_{s-1}) \\ &= \rho(A_k \cup B_\ell) + \sum_{s=j}^{\ell} \frac{1}{p(B_s)} (p(B_s)(\rho(b_s) - \rho(b_{s+1}))) \\ &= \rho(A_k \cup B_\ell) + (\rho(b_j) - \rho(A_k \cup B_\ell)) \\ &= \rho(b_j).\end{aligned}$$

After processing the root of the series-parallel structure tree, we are left with the final set of jobs $c_1, c_2, \ldots, c_k$ returned by the algorithm (some of them possibly composite jobs), with $\rho(c_1) \geq \rho(c_2) \geq \cdots \geq \rho(c_k)$, and the jobs scheduled as ordered. Let $J(c_i)$ denote the set of all the actual jobs contained in the job $c_i$. There are two possibilities. Either $\rho(c_k) \geq 0$ or $\rho(c_k) < 0$. In the first case, we can set

$$\begin{aligned} y_{J(c_1)} &= \rho(c_1) - \rho(c_2) \\ y_{J(c_1)\cup J(c_2)} &= \rho(c_2) - \rho(c_3) \\ &\vdots \\ y_{\cup_{i=1}^{k-1} J(c_i)} &= \rho(c_{k-1}) - \rho(c_k) \\ y_{\cup_{i=1}^{k} J(c_i)} &= \rho(c_k), \end{aligned}$$

so that complementary slackness is obeyed for the $y$ variables with respect to the schedule, and so that for any given actual job $j \in J(c_i)$, $\sum_{S:j\in S} y_S = \rho(c_i)$. We have therefore derived that the schedule is optimum both for the problem and for the linear programming formulation of Queyranne and Wang [11].

In the second case, i.e., when $\rho(c_k) < 0$, we can postpone the processing of the jobs $J(c_k)$ and make the schedule of arbitrarily negative objective function value. Since any feasible schedule leads to a feasible primal solution for the LP formulation we are considering, this LP is also unbounded.

In summary, we have just shown that if there is an optimum schedule, then this schedule provides an optimum solution to the LP formulation, and if there is none (and the problem is unbounded), the LP is also unbounded. We have therefore simultaneously given a proof of correctness of Lawler's algorithm and an alternate proof of the polyhedral result of Queyranne and Wang [11].

**Acknowledgments.** We thank David Shmoys for mentioning this problem to us and pointing out reference [2]. We also thank Jan Karel Lenstra for publicizing the problem in his talk at the 1997 International Symposium on Mathematical Programming, and Gideon Weiss for convincing us to modify slightly our definition of 2D Gantt charts. Finally, we thank an anonymous referee for several very useful comments.

## REFERENCES

[1] C. Chekuri and R. Motwani, *Precedence Constrained Scheduling to Minimize Weighted Completion Time on a Single Machine*, Technical Note STAN-CS-TN-97-58, Department of Computer Science, Stanford University, Stanford, CA, 1997.

[2] W. L. Eastman, S. Even, and I. M. Isaacs, *Bounds for the optimal scheduling of n jobs on m processors*, Management Sci., 11 (1964), pp. 268–279.

[3] L. Hall and F. Chudak, *private communication*, 1997.

[4] E. L. Lawler, *Sequencing jobs to minimize total weighted completion time subject to precedence constraints*, Ann. Discrete Math., 2 (1978), pp. 75–90.

[5] E. L. Lawler and J. K. Lenstra, *Machine scheduling with precedence constraints*, in Ordered Sets, I. Rival, ed., D. Reidel, Boston, MA, 1982, pp. 655–675.

[6] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *Sequencing and scheduling: Algorithms and complexity*, in Logistics of Production and Inventory, S. Graves, A. Rinnooy Kan, and P. Zipkin, eds., Handbooks Oper. Res. Management Sci. 4, North Holland, Amsterdam 1993, pp. 445–522.

[7] J. K. Lenstra and A. H. G. Rinnooy Kan, *Complexity of scheduling under precedence constraints*, Oper. Res., 26 (1978), pp. 22–35.

[8] F. Margot, M. Queyranne, and Y. Wang, *private communication*, 1997.

[9] C. L. Monma, *Properties and Efficient Algorithms for Certain Classes of Sequencing Problems*, Ph.D. thesis, Cornell University, School of Operations Research and Industrial Engineering, Ithaca, NY, 1978.

[10] C. L. Monma and J. B. Sidney, *Sequencing with series-parallel precedence constraints*, Math. Oper. Res., 4 (1979), pp. 215–224.

[11] M. Queyranne and Y. Wang, *Single-machine scheduling polyhedra with precedence constraints*, Math. Oper. Res., 16 (1991), pp. 1–20.

[12] J. B. Sidney, *Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs*, Oper. Res., 23 (1975), pp. 283–298.

[13] W. E. Smith, *Various optimizers for single-stage production*, Naval Res. Logist., 3 (1956), pp. 59–66.

[14] A. Von Arnim, U. Faigle, and R. Schrader, *The permutahedron of series-parallel posets*, Discrete Appl. Math., 28 (1990), pp. 3–9.

SIAM J. Discrete Math.
Vol. 13, No. 3, pp. 295–301

# THE SIZE OF A GRAPH WITHOUT TOPOLOGICAL COMPLETE SUBGRAPHS[*]

M. CERA[†], A. DIÁNEZ[†], AND A. MÁRQUEZ[‡]

**Abstract.** In this note we show a new upperbound for the function $ex(n; TK^p)$, i.e., the maximum number of edges of a graph of order $n$ not containing a subgraph homeomorphic to the complete graph of order $p$. Further, for $\lceil \frac{2n+5}{3} \rceil \leq p < n$ we provide exact values for this function.

**Key words.** extremal graph theory, topological complete subgraphs

**AMS subject classifications.** 05C35, 05C70

**PII.** S0895480197315941

## 1. Introduction.

As Bollobas says in [1], for many graph theorists one of the most general extremal problems is to find exact values of the function $ex(n; TK^p)$, i.e., the maximum number of edges of a graph of order $n$ not containing a subgraph homeomorphic to $K^p$, where $K^p$ is the complete graph with $p$ vertices. But exact results for that function are only known for small values of $p$, trivially $ex(n, TK^3) = n - 1$ $(n \geq 3)$, and Dirac [2] proved that $ex(n, TK^4) = 2n - 3$ $(n \geq 3)$. For the case $p = 5$, Dirac conjectured [3] that $ex(n, TK^5) = 3n - 6$. But this value remains a conjecture and only some upperbounds due to Thomassen [7] are known, as $ex(n, TK^5) \leq 4n - 11$. For greater values of $p$, Mader [5] showed that for every $p$ there exists a constant $c_p$ such that $ex(n, TK^p) \leq c_p n$ $(n \geq p)$. Later Mader [6] himself proved that

$$ex(n; TK^p) \leq t(p)n - \binom{t(p)+1}{2},$$

where $n \geq t(p) = 3 \cdot 2^{p-3} - p$ with $p \geq 4$. And by considering the graph $K^{p-2} + \overline{K^{n-p+2}}$ it is easy to get that

$$ex(n; TK^p) \geq (p-2)n - \binom{p-1}{2}.$$

This inequality gives, actually, the exact results of the function for small values of $p$, but Mader [5] noticed, considering the complete 5-partite graph with two vertices in each class, that 7 is the greatest value of $p$ for which equality can hold.

The aim of this note is to provide an upperbound for the function $ex(n; TK^p)$, showing that it is optimal in some situations, in such a way that for sufficiently large values of $n$ and $p$ the previous upperbound will be an equality. Actually, if $\lceil \frac{2n+5}{3} \rceil \leq p < n$, we will show exact values for $ex(n; TK^p)$. Notations and terminologies not explicitly given here can be found in [1].

[*]Received by the editors January 31, 1997; accepted for publication (in revised form) February 1, 2000; published electronically May 15, 2000.
http://www.siam.org/journals/sidma/13-3/31594.html

[†]E.T.S. Arquitectura, Universidad de Sevilla, Reina Mercedes 2, 41012-Sevilla, Spain (mcera@cica.es, ana@apolo.us.es).

[‡]Facultad de Informática, Universidad de Sevilla, Reina Mercedes s/n, 41012-Sevilla, Spain (almar@cica.es).

**2. Definitions and notations.** The study of $ex(n, TK^p)$ may be stated equivalently as $ex(n, TK^{n-q})$ for appropriate values of $q$. In general, the technique we will apply to get our bounds is to choose $q$ vertices of a graph $G$ with $n$ vertices in such a way that the remaining $n-q$ vertices of $G$ will be the branch vertices of the subgraph of $G$ homeomorphic to $K^{n-q}$. (A vertex is called a *branch vertex* if its degree is not 2.)

In order to achieve the goal described above, we introduce some definitions and notations. For a graph $H$ and a set $\{v_1, \dots, v_q\}$ of vertices of $H$, we denote by $H_0 = H$ and by $H_k$ for $k = 1, \dots, q$ the induced subgraph in $H$ by the set of vertices $V(H) - \{v_1, \dots, v_k\}$ $(H_k = \langle V(H) - \{v_1, \dots, v_k\}\rangle_H)$.

Most of the lemmas (as previous results) of this paper are devoted to obtaining graphs with certain prescribed properties, and those graphs will be used to prove the bounds given in the theorems. For the sake of simplicity (or in order to avoid excessive repetitions), we can say that if $r$, $t$ are two nonnegative integers, we define the family of graphs $\mathcal{C}_r^t$ as the set of graphs $H$ such that there exists a set $\{v_1, \dots, v_r\}$ of vertices of $H$ verifying

(1) $\delta_{H_{j-1}}(v_j) \geq \delta_{H_j}(v_{j+1})$, for $j = 1, \dots, r-1$.

(2) For each $h$ positive integer, if there exists $v \in H_k$ with $k = 1, \dots, r$ such that $\delta_{H_k}(v) \geq h$, then $\delta_{H_j}(v_{j+1}) \geq h$ for all $j = 1, \dots, k$.

(3) $H_r$ has at most $t$ edges $(|E(H_r)| \leq t)$.

First of all, we check that the families given above are nonempty.

LEMMA 1. *Let $H$ be a graph with $n$ vertices. Then, for any $r \leq n$, there exists $t$ such that $H$ is in $\mathcal{C}_r^t$.*

*Proof.* Let $\delta_0$ be the maximum degree of $H$ and $\Delta_0$ the set of vertices of $H$ with degree $\delta_0$. Let $v_1$ be a vertex of $\Delta_0$. We consider $H_1 = \langle V(H) - \{v_1\}\rangle_H$, $\delta_1 = \max_{v \in V(H_1)}\{\delta_{H_1}(v)\}$, and $\Delta_1$ the set of vertices of $H_1$ with degree $\delta_1$. For each $v \in \Delta_1$ we note by $H_1(v) = \langle v_1, v\rangle_H$ and $\hat{\delta_1} = \max_{v \in \Delta_1}\{\delta_{H_1(v)}(v)\}$, we get $v_2 \in \Delta_1$ such that $\delta_{H_1(v_2)}(v_2) = \hat{\delta_1}$. Then we may construct the chain of subgraph $\{H_k\}_{0 \leq k \leq r}$, noting by $\delta_k = \max_{v \in V(H_k)}\{\delta_{H_k}(v)\}$ and by $\Delta_k = \{v \in V(H_k) \ / \ \delta_{H_k}(v) = \delta_k\}$. For each $v \in \Delta_k$, let$H_k(v)$ be the subgraph $\langle v_1, \dots, v_k, v\rangle_H$ and $\hat{\delta_k} = \max_{v \in \Delta_k}\{\delta_{H_k(v)}(v)\}$. We choose $v_{k+1} \in \Delta_k$ in such a way that $\delta_{H_k(v_{k+1})}(v_{k+1}) = \hat{\delta_k}$. Hence, if $v \in V(H_k)$ and $\delta_{H_k}(v) \geq h$, then $\delta_{H_j}(v_{j+1}) \geq h$ for $j \leq k$ with $k = 1, \dots, r$. The result follows. □

In the previous lemma, the most important role is played by the vertices ($r$), and we do not care about the number of edges ($t$); obviously, from now on we will try to obtain tight values on the number of edges and from those values we will get our bounds.

**3. An upperbound of $\mathbf{ex(n, TK^p)}$.** In this section, we provide a new upperbound of $ex(n, TK^p)$; this upperbound will turn out to be the exact value of the function in many cases, as we will see in the next section.

First, we will give a sufficient condition on the edges of a graph to belong to the class $\mathcal{C}_q^q$.

LEMMA 2. *Let $n$, $q$ be two positive integers, with $q < n$. If $H$ is a graph with $n$ vertices and $2q$ edges, then*

(1) *$H \in \mathcal{C}_q^q$.*

(2) *$\delta_{H_q}(v) \leq 1$ for $v \in V(H_q)$.*

*Proof.* Let $H_q = \{v_1, \dots, v_q\}$ be a set obtained as in Lemma 1. If there exists a vertex $v \in H_q$ such that $\delta_{H_q}(v) \geq 1$, then $\delta_{H_j}(v_{j+1}) \geq 1$ for $j = 1, \dots, q$. Thus

$|E(H_q)| \leq q$, and therefore $H \in \mathcal{C}_q^q$. Furthermore, if there exists $v \in H_q$ such that $\delta_{H_q}(v) \geq 2$, then $\delta_{H_j}(v_{j+1}) \geq 2$ for every $j = 1, \ldots, q$, and hence $|E(H)| \geq 2q+2 > q$, but this is not possible. Therefore, it follows that $\delta_{H_q}(v) \leq 1$ for $v \in V(H_q)$. □

As we said before, once we have obtained in Lemma 2 an upperbound to the number of edges (refining, in this way, Lemma 1), we can get an upperbound to $ex(n, TK^p)$; thus, the first result related to this function is the following theorem.

Theorem 3. *Let $n$, $p$ be two positive integers, with $n > p$. Then*

$$(p-2)n - \binom{p-1}{2} \leq ex(n, TK^p) \leq \binom{n}{2} - (2n - 2p + 1).$$

*Proof.* The upperbound is equivalent to

$$ex(n, TK^{n-q}) \leq \binom{n}{2} - (2q+1),$$

where $q$ is a positive integer with $q < n$. Let $G$ be a graph with $\binom{n}{2} - 2q$ edges. We will prove that $G$ contains a subgraph homeomorphic to $K^{n-q}$.

Let $H = \overline{G}$ be the complement of $G$. (The complement $\overline{G}$ of a graph $G$ also has $V(G)$ as its vertex set, but two vertices are adjacent in $\overline{G}$ if and only if they are not adjacent in $G$.) By Lemma 2, there exists a subset $\{v_1, \ldots, v_q\}$ of vertices of $G$ such that $s$ ($s \leq q$) nonadjacent edges $\{e_1, \ldots, e_s\}$ are missing in $G_q$ to create a complete graph of size $n - q$. We will show that the vertices of $G_q$ are going to be branch vertices of a subgraph of $G$ homeomorphic to $K^{n-q}$. Let $a_i$, $b_i$ be the vertices of the edge $e_i$ ($e_i = (a_i, b_i)$) for $i = 1, \ldots, s$. There exists a path from $a_i$ to $b_i$ crossing a vertex of the set $\{v_1, \ldots, v_q\}$ for $i = 1, \ldots, s$ and those paths are nonadjacent for each $i \in \{1, \ldots, s\}$. These assertions can be shown if we construct the following bipartite graph: the two nonadjacent vertices sets $X = \{e_1, \ldots, e_s\}$ and $Y = \{v_1, \ldots, v_q\}$, and a vertex $e_i$ of $X$, are joined with a vertex $v_j$ of $Y$ if there exists the path $a_i v_j b_i$ in $G$. This bipartite graph has a complete matching because if there exists $e_i \in X$ nonadjacent to $v_{q-i+1} \in Y$, then either $a_i$ or $b_i$ has degree at least 2 in $H_{q-i}$. And by Lemma 2 we know that $\delta_{H_{j-1}}(v_j) \geq 2$ for every $j = 1, \ldots, q - i + 1$, thus $|E(H)| \geq 2(q - i + 1) + i - 1 + s > 2q$, but this is not possible. Hence, $G$ contains a subgraph homeomorphic to $K^{n-q}$. □

**4. Exact values for the function $\mathbf{ex(n, TK^p)}$.** The upperbound provided in Theorem 3 turns out to be the exact value of the function $ex(n, TK^p)$, for a wide interval of $p$, as stated in the next theorem.

Theorem 4. *Let $n$, $p$ be two positive integers. If $\left\lceil \frac{3n+2}{4} \right\rceil \leq p < n$, then*

$$ex(n, TK^p) = \binom{n}{2} - (2n - 2p + 1).$$

*Proof.* Consider the graph obtained from $K^n$ removing $2q + 1$ nonadjacent edges (obviously, we need that $n \geq 4q + 2$). This graph does not contain a subgraph homeomorphic to $K^{n-q}$. Hence $ex(n; TK^{n-q}) = \binom{n}{2} - (2q + 1)$. □

Now, let $G$ be a graph having $4q - k + 1$ vertices, with $q \geq 4$ and $0 \leq k \leq q - 1$, in such a way that $\overline{G}$ is the graph formed by $k + 1$ nonadjacent triangles and $2(q - k) - 1$ nonadjacent edges, as in Figure 1. If we choose a set of $q$ vertices of $G$, it is evident that $\overline{G_q}$ has at least $q + 1$ edges; hence the graph $G$ constructed does not contain a subgraph homeomorphic to $K^{n-q}$.

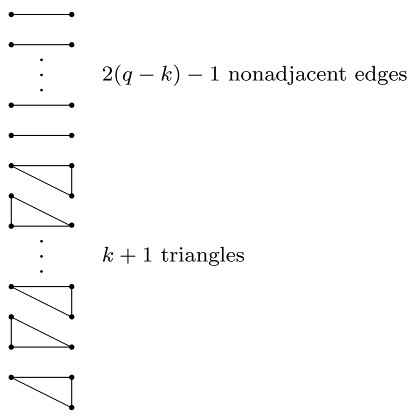


FIG. 1. *Structure of* $\overline{G}$.

LEMMA 5. *Let $q$, $k$ be two positive integers, with $q \geq 4$ and $0 \leq k \leq q-1$. It is verified that*

$$ex(4q-k+1, TK^{3q-k+1}) \geq \binom{4q-k+1}{2} - (2q+k+2).$$

Our next aim is to prove that the previous inequality is, in fact, an equality. For that goal we need some prior results. First, we will show that given a graph with maximum degree 2 and at least $m$ vertices of maximum degree, it is possible to get at least $\lceil \frac{m+2}{3} \rceil$ nonadjacent vertices of maximal degree. We recall that given a graph $H$ and $v \in H$, the set of adjacent vertices to $v$ in $H$ is denoted by $\Gamma(v)$ (see [1]).

LEMMA 6. *Let $k$ be a nonnegative integer and $H$ a graph with maximum degree 2 and at least $3k+1$ vertices of maximum degree. Then there exist at least $k+1$ nonadjacent vertices with degree 2.*

*Proof.* We apply induction on $k$. For $k = 1$ the result is obvious. Suppose now that $k > 2$ and the result holds for $k-1$. Let $w$ be a vertex of $H$ of degree 2, we denote by $H^* = \langle \{v \in V(H) - \{\Gamma(w) \cup \{w\}\}\} \rangle_H$. Hence $H^*$ is a graph with at least $3k+1-3 = 3(k-1)+1$ vertices with degree 2 and, by the induction hypothesis, $H^*$ has at least $(k-1)+1 = k$ nonadjacent vertices $\{w_1, \ldots, w_k\}$ of degree 2. Thus the $k+1$ vertices $w, w_1, \ldots, w_k$ are nonadjacent with degree 2. $\square$

The next result basically asserts that given a positive integer $q$ and given $H$, a graph with maximum degree 2 whose number of vertices and edges depends on $q$, it is possible to get a set of vertices such that upon removing those vertices from $H$, the resulting graph has at most $q$ edges.

LEMMA 7. *Let $q$, $k$ be two nonnegative integers, $k \leq q-2$. Let $H$ be a graph with $4q-k+1-i$ vertices and $2q+k+1-2i$ edges, $i \in \{1, \ldots, q\}$. If the maximum degree of $H$ is at most 2, then $H \in \mathcal{C}^q_{q-i}$.*

*Proof.* If $n_j$ denotes the number of vertices of degree $j$ in $H$, it is verified that

$$\begin{aligned} 2n_2 + n_1 &= 2(2q+k+1-2i), \\ n_2 + n_1 + n_0 &= 4q-k+1-i. \end{aligned}$$

From those equalities it is deduced that $n_2 = 3(k-i)+1+n_0$. For $i \leq k$, by Lemma 6, the number of nonadjacent vertices with degree 2 is greater than or equal to $k-i+1$. We choose $\{v_1,\dots,v_{q-i}\}$ as in Lemma 1. So we guarantee that of the previous vertices at least $k-i+1$ have degree 2, and, therefore, if we remove the vertices $\{v_1,\dots,v_{q-i}\}$ from $H$, we would delete at least $2(k-i+1)+q-(k-i+1)$ edges. Thus we have that

$$|E(H_{q-i})| \le 2q+k+1-2i-(2(k-i+1)+q-(k-i+1)) \le q.$$

For $i>k$, if we take the subset $\{v_1,\dots,v_{q-i}\}$ of vertices of $H$ like in Lemma 1, it is immediate that

$$|E(H_{q-i})| \le 2q+k+1-2i-(q-i) = q+k-i+1 \le q. \qquad \square$$

Given a graph $H$ we will show that it is possible to obtain a set of $q$ vertices in such a way that if we remove them from $H$, then the graph $H_q$ has at most $q$ edges.

LEMMA 8. *Let $q$, $k$ be two positive integers with $k \le q-2$. Let $H$ be a graph with $4q-k+1$ vertices and $2q+k+1$ edges. Then $H \in \mathcal{C}_q^q$.*

*Proof.* First, suppose that the maximum degree of $H$ is 2. In this case we have

$$\begin{aligned} 2n_2+n_1 &= 2(2q+k+1),\\ n_2+n_1+n_0 &= 4q-k+1. \end{aligned}$$

From those equalities it is deduced that $n_2 = 3k+1+n_0$ and, by Lemma 6, the number of nonadjacent vertices with degree 2 is at least $k+1$. We take $\{v_1,\dots,v_q\}$ as in Lemma 1. If the number of nonadjacent vertices of degree 2 is greater than or equal to $q$, it is obvious that $|E(H_q)| \le q$. If, on the contrary, it is smaller than $q$, then if we remove the vertices $\{v_1,\dots,v_q\}$ from $H$, we would be suppressing at least $2(k+1)+q-(k+1)$ edges and, therefore,

$$|E(H_q)| \le 2q+k+1-(2(k+1)+q-(k+1)) = q.$$

If the maximum degree of $H$ is at least 3, there exists $j \in \{1,\dots,q\}$ such that the maximum degree of $H_j$ is smaller than or equal to 2 and upon applying Lemma 7 the result follows. $\square$

In order to show that we have an equality in Lemma 5, the following theorem will be based on the same idea of the proof of Theorem 3. We start from a graph $G$ with a given number of vertices and edges and we will construct a bipartite graph in such a way that if we show the existence of a complete matching in this bipartite graph, then it will guarantee us that $G$ contains a subgraph homeomorphic to $K^{n-q}$. We recall Hall's condition for complete matching.

THEOREM 9 (see [4]). *Given a bipartite graph with classes $X$ and $Y$, if $|\Gamma(A)| \ge |A|$ for all $A \subset X$, then there exists a complete matching, where $\Gamma(A) = \bigcup_{v\in A}\Gamma(v)$.*

THEOREM 10. *Let $n$, $p$ be two positive integers with $\left\lceil \frac{2n+5}{3} \right\rceil \le p < \left\lceil \frac{3n+2}{4} \right\rceil$. Then*

$$ex(n, TK^p) = \binom{n}{2} - (5n-6p+3).$$

*Proof.* It is equivalent to prove that

$$ex(n; TK^{n-q}) = \binom{n}{2} - (2q+k+2)$$

for $n = 4q - k + 1$ with $q \geq 4,\ 0 \leq k \leq q - 4$.

Let $G$ be a graph with $\binom{n}{2} - (2q + k + 1)$ edges. Let $\{v_1, \dots, v_q\}$ be the set of vertices obtained in Lemma 8 by taking $H = \overline{G}$. Let $e_1 = (a_1, b_1), \dots, e_s = (a_s, b_s)$ be the edges of $H_q$. We will prove that there exists a path from $a_i$ to $b_i$ traversing through $\{v_1, \dots, v_q\}$ for $i = 1, \dots, s$ and that those paths are nonadjacent for each $i \in \{1, \dots, s\}$. For that, we consider the bipartite graph whose classes are $Y = \{v_1, \dots, v_q\}$ and $X = \{e_1, \dots, e_s\}$ such that $v_j$ is adjacent to $e_i$ in the bipartite graph if there exists the path $a_i v_j b_i$ in $G$. If we prove the existence of a complete matching in this bipartite graph we will have shown the result. Now we use Hall's condition to show the existence of complete matching. We get $i \in \{1, \dots, q\}$. If $e_i$ is not adjacent to any vertex of the set $\{v_{q-2}, v_{q-1}, v_q\}$, then either $\delta_{H_q}(a_i) \geq 2$ or $\delta_{H_q}(b_i) \geq 2$ and, furthermore, either $\delta_{H_{q-2}}(a_i) \geq 3$ or $\delta_{H_{q-2}}(b_i) \geq 3$; hence $\delta_{H_{j-1}}(v_j) \geq 3$ for $j = 1, \dots, q - 2$ and

$$|E(H)| \geq 3(q - 2) + 4 + s = 3q + s - 2.$$

Since $k \leq q - 3$ we would have that $|E(H)| > 2q + k + 1$, but this is not possible, thus $|\Gamma(\{e_i\})| \geq 1$.

We denote by $A_{ij}$ the set of vertices $\{e_i, e_j\}$ for $i, j \in \{1, \dots, q\}$ with $i \neq j$. If $\Gamma(A_{ij}) \leq 1$, then at least three vertices of the set $\{v_{q-3}, v_{q-2}, v_{q-1}, v_q\}$ are adjacent neither to $e_i$ nor to $e_j$. Hence $\delta_{H_{j-1}}(v_j) \geq 3$ for $j = 1, \dots, q - 3$ and

$$\begin{aligned} |E(H)| &\geq 3(q - 3) + 4 + 1 + s = 3q + s - 4 \\ &> 2q + k + 1 \end{aligned}$$

since $k \leq q - 4$; therefore $|\Gamma(A_{ij})| \geq 2$.

TABLE 1
*Exact values of the function $ex(n, TK^p)$.*

| $p$ | $ex(n, TK^p)$ | Reference |
|---|---|---|
| 3 | $n - 1$ | |
| 4 | $2n - 3$ | [2] |
| 5 | $3n - 6$ ? (conjecture) | [3] |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $\left\lceil \frac{2n+5}{3} \right\rceil \leq p < \left\lceil \frac{3n+2}{4} \right\rceil$ | $\binom{n}{2} - (5n - 6p + 3)$ | Theorem 10 |
| $\left\lceil \frac{3n+2}{4} \right\rceil \leq p < n$ | $\binom{n}{2} - (2n - 2p + 1)$ | Theorem 4 |

For sets of vertices of size $m$ with $3 \leq m \leq s$ we consider $A_{i_1,\dots,i_m} = \{e_{i_1}, \dots, e_{i_m}\}$, where $\{i_1, \dots, i_m\} \subset \{1, \dots, q\}$ with $i_1 < i_2 < \cdots < i_m$. If $\Gamma(A_{i_1,\dots,i_m}) \leq m - 1$, then there exists at least one nonadjacent vertex to any element of $A_{i_1,\dots,i_m}$ in the set of vertices $\{v_{q-(m-1)}, \dots, v_q\}$; thus

$$|E(H)| \geq (q - (m - 1))m + (m - 1) + s,$$

but for $3 \leq m \leq s,\ k \leq q-3$ we have that

$$\begin{aligned}(q-(m-1))m+(m-1)+s &\geq mq-m^2+3m-1\\ &> 2q+k+1,\end{aligned}$$

and this is not possible. So $\Gamma(A_{i_1,\ldots,i_m}) \geq m$, and upon applying Hall's condition the result follows. $\square$

**5. Conclusions.** Up to now, exact values of the function $ex(n, TK^p)$ were known only for $p = 3, 4$. In this work, we provide a new upperbound to that function and we give its exact value when $\left\lceil \frac{2n+5}{3} \right\rceil \leq p < n$. We can summarize our results in Table 1.

## REFERENCES

[1] B. Bollobas, *Extremal Graph Theory,* Academic Press, London, 1978.

[2] G. A. Dirac, *In abstrakten Graphen vorhandene vollständige* 4*-Graphenund ihre Unterteilungen*, Math. Nachr., 22 (1960), pp. 61–85.

[3] G. A. Dirac, *Homeomorphism theorem for graphs*, Math. Ann., 153 (1964), pp. 69–80.

[4] P. Hall, *On representatives of subsets*, J. London Math. Soc., 10 (1935), pp. 26–30.

[5] W. Mader, *Homomorphieegenshaften und mittlere Kantendichte von Graphen*, Math. Ann., 174 (1967), pp. 265–268.

[6] W. Mader, *Hinreichende Bedingungen für die Existenz von Teilgraphen, diezu einem vollständigen Graphen Homöomorph sind*, Math. Nachr., 53 (1972), pp. 145–150.

[7] C. Thomassen, *Some homomorphism properties of graphs*, Math. Nachr., 64 (1974), pp. 119–133.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 3, pp. 302–312

# EVALUATION, STRENGTH, AND RELEVANCE OF VARIABLES OF BOOLEAN FUNCTIONS[*]

PETER L. HAMMER[†], ALEXANDER KOGAN[†‡], AND URIEL G. ROTHBLUM[§]

**Abstract.** Given a Boolean function $f$, we define the importance of a set $S$ of variables by an expression measuring to what extent the variables in $S$ determine the value of $f$. This "evaluation" uses a "constancy" measure which is assumed to be a real-valued convex function defined on $[0, 1]$. In spite of the generality of the constancy measure, it is shown that any such evaluation is in strong agreement with the classical concept of the Winder-strength of variables of a monotone Boolean function. Further, we study a special class of evaluations called relevances, characterize completely the cases of extreme relevance value, relating the sets of maximum relevance to fictitious (dummy) variables and support sets, and establish a lower bound on the relevance of sets "containing" implicants or implicates of a Boolean function.

**Key words.** Boolean function, Winder preorder, influence of variables, support set, convex constancy measure, majorization

**AMS subject classifications.** 06E30, 94C10, 94A17

**PII.** S089548019732787X

## 1. Introduction.

It is well known that numerous real life phenomena lead to mathematical models involving Boolean functions. In such models, variables represent characteristic features (attributes) of the modeled phenomenon, and the value of the function represents the outcome. In many applications, it is important to distinguish variables having a major impact on the outcome from those having only a minor effort on it. This analysis of the "importance" of variables in a Boolean function may in some instances be one of the main objectives of model building.

Concepts of importance of variables of a Boolean function were introduced and studied in various forms in areas as diverse as logic design, voting and game theory, reliability theory, artificial intelligence, and machine learning. Some of the most fundamental notions of importance originating in these areas include those of Chow parameters (Chow [5, 6] and Winder [20]), Winder preorder (Winder [18, 19]), Shapley values (Shapley and Shubik [16]), and Banzaf indices (Banzaf [1] and Dubey and Shapley [7]). While this research direction started more than forty years ago, it remains active in many areas until this day (Ben Orr and Linial [2], Ben Orr, Linial, and Saks [3], Felsenthal and Machover [9], Kahn, Kalai, and Linial [11], and Taylor and Zwicker [17]).

The approaches taken by various authors to the concept of importance are extremely varied. Some authors look for numerical indices measuring importance of individual variables, while others express the importance by intervals, vectors, etc.

[*]Received by the editors March 22, 1997; accepted for publication (in revised form) December 1, 1999; published electronically May 15, 2000. The research of the first and second authors was partially supported by ONR grant N00014-92-J-1375 and by NSF grant NSF-DMS-98-06389. The research of the third author was partially supported by ONR grant N00014-92-J-1142.

http://www.siam.org/journals/sidma/13-3/32787.html

[†]RUTCOR—Rutgers University's Center for Operations Research, Rutgers University, 640 Bartholomew Road, Piscataway, New Jersey 08854-8003 (hammer@rutcor.rutgers.edu, kogan@rutcor.rutgers.edu).

[‡]Accounting and Information Systems, Faculty of Management, Rutgers University, 180 University Ave., Newark, NJ 07102.

[§]Faculty of Industrial Engineering and Management, Technion—Israel Institute of Technology, Haifa 32000, Israel (rothblum@ie.technion.ac.il).

Some other papers define power indices for sets of variables, or simply define conditions under which a group of variables can be considered "more influential" than another group.

In spite of the wide diversity of approaches to the concepts of importance (or power, influence, relevance, etc.), it seems to be generally agreed upon that a good measure should reflect the extent to which a set of variables is capable of determining the value of the function. Because of that, the various concepts of importance rely heavily on the particular measure of "constancy" of a Boolean function, i.e., how "constant" a Boolean function becomes under certain partial truth value assignments. In theoretical studies, the most used constancy measure was the simplest Boolean measure, i.e., 1 if the function is a constant, and 0 if it is not. Some of the more applied studies (for example, the splitting criteria in decision tree inference in machine learning, see Breiman [4] and Quinlan [15]) use more sophisticated measures of constancy, usually derived from such standard measures as variance or entropy.

In this paper, we consider a new concept of importance based on an abstract numerical measure of constancy which depends only on the ratio of the number of true points to the total number of points in the Boolean cube. In the first part of the paper (section 3), this constancy measure is assumed to be an arbitrary real-valued convex function defined on $[0, 1]$. In spite of the generality of the constancy measure, it is shown in this paper that any such measure of importance is in strong agreement with the classical concept of the Winder-strength of variables of a monotone Boolean function. The proof of this result, given in section 3, relies on a combinatorial lemma of independent interest and on the theory of majorization (Marshall and Olkin [13]).

In the second part of the paper (section 4), the measure of constancy is given a more specific meaning. It is assumed now that this measure is a real-valued convex function $\rho$ defined on $[0, 1]$, such that $\rho(0) = \rho(1)$, $\rho(1/2) = 0$, and $\rho(x) = \rho(1 - x)$ for any $x$ in $[0, 1]$. This restricted concept of constancy remains wide enough to include as special cases all the commonly used measures of constancy, such as the Boolean measure and the linear measure, as well as the measures based on variance and entropy. The importance of variables based on this constancy measure is called *relevance*. We characterize completely the cases of extreme relevance value, relate the sets of maximum relevance to the well-known concepts of fictitious (dummy) variables and support sets, and establish a lower bound on the relevance of a set "containing" implicants or implicates of a Boolean function. For the sake of brevity, proofs of the results in section 4 were omitted; they can be found in [10].

**2. Boolean functions and evaluations.** We refer to 0 and 1 as the *Boolean constants* and to vectors whose components are Boolean constants as *Boolean vectors*.

A *Boolean function of n variables* is a function $f : \{0, 1\}^n \to \{0, 1\}$. Given such a function $f$, we say that a Boolean vector $\alpha$ is a *true (false) vector* of $f$ if $f(\alpha) = 1$ ($f(\alpha) = 0$). The *cardinality* of $f$, denoted $|f|$, is defined as the cardinality of the set of true vectors of $f$, i.e., $|f| = \sum_{\alpha \in \{0,1\}^n} f(\alpha)$.

Throughout this paper, $n$ is a fixed positive integer. The variables of a Boolean function of $n$ variables are always denoted as $x_1, x_2, \ldots, x_n$; we will usually identify these variables with the corresponding indices $1, \ldots, n$. In particular, we let $N$ denote both the set of variables $x_1, x_2, \ldots, x_n$ and the set of integers $\{1, \ldots, n\}$. Given a subset $S$ of $N$, we denote by $x_S$ the subvector of $x = (x_1, \ldots, x_n)$ corresponding to $S$, where the order of the variables of $x_S$ is induced by their (natural) order in $N$.

We use the standard partial order $\leq$ on Boolean vectors; namely, we write $\alpha \leq \beta$ if $\alpha_i \leq \beta_i$ for all $i \in N$. Given two Boolean functions, $f$ and $g$, defined on the same set of $n$ variables, we shall write $f \leq g$ if $f(\alpha) \leq g(\alpha)$ for every $\alpha \in \{0,1\}^n$. A Boolean function of $n$ variables is called *monotone nondecreasing*, or simply *monotone* or *positive*, if $f(\alpha) \leq f(\beta)$ for all Boolean vectors $\alpha, \beta \in \{0,1\}^n$ satisfying $\alpha \leq \beta$.

Given a Boolean function $f$ of $n$ variables, a subset $S$ of its variable set $N$, and a vector $\alpha \in \{0,1\}^{|S|}$, we denote by $f(x_S = \alpha)$ the Boolean function obtained by fixing the values of the variables of $S$ to the corresponding values of $\alpha$; in this way, the variables of $f(x_S = \alpha)$ are those in $N \backslash S$. In particular, if $S = N$ and $\alpha \in \{0,1\}^n$, $f(x = \alpha)$ is the Boolean constant $f(\alpha)$. Also, from the definition of the cardinality of a Boolean function, we see that

$$0 \leq |f(x_S = \alpha)| \leq 2^{n-|S|} \text{ for every subset } S \text{ of } N \text{ and } \alpha \in \{0,1\}^{|S|}.$$

If $S$ and $T$ are disjoint subsets of $N$, $\alpha \in \{0,1\}^{|S|}$ and $\beta \in \{0,1\}^{|T|}$, we denote $[f(s_X = \alpha)](x_T = \beta) = [f(x_T = \beta)](x_S = \alpha)$ by $f(x_S = \alpha,\, x_T = \beta)$. Also, if $S$ and $T$ are singletons, say $S = \{i\}$ and $T = \{j\}$, we write $f(x_i = \alpha)$ and $f(x_i = \alpha,\, x_j = \beta)$ for $f(x_S = \alpha)$ and $f(x_S = \alpha,\, x_T = \beta)$, respectively. Given a subset $S$ of $N$ and $\alpha \in \{0,1\}^{|S|}$, we sometimes refer to $x_S = \alpha$ as *the partial assignment of the variables of $S$ to* the corresponding Boolean constants.

As usual, the *disjunction* $a \vee b$ represents the maximum of $a$ and $b$. Similarly, the *conjunction* $a \wedge b$, also denoted by $a$ & $b$ or simply by $ab$, denotes the minimum of $a$ and $b$. For example, $f(x) = x_1 \vee x_2 x_3$ is the Boolean function on three variables given by $f(\alpha) = 1$ if $\alpha_1 = 1$ or if $\alpha_2 = \alpha_3 = 1$, and $f(\alpha) = 0$ in all other cases.

A subset $C$ of the reals is called *convex* if $\tau b + (1-\tau)c \in C$ for all $b, c \in C$ and any $0 \leq \tau \leq 1$. A real-valued function $\rho$ on such a convex set $C$ is called *convex* if $\rho[\tau b + (1-\tau)c] \leq \tau\rho(b) + (1-\tau)\rho(c)$ for all $b, c \in C$ and $0 \leq \tau \leq 1$.

The main objective of this paper is to analyze the extent to which a subset of variables of a Boolean function determines the value of the function. In order to be able to give a precise meaning to the concept, we shall introduce the following definition.

Definition 2.1. *Given a Boolean function $f$ of $n$ variables, for any real-valued convex function $\rho$ on $[0,1]$, we define the $(\rho, f)$*-evaluation *as the set function $E_{\rho,f} : 2^N \to R$, given by*

$$E_{\rho,f}(S) = \frac{1}{2^{|S|}} \sum_{\alpha \in \{0,1\}^{|S|}} \rho\left[\frac{|f(x_S = \alpha)|}{2^{n-|S|}}\right] \text{ for each subset } S \text{ of } N. \tag{2.1}$$

For example, $E_{\rho,f}(\emptyset) = \rho[\frac{|f|}{2^n}]$ and $E_{\rho,f}(N) = 2^{-n}\{\sum_{\alpha \in \{0,1\}^n} \rho[f(\alpha)]\}$.

The evaluation $E_{\rho,f}(S)$ can be viewed as a measure of the average constancy of the Boolean function $f(x_S = \alpha)$. The constancy measure $\rho[\frac{|f(x_S=\alpha)|}{2^{n-|S|}}]$ is an abstraction of constancy measures used in various areas such as statistics (e.g., variance), information theory (e.g., entropy), etc.

Using the convexity of $\rho$, it is easy to show that *evaluation is nondecreasing with respect to set inclusion*, i.e., $E_{\rho,f}(S) \leq E_{\rho,f}(T)$ for any Boolean function $f$, any real-valued convex function $\rho$ defined on $[0,1]$, and any subsets $S$ and $T$ of $N$ satisfying $S \subseteq T$. Consequently, for any subset of variables $S \subseteq N$, we have $E_{\rho,f}(\emptyset) \leq E_{\rho,f}(S) \leq E_{\rho,f}(N)$ . The value $E_{\rho,f}(\emptyset)$ measures the constancy of the function $f$ and represents the zero level of information (i.e., no variable values are expected to be known). On the other hand, the value of $E_{\rho,f}(N)$ represents the complete information (i.e., all the values are expected to be known). It is, therefore, natural to measure the *relative* *information content* of the set $S$ by

$$I_{\rho,f}(S) = \frac{E_{\rho,f}(S) - E_{\rho,f}(\emptyset)}{E_{\rho,f}(N) - E_{\rho,f}(\emptyset)}\ . \tag{2.2}$$

Clearly, the value of $I_{\rho,f}(S)$ is not defined when $E_{\rho,f}(N) = E_{\rho,f}(\emptyset)$. This special case will be characterized in Lemma 4.8 showing that it holds exactly when $f$ is a constant.

## 3. Evaluations and strength of variables.

After introducing in this section the concept of strength of a variable in a positive Boolean function, we prove that sets of stronger variables have higher evaluations. The proof will make use of a technical result (Lemma 3.2) which may be of independent combinatorial interest.

Winder [18] introduced a partial order on the variables of a Boolean function (also Muroga [14]). Let $f$ be a Boolean function of $n$ variables, and let $i$ and $j$ be the indices of two of its variables. We say that $x_i$ is *at least as strong as $x_j$ with respect to $f$* (according to the Winder partial order), written as $x_i \gtrsim_f x_j$, or simply $i \gtrsim_f j$, if

$$f(x_i = 0,\ x_j = 1) \leq f(x_i = 1,\ x_j = 0). \tag{3.1}$$

For example, if $f(x) = x_1 \vee x_2x_3$, we have $1 \gtrsim_f 2$ because $f(1,0,0) = 1 \geq 0 = f(0,1,0)$ and $f(1,0,1) = 1 \geq 1 = f(0,1,1)$.

We shall further define a set of variables $S \subseteq N$ to be *at least as strong as* a set of variables $T \subseteq N$ *with respect to $f$* (according to the Winder partial order), written $S \gtrsim_f T$, if $|S| = |T|$, and there is a one-to-one correspondence between the variables of $S$ and those of $T$ such that each $i \in S$ is at least as strong (with respect to $f$) as its image in $T$. If $S \gtrsim_f T$ and $T \gtrsim_f S$, we shall say that $S$ and $T$ are *equivalent in strength with respect to $f$* and write $S \approx_f T$.

THEOREM 3.1. *Let $f$ be a positive Boolean function of $n$ variables, and let $S$ and $T$ be two subsets of the set of variables of $f$ such that $S \gtrsim_f T$. Then for every real-valued convex function $\rho$ defined on $[0,1]$, $E_{\rho,f}(S) \geq E_{\rho,f}(T)$.*

In order to prove this theorem, we make use of a technical lemma, which needs some new concepts. For a set $Q$ of $n$-dimensional Boolean vectors, we denote by $Q^c$ its *complement* in $\{0,1\}^n$; that is, $Q^c \equiv \{0,1\}^n \setminus Q$. A set $Q \subseteq \{0,1\}^n$ is called a *filter* (*ideal*) if for any $\alpha \in Q$, all vectors $\beta \in \{0,1\}^n$ satisfying $\beta \geq \alpha$ ($\beta \leq \alpha$) are in $Q$. Note that $Q$ is a filter if and only if $Q^c$ is an ideal. Given subsets $U$ and $V$ of $\{0,1\}^n$, a *right-shift of* $U \times V$ is a mapping of $U \times V$ into $\{0,1\}^n \times \{0,1\}^n$ such that if $(u', v')$ is the image of $(u, v)$, then for every $i = 1, \ldots, n$ either ($u'_i = u_i$ and $v'_i = v_i$) or ($u'_i < u_i$ and $v'_i > v_i$).

LEMMA 3.2.[1] *For any filter $Q \subseteq \{0,1\}^n$, there exists a one-to-one right-shift of $Q \times Q^c$ onto $Q^c \times Q$.*

*Proof.* The proof uses induction on the dimension $n$. For $n = 1$, the only nontrivial filter is $Q = \{1\}$, in which case $Q \times Q^c = \{(1,0)\}$, $Q^c \times Q = \{(0,1)\}$, and the mapping of $(1,0)$ to $(0,1)$ is a one-to-one right-shift of $Q \times Q^c = \{(1,0)\}$ onto $Q^c \times Q = \{(0,1)\}$.

Assume that for some positive integer $k$, the lemma holds for $n < k$, and consider a filter $Q \subseteq \{0,1\}^k$. Let $Q_0 \equiv \{\alpha \in \{0,1\}^{k-1} : (\alpha, 0) \in Q\}$ and $Q_1 \equiv \{\alpha \in \{0,1\}^{k-1}$ :

[1]The main theorem of [8] asserts that "If a family $A_1, A_2, \ldots, A_s$ of subsets of a set $M$ contains every subset of each member, then the complements in $M$ of the $A$'s have a permutation $C_1, C_2, \ldots, C_s$ such that $C_i \supset A_i$;" a short proof of this is given in [12, problem 13.9]. Recently Ron Aharoni and Ron Holzman found an alternative proof of Lemma 3.2 by a straightforward reduction to the above theorem. Thereafter, we obtained an alternative proof of the above theorem based on Lemma 3.2, thus establishing the equivalence of the two results.

$(\alpha, 1) \in Q\}$. As $Q$ is a filter in $\{0,1\}^k$, $Q_0$ and $Q_1$ are filters (in $\{0,1\}^{k-1}$), $Q_0 \subseteq Q_1$, $Q_1^c \subseteq Q_0^c$, and $Q_1 \setminus Q_0 = Q_0^c \setminus Q_1^c$. In the following arguments, the degenerate cases with $Q_0 = \emptyset$ and/or $Q_1 = \{0,1\}^{k-1}$ require special attention.

We notice that $Q \times Q^c$ is isomorphic (through reordering coordinates and shifting parentheses) to the union of the following four disjoint sets:

(a) $(Q_0 \times Q_0^c) \times (\{0\} \times \{0\})$,
(b) $(Q_0 \times Q_1^c) \times (\{0\} \times \{1\})$,
(c) $(Q_1 \times Q_0^c) \times (\{1\} \times \{0\})$, and
(d) $(Q_1 \times Q_1^c) \times (\{1\} \times \{1\})$.

Further, as $Q_0 \subseteq Q_1$ and $Q_1^c \subseteq Q_0^c$, we see that $(Q_1 \times Q_0^c) \times (\{1\} \times \{0\})$ is the union of the following four disjoint sets:

(c1) $[Q_0 \times (Q_0^c \setminus Q_1^c)] \quad \times(\{1\} \times \{0\})$,
(c2) $[(Q_0 \times Q_1^c)] \quad \times(\{1\} \times \{0\})$,
(c3) $[(Q_1 \setminus Q_0) \times Q_1^c] \quad \times(\{1\} \times \{0\})$, and
(c4) $[(Q_1 \setminus Q_0) \times (Q_0^c \setminus Q_1^c)] \quad \times(\{1\} \times \{0\})$.

Therefore, by identifying isomorphic sets, we conclude that $Q \times Q^c$ has a decomposition into seven disjoint sets. Similarly, $Q^c \times Q$ can be decomposed into the union of seven disjoint sets which are isomorphic to:

(A) $(Q_0^c \times (Q_0) \quad \times(\{0\} \times \{0\})$,
(B) $(Q_1^c \times Q_0) \quad \times(\{1\} \times \{0\})$,
(C1) $[(Q_0^c \setminus Q_1^c) \times Q_0] \quad \times(\{0\} \times \{1\})$,
(C2) $[Q_1^c \times Q_0] \quad \times(\{0\} \times \{1\})$,
(C3) $[Q_1^c \times (Q_1 \setminus Q_0)] \quad \times(\{0\} \times \{1\})$,
(C4) $[(Q_0^c \setminus Q_1^c) \times (Q_1 \setminus Q_0)] \quad \times(\{0\} \times \{1\})$, and
(D) $(Q_1^c \times Q_1) \quad \times(\{1\} \times \{1\})$.

TABLE 1
*Definition of the right-shift $g$ of $Q \times Q^c$ onto $Q^c \times Q$.*

| Index of the set containing (x,y,u,v) | x ∈ | y ∈ | u | v | g(x,y,u,v) | Index of the set containing g(x,y,u,v) |
|---|---|---|---|---|---|---|
| a | $Q_0$ | $Q_0^c$ | 0 | 0 | $[g_0$ (x,y),0,0] | A |
| b | $Q_0$ | $Q_1^c$ | 0 | 1 | $[g_0$ (x,y), 0, 1] | C1 ∪ C2 |
| c1 | $Q_0$ | $Q_0^c \setminus Q_1^c$ | 1 | 0 | $[g_0$ (x,y), 0, 1 ] | C1 ∪ C2 |
| c2 ∪ c3 | $Q_1$ | $Q_1^c$ | 1 | 0 | $[g_1$ (x,y), 1, 0] if $g_1(x,y) \in Q_1^c \times Q_0$ | B |
| | | | | | $[g_1$ (x,y), 0, 1] if $g_1(x,y) \in Q_1^c \times (Q_1 \setminus Q_0)$ | C3 |
| c4 | $Q_1 \setminus Q_0$ | $Q_0^c \setminus Q_1^c$ | 1 | 0 | (x,y,0,1) | C4 |
| d | $Q_1$ | $Q_1^c$ | 1 | 1 | $[g_1$ (x,y), 1, 1] | D |

A one-to-one right-shift of $Q \times Q^c$ can now be constructed by considering separately the pieces of $Q \times Q^c$, as is summarized in Table 1. (For the row indexed by $c4$ recall that $Q_1 \setminus Q_0 = Q_0^c \setminus Q_1^c$.) As $g_0$ and $g_1$ are one-to-one right-shifts of $Q_0 \times Q_0^c$ onto $Q_0^c \times Q_0$ and of $Q_1 \times Q_1^c$ onto $Q_1^c \times Q_1$, respectively, it is easy to verify that the function $g$ defined by the above table is a one-to-one right-shift of $Q \times Q^c$ onto $Q^c \times Q$. This completes our inductive argument. $\square$

Let $u$ be a real-valued function on a finite set $A$. For $B \subseteq A$, let $u(B) = \sum_{b \in B} u(b)$. For each integer $1 \leq i \leq |A|$, let $u^i = \max_{B \subseteq A, |B|=i} u(B)$. If $u$ and $v$ are real-valued functions defined on the same finite set $A$, we say that $v$ *majorizes* $u$

if $v^i \geq u^i$ for each $1 \leq i \leq |A|$, and $v^{|A|} = u^{|A|}$. Standard results about majorization and Schur convexity (e.g., Marshall and Olkin [13, p. 64]) show that if $v$ majorizes $u$, then for every convex function $\rho$, we have $\sum_{a \in A} \rho(v(a)) \geq \sum_{a \in A} \rho(u(a))$.

We are now ready to prove Theorem 3.1.

*Proof.* For a subset $U$ of $\{1, \dots, n\}$, let $|f_U|$ be the function which maps every $\alpha \in \{0,1\}^{|U|}$ to $|f(x_U = \alpha)|$. Let $m$ be the common cardinality of $S$ and $T$, and let $q$ be the common cardinality of $S \setminus T$ and $T \setminus S$. We will show that $|f_S|$ majorizes $|f_T|$. It will then follow that $2^{n-m}|f_S|$ majorizes $2^{n-m}|f_T|$, and, therefore, from the convexity of $\rho$,

$$2^m E_{\rho,f}(S) = \sum_{\alpha \in \{0,1\}^m} \rho\left[\frac{|f(x_S = \alpha)|}{2^{n-m}}\right] \geq \sum_{\alpha \in \{0,1\}^m} \rho\left[\frac{|f(x_T = \alpha)|}{2^{n-m}}\right] = 2^m E_{\rho,f}(T)\,.$$

As the Winder partial order is transitive, it follows that $S \setminus T \gtrsim_f T \setminus S$. Since the definition of majorization is invariant under coordinate renumbering, we may assume that the coordinates of $S \setminus T$ and $T \setminus S$ are renumbered so that for $k = 1, \dots, q$, with $i_k$ and $j_k$ as the $k$th smallest coordinate of $S \setminus T$ and $T \setminus S$, respectively, we have that $i_k \gtrsim_f j_k$. Once such renumbering is implemented, it follows that for each $\alpha, \beta \in \{0,1\}^q$, $\gamma \in \{0,1\}^{m-q}$, and every right-shift $g$ of $\{\alpha\} \times \{\beta\}$, we have

$$(3.2)\quad |f(x_{S\setminus T, T\setminus S} = g(\alpha,\beta),\ x_{S\cap T} = \gamma)| \leq |f(x_{S\setminus T, T\setminus S} = (\alpha,\beta),\ x_{S\cap T} = \gamma)|\,.$$

(We note that the definition of a right-shift is not invariant with respect to coordinate renumbering.)

Let $1 \leq t \leq 2^m$. We will prove that $|f_S|^t \geq |f_T|^t$. The positivity of $f$ implies that $|f_T|$ is monotone, and therefore there exists a filter $Q \subseteq \{0,1\}^m$ such that $|f_T|^t = \sum_{\delta \in Q} |f(x_T = \delta)|$. For each $\gamma \in \{0,1\}^{m-q}$, let $Q_\gamma$ be the filter $\{\beta \in \{0,1\}^q : (\beta, \gamma) \in Q\}$. We then have that

$$\begin{aligned}
|f_T|^t &= \sum_{\gamma \in \{0,1\}^{m-q}} \sum_{\beta \in Q_\gamma} |f(x_{T\setminus S} = \beta,\ x_{S\cap T} = \gamma)| \\
&= \sum_{\gamma \in \{0,1\}^{m-q}} \sum_{\beta \in Q_\gamma} \sum_{\alpha \in \{0,1\}^q} |f(x_{S\setminus T} = \alpha,\ x_{T\setminus S} = \beta,\ x_{S\cap T} = \gamma)| \\
(3.3)\qquad &= \sum_{\gamma \in \{0,1\}^{m-q}} \sum_{\alpha \in \{0,1\}^q} \sum_{\beta \in Q_\gamma} |f(x_{S\setminus T} = \alpha,\ x_{T\setminus S} = \beta,\ x_{S\cap T} = \gamma)|.
\end{aligned}$$

We also note that

$$\begin{aligned}
(3.4)\quad |f_S|^t &\geq \sum_{\delta \in Q} |f(x_S = \delta)| \\
(3.5)\qquad &= \sum_{\gamma \in \{0,1\}^{m-q}} \sum_{\alpha \in Q_\gamma} \sum_{\beta \in \{0,1\}^{m-q}} |f(x_{S\setminus T} = \alpha,\ x_{T\setminus S} = \beta,\ x_{S\cap T} = \gamma)|.
\end{aligned}$$

We conclude that

$$|f_S|^t - |f_T|^t \geq \sum_{\gamma\in\{0,1\}^{m-q}} \left( \sum_{(\alpha,\beta)\in Q_\gamma\times(Q_\gamma)^c} |f((x_{S\setminus T}, x_{T\setminus S}) = (\alpha,\beta),\ x_{S\cap T} = \gamma)| \right.$$

$$\left. - \sum_{(\alpha,\beta)\in(Q_\gamma)^c\times Q_\gamma} |f((x_{S\setminus T}, x_{T\setminus S}) = (\alpha,\beta),\ x_{S\cap T} = \gamma)| \right). \tag{3.6}$$

Let $\gamma \in \{0,1\}^{m-q}$. Since $Q_\gamma$ is a filter, Lemma 3.2 implies that there exists a one-to-one right-shift, say $g_\gamma$, of $Q_\gamma \times (Q_\gamma)^c$ onto $(Q_\gamma)^c \times Q_\gamma$. Using (3.2), it then follows from $S \setminus T \gtrsim_f T \setminus S$ that

$$\sum_{(\alpha,\beta)\in(Q_\gamma)^c\times Q_\gamma} |f((x_{S\setminus T}, x_{T\setminus S}) = (\alpha,\beta),\ x_{S\cap T} = \gamma)|$$

$$= \sum_{(\alpha,\beta)\in Q_\gamma\times(Q_\gamma)^c} |f((x_{S\setminus T}, x_{T\setminus S}) = g_\gamma(\alpha,\beta),\ x_{S\cap T} = \gamma)|$$

$$\leq \sum_{(\alpha,\beta)\in Q_\gamma\times(Q_\gamma)^c} |f((x_{S\setminus T}, x_{T\setminus S}) = (\alpha,\beta),\ x_{S\cap T} = \gamma)|. \tag{3.7}$$

From (3.6) and (3.7) we conclude that $|f_S|^t \geq |f_T|^t$. We finally note that for $t = 2^m$, $Q = \{0,1\}^m$, and equalities hold trivially in (3.5), (3.6), and (3.7), completing the proof of the theorem. □

COROLLARY 3.3. *Let $f$ be a positive Boolean function of $n$ variables, and let $S$ and $T$ be two subsets of the set of variables of $f$ such that there exists a subset $S'$ of $S$ with $S' \gtrsim_f T$. Then for every real-valued convex function $\rho$ defined on* $[0,1]$, *$E_{\rho,f}(S) \geq E_{\rho,f}(T)$.*

COROLLARY 3.4. *Let $f$ be a positive Boolean function of $n$ variables, and let $S$ and $T$ be two subsets of the set of variables of $f$ such that $S \approx_f T$. Then for every real-valued convex function $\rho$ defined on* $[0,1]$, *$E_{\rho,f}(S) = E_{\rho,f}(T)$.*

We note that if the sets $S$ and $T$ are such that $S \gtrsim_f T$ but $S$ and $T$ are not equivalent in strength (i.e., $T \not\gtrsim_f S$), then it is not necessary that $E_{\rho,f}(S) > E_{\rho,f}(T)$. For example, consider the Boolean function $f = x_1 \vee x_2 \vee x_3x_4$. Let $S = \{1,2\}$ and $T = \{3,4\}$. Clearly, $S \gtrsim_f T$, $T \not\gtrsim_f S$, and

| $\alpha$ | $\lvert f(x_S = \alpha)\rvert$ | $\lvert f(x_T = \alpha)\rvert$ |
|---|---|---|
| $(0,0)$ | 1 | 3 |
| $(0,1)$ | 4 | 3 |
| $(1,0)$ | 4 | 3 |
| $(1,1)$ | 4 | 4 |

Let $\rho(c) = c$ for $c = [0,1]$. Then $E_{\rho,f}(S) = \frac{1}{4}\left(\frac{1}{4}+1+1+1\right) = \frac{13}{16}$ and $E_{\rho,f}(T) = \frac{1}{4}\left(\frac{3}{4}+\frac{3}{4}+\frac{3}{4}+1\right) = \frac{13}{16}$. However, for the function $\rho'$ given by $\rho'(c) = c^2$, we find that $E_{\rho',f}(S) = \frac{49}{64} > \frac{43}{64} = E_{\rho',f}(T)$.

**4. Evaluations and relevances.** A real-valued convex function $\rho$ on $[0,1]$ which is symmetric around $\frac{1}{2}$ (i.e., $\rho(c) = \rho(1-c)$ for all $0 \leq c \leq 1$) and satisfies $\rho\left(\frac{1}{2}\right) = 0$ and $\rho(0) = \rho(1) = 1$ is called a *relevance function*. The following are four standard relevance functions:

(i) the *consensus relevance function* given by

$$\rho(c) = \begin{cases} 0 & \text{if } 0 < c < 1, \\ 1 & \text{if } c = 0 \text{ or } c = 1, \end{cases}$$

(ii) the *linear relevance function* given by

$$\rho(c) = 2\left|c - \frac{1}{2}\right| ,$$

(iii) the *quadratic (variance) relevance function* given by

$$\rho(c) = 4\left(c - \frac{1}{2}\right)^2 , \quad \text{and}$$

(iv) the *entropy relevance function* given by

$$\rho(c) = 1 + c(\log_2 c) + (1-c)\log_2(1-c).$$

In what follows we focus our attention on those $(\rho, f)$-evaluations $E_{\rho,f}$ where $\rho$ is a relevance function.

LEMMA 4.1. *Let $\rho$ be a relevance function. Then*

(a) $0 \le \rho(c) \le 1$ *for all* $0 \le c \le 1$,

(b) $\rho(c) = 1$ *if and only if* $c = 0$ *or* $c = 1$, *and*

(c) *either* $\{c \in [0,1] : \rho(c) = 0\} = \left[\frac{1}{2} - \delta,\ \frac{1}{2} + \delta\right]$ *for some* $0 \le \delta < \frac{1}{2}$, *or* $\{c \in [0,1] : \rho(c) = 0\} = \left(\frac{1}{2} - \delta,\ \frac{1}{2} + \delta\right)$ *for some* $0 < \delta \le \frac{1}{2}$.

The third statement of Lemma 4.1 asserts that the set of elements attaining the minimum value 0 is a symmetric interval around $\frac{1}{2}$, not containing 0 or 1; this interval can be as large as $(0,1)$, in the case of the consensus relevance function, or as small as $\left\{\frac{1}{2}\right\}$, for example, for the linear, quadratic, and entropy relevance functions.

LEMMA 4.2. *Let $\rho$ be a relevance function, and let $f$ be a Boolean function. Then* $0 \le E_{\rho,f}(S) \le 1$ *for all subsets $S$ of $N$.*

Let $f$ be a Boolean function. A subset $S$ of the variables of $f$ is called a *support set* of $f$ if the value of the function $f$ is completely determined by the values of the variables in $S$. The next theorem characterizes support sets of Boolean functions in terms of relevance functions and evaluations.

THEOREM 4.3. *Let $f$ be a Boolean function and let $S$ be a subset of $N$. Then the following are equivalent.*

(a) *$S$ is a support set of $f$.*

(b) $E_{\rho,f}(S) = E_{\rho,f}(N)$ *for every real-valued convex function $\rho$ defined on* $[0,1]$.

(c) $E_{\rho,f}(S) = 1$ *for some relevance function $\rho$.*

Let $f$ be a Boolean function of $n$ variables. Variable $x_i$ of $f$ is called *fictitious for $f$* if $f(\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n) = f(\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n)$ for every $\alpha \in \{0,1\}^n$.

LEMMA 4.4. *Let $f$ be a Boolean function of $n$ variables. A set $S$ is a support set of $f$ if and only if all the variables in $N \setminus S$ are fictitious for $f$.*

COROLLARY 4.5. *Let $f$ be a Boolean function of $n$ variables, and let $S$ be the set of all nonfictitious variables of $f$. Then*

(a) *a set $T \subseteq N$ is a support set of $f$ if and only if $T \supseteq S$,*

(b) $E_{\rho,f}(S) = E_{\rho,f}(N)$ *for each real-valued convex function $\rho$ defined on* $[0,1]$, *and*

(c) *if $E_{\rho,f}(T) = 1$, for some relevance function $\rho$ and subset $T$ of $N$, then $T \supseteq S$.*

Following Corollary 4.5, we refer to the set $S$ of all variables that are not fictitious for a Boolean function $f$ as the *(unique) minimum support set* of $f$. Corollary 4.5 shows that this set can be determined by *separately* testing each variable for fictitiousness, and therefore, the time complexity of the problem of finding the minimum support set is polynomially equivalent to the problem of testing if a variable is fictitious for a given Boolean function.

THEOREM 4.6. *Let $f$ be a Boolean function. If $y$ is a fictitious variable of $f$, then for any subset $S$ of the variables of $f$, such that $y \notin S$, and for any real-valued convex function $\rho$ defined on $[0,1]$,*

$$E_{\rho,f}(S) = E_{\rho,f}(S \cup \{y\}). \tag{4.1}$$

*Conversely, if $y$ is a nonfictitious variable of $f$, then there exists a subset $S$ of the variables of $f$, such that $y \notin S$ and (4.1) does not hold, i.e., for any relevance function $\rho$, $E_{\rho,f}(S) < E_{\rho,f}(S \cup \{y\})$.*

COROLLARY 4.7. *Let $f$ be a Boolean function and $S$ a set of fictitious variables of $f$. Then for any real-valued convex function $\rho$ defined on $[0,1]$, $E_{\rho,f}(S) = E_{\rho,f}(\emptyset)$.*

LEMMA 4.8. *A Boolean function $f$ of $n$ variables is a constant if and only if $\emptyset$ is the minimum support set of $f$.*

Lemma 4.8 characterizes those Boolean functions whose minimum support set has the smallest possible cardinality (0). The opposite extreme case consists of those Boolean functions whose minimum support set has the largest possible cardinality ($n$). For this, consider the Boolean function $h$ of $n$ variables with $h(\alpha) = 0$ if and only if $\sum_{i=1}^{n} \alpha_i$ is even, where $\alpha \in \{0,1\}^n$. This function and its complement $h'$ defined for $\alpha \in \{0,1\}^n$ by $h'(\alpha) = 1 - h(\alpha)$ are called the (two) *parity functions*. Obviously, no variable is fictitious for either of the two parity functions, and therefore, the minimum support set of each of them is the set $N$. The parity functions are not the only Boolean functions with this property. For example, the minimum support set of the function defined by $f(1,1,\ldots,1) = 1$ and $f(\alpha) = 0$ for all $\alpha \in \{0,1\}^n \setminus \{(1,1,\ldots,1)\}$ is also $N$.

Lemma 4.2 shows that the range of the relevance function for a Boolean function is between 0 and 1. The next two lemmas explore the two extreme cases.

LEMMA 4.9. *Let $f$ be a Boolean function of $n$ variables. Then the following are equivalent.*

(a) *$f$ is a constant function.*
(b) *For each real-valued convex function $\rho$ defined on $[0,1]$, $E_{\rho,f}(S) = E_{\rho,f}(\emptyset)$ for every subset $S \subseteq N$.*
(c) *For some relevance function $\rho$, $E_{\rho,f}(\emptyset) = 1$.*

LEMMA 4.10. *Let $f$ be a Boolean function of $n$ variables. Then the following are equivalent.*

(a) *$f$ is a parity function.*
(b) *For each real-valued convex function $\rho$ defined on $[0,1]$, $E_{\rho,f}(S) = E_{\rho,f}(\emptyset) = \rho(\frac{1}{2})$ for every proper subset $S$ of $N$.*
(c) *For some relevance function $\rho$, $E_{\rho,f}(S) = 0$ for every subset $S$ of $N$ containing $n-1$ variables.*

Let $f$ be a Boolean function of $n$ variables. A subset $S$ of $N$ is called an *implicative set* of $f$ if for some $\alpha \in \{0,1\}^{|S|}$, $f(x_S = \alpha)$ is either identically 1 or identically 0 (in the former case, the partial assignment $x_S = \alpha$ is called an *implicant* of $f$, and in the latter case, it is called an *implicate* of $f$). For example, for the Boolean function

$f(x_1, x_2, x_3, x_4) = x_1x_2 \vee x_3x_4$ the sets $S \equiv \{1, 2\}$ and $T \equiv \{2, 4\}$ are implicative, since $f[x_S = (1, 1)]$ is identically 1, and $f[x_T = (0, 0)]$ is identically 0. The next lemma shows that an implicative set of variables of a Boolean function cannot be "totally irrelevant."

LEMMA 4.11. *Let $f$ be a Boolean function on $n$ variables, let $S$ be an implicative set of $f$, and let $\rho$ be an arbitrary relevance function. Then $E_{\rho,f}(S) \geq 2^{-|S|}$.*

One may conjecture that implicative sets have higher relevance than nonimplicative ones. However, this need not be the case. Indeed, let $f$ be the positive Boolean function of four variables given by

$$f(x) = \begin{cases} 0 & \text{if } (x_1, x_2, x_3) = (0, 0, 0), \\ 1 & \text{if } (x_1, x_2, x_3) = (1, 1, 1), \\ 0 & \text{if } (x_1, x_2, x_3) \notin \{(0, 0, 0), (1, 1, 1)\} \text{ and } x_4 = 0, \\ 1 & \text{if } (x_1, x_2, x_3) \notin \{(0, 0, 0), (1, 1, 1)\} \text{ and } x_4 = 1. \end{cases}$$

Then $S \equiv \{1, 2, 3\}$ is an implicative set because $f[x_S = (0, 0, 0)]$ is identically 0 (and also, because $f[x_S = (1, 1, 1)]$ is identically 1); however, its complement $S^c \equiv \{4\}$ is not an implicative set. Since $|f(x_4 = 0)| = 1$ and $|f(x_4 = 1)| = 7$, for the linear relevance function $\rho$ we have $E_{\rho,f}(S^c) = \frac{3}{4}$. On the other hand,

$$|f(x_S = \alpha)| = \begin{cases} 0 & \text{if } \alpha = (0, 0, 0), \\ 2 & \text{if } \alpha = (1, 1, 1), \\ 1 & \text{otherwise.} \end{cases}$$

Therefore, for any relevance function $\rho$ we have $E_{\rho,f}(S) = \frac{1}{4}$. Thus if $\rho$ is the linear relevance function, then $E_{\rho,f}(S) = \frac{1}{4} < \frac{3}{4} = E_{\rho,f}(S^c)$.

**Acknowledgments.** The authors express their gratitude to an anonymous referee, whose many useful suggestions helped substantially improve this paper.

## REFERENCES

[1] F. J. BANZAF III, *Weighted voting doesn't work: A mathematical analysis*, Rutgers Law Review, 19 (1965), pp. 317–343.

[2] M. BEN ORR AND N. LINIAL, *Collective coin flipping*, in Randomness and Computation, Advances in Computing Research 5, S. Micali, ed., JAI Press, Greenwich, CT, 1989, pp. 91–125.

[3] M. BEN ORR, N. LINIAL AND M. SAKS, *Collective Coin Flipping and Other Models of Imperfect Randomness*, RUTCOR Research Report RRR 44–87, Rutgers University, New Brunswick, NJ, 1987.

[4] L. BREIMAN, J. H. FRIEDMAN, R. A. OLSHEN, AND C. J. STONE, *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA, 1984.

[5] C. K. CHOW, *On the characterization of threshold functions*, in Proceedings of the Second Annual Symposium and Papers from the First Annual Symposium on Switching Circuit Theory and Logical Design, American Institute of Electrical Engineers, Detroit, MI, 1961, pp. 34–38.

[6] C. K. CHOW, *Boolean functions realizable with single threshold devices*, Proc. IRE, 49 (1961), pp. 370–371.

[7] P. DUBEY AND L. S. SHAPLEY, *Mathematical properties of the Banzaf power index*, Math. Oper. Res., 4 (1979), pp. 99–131.

[8] P. ERDÖS, M. HERZOG, AND J. SCHÖNHEIM, *An extremal problem on the set of noncoprime divisors of a number*, Israel J. Math., 8 (1970), pp. 408–412.

[9] D. Felsenthal and M. Machover, *Postulates and paradoxes of relative voting power—A critical re-appraisal*, Theory and Decision, 38 (1995), pp. 195–229.
[10] P. L. Hammer, A. Kogan, and U. G. Rothblum, *Relevance Functions and Evaluations of Sets of Variables of a Boolean Function*, RUTCOR Research Report 4–96, Rutgers University, New Brunswick, NJ, 1996.
[11] J. Kahn, G. Kalai, and N. Linial, *The influence of variables on Boolean functions*, in Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, White Plains, NY, 1988, pp. 68–80.
[12] L. Lovász, *Combinatorial Problems and Exercises*, North-Holland, Amsterdam, 1979.
[13] A. W. Marshall and I. Olkin, *Inequalities, Theory of Majorization and Its Applications*, Academic Press, New York, 1979.
[14] S. Muroga, *Threshold Logic and Its Applications*, Wiley, New York, 1971.
[15] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan-Kaufmann, San Mateo, CA, 1993.
[16] L. S. Shapley and M. Shubik, *A method for evaluating the distribution of power in a committee system*, Amer. Polit. Sci. Rev., 48 (1954), pp. 787–792.
[17] A. D. Taylor and W. S. Zwicker, *Interval measures of power*, Math. Social Sci., 33 (1997), pp. 23–74.
[18] R. O. Winder, *Threshold Logic*, Ph.D. dissertation, Department of Mathematics, Princeton University, Princeton, NJ, 1962.
[19] R. O. Winder, *The fundamentals of threshold logic*, in Applied Automata Theory, J. Tou, ed., Academic Press, New York, 1968, pp. 236–318.
[20] R. O. Winder, *Chow parameters in threshold logic*, J. ACM, 18 (1971), pp. 265–289.

SIAM J. Discrete Math.
Vol. 13, No. 3, pp. 313–323

# A TIGHT BOUND ON THE IRREGULARITY STRENGTH OF GRAPHS[*]

TILL NIERHOFF[†]

**Abstract.** An assignment of positive integer weights to the edges of a simple graph $G$ is called *irregular* if the weighted degrees of the vertices are different. The irregularity strength $s(G)$ is the maximal weight, minimized over all irregular assignments. It is set to $\infty$ if no such assignment is possible. Let $G \neq K_3$ be a graph on $n$ vertices, with $s(G) < \infty$. Aigner and Triesch [*SIAM J. Discrete Math*, 3 (1990), pp. 439–449] used the congruence method to construct irregular assignments, showing $s(G) \leq n-1$ if $G$ is connected and $s(G) \leq n+1$ in general. We refine the congruence method in the disconnected case and show that $s(G) \leq n-1$ holds for all graphs with $s(G)$ finite, except for $K_3$. This is tight and settles a conjecture of Aigner and Triesch.

**Key words.** irregular assignments, irregularity strength, congruence method

**AMS subject classification.** 05C78

**PII.** S0895480196314291

## 1. Introduction.

The analysis of the irregularity strength of graphs was initiated by Chartrand et al. [2]. For a survey see the article by Lehel [4] and also the web page [3]. The assertion proved here was conjectured by Aigner and Triesch in [1], while the question was raised in a less affirmative way in [4] earlier.

Definition 1.1. *Let $G = (V, E)$ be a graph and let $x : E \to \{1, \dots, r\}$ be an assignment of weights to the edges. Denote by $x^* : V \to \mathbb{N}$ the corresponding* weighted degree map*: $x^*(v) := \sum_{e \in E: v \in e} x(e)$. The assignment $x$ is called* irregular *if $x^*$ is injective. The* irregularity strength *of $G$ is defined by*

$$s(G) := \inf\{r \in \mathbb{N} : \textit{ there exists an irregular assignment } x : E \to \{1, \dots, r\}\}.$$

Since any simple graph $G$ has at least two vertices of the same degree, we know that $s(G) \geq 2$. It is clear that $s(G) = \infty$ whenever $G$ contains an isolated edge or more than one isolated vertex. By assigning different powers of 2, say, to the edges of any other graph, we can see that the converse is also true.

The exponential bound on $s(G)$ that can be derived from the assignment in the latter case is by far too wasteful. Let $n$ be the number of vertices of $G$ and assume that $s(G) < \infty$. Already in [2] it was shown that $s(G) \leq 2n-3$. Aigner and Triesch [1] described the so-called congruence method to construct irregular assignments. They used it to show that $s(G) \leq n+1$ and that if $G \neq K_3$ is connected, then even $s(G) \leq n-1$.

The relation between connectedness of $G$ and the value of $s(G)$ seemed to be due to proof technical reasons. Therefore, Aigner and Triesch conjectured that "$s(G) \leq n-1$ for any $n$-vertex graph $G$ with $s(G) < \infty$, except for $K_3$." In this article we refine the congruence method to verify that conjecture.

Theorem 1.2. *Let $G \neq K_3$ be a graph on $n$ vertices without isolated edges and with at most one isolated vertex. Then $s(G) \leq n-1$.*

[*]Received by the editors December 30, 1996; accepted for publication (in revised form) December 6, 1999; published electronically May 15, 2000.
http://www.siam.org/journals/sidma/13-3/31429.html

[†]Humboldt-Universität zu Berlin, Institut für Informatik, D-10099 Berlin, Germany (nierhoff@informatik.hu-berlin.de).

Notice that the bound is tight as the stars $K_{1,n-1}$, $n \geq 3$, or the union of two paths on three vertices, $2 \cdot P_3$, show. The proof in section 3 consists of a general case (Condition 3.2) and some special cases (the other conditions), proven by the congruence method in several variants. Also, in its existing form, the congruence method is not appropriate for tight results on graphs with an odd number of vertices. The proof is therefore preceded by section 2, where the congruence method is provided with a versatile interface (Lemma 2.13 and Remark 2.14) and refined to cope with odd vertex sets.

**2. Refined congruence method.** Let $\mathbb{Z}_r$, $r \in \mathbb{N}$, be the additive group modulo $r$. The congruence method allows us to construct assignments for a graph such that the weighted degrees lie in certain given elements of $\mathbb{Z}_r$. Its core is the following lemma, which was originally proved using linear algebra. We give an alternative proof that provides a simple construction.

LEMMA 2.1 (Aigner and Triesch [1]). *Let $G = (V, E')$ be a graph and let $F = (V, E)$, $E \subset E'$ be a forest of spanning trees of the components of $G$. Let $y : V \rightarrow \mathbb{Z}_r$ be such that for every tree in $F$ with color classes $R$ and $S$*

$$\sum_{v \in R} y(v) \equiv \sum_{w \in S} y(w) \pmod{r}. \tag{1}$$

*Then there exists an assignment $x : E' \rightarrow \{1, \ldots, r\}$ for $G$ such that*

$$x^*(v) \equiv y(v) \pmod{r}$$

*holds for every vertex $v \in V$. The assignment $x$ is said to* realize *$y$.*

*Proof.* Assume without loss of generality that $F$ is a tree. Construct $x$ from $y$ using a depth-first search:

Fix a start vertex $u$ in color class $R$, say, and descend to every neighbor, recursively. On the way back from vertex $w$ to $v$, say, $x(e)$ has been defined for all edges $e$ incident to $w$ but $\{v, w\}$. Let $x^*(w)$ be their sum. Then define $x(\{v, w\}) := y(w) - x^*(w) \pmod{r}$, and update $x^*(w)$ properly.

For all $w \in V \setminus \{u\}$, $x^*(w) \equiv y(w) \pmod{r}$ holds in the end. Use $\sum_{v \in R} x^*(v) = \sum_{e \in E} x(e) = \sum_{w \in S} x^*(w)$, to compute

$$x^*(u) = \sum_{w \in S} x^*(w) - \sum_{v \in R \setminus \{u\}} x^*(v) \equiv \sum_{w \in S} y(w) - \sum_{v \in R \setminus \{u\}} y(v) \pmod{r}.$$

Therefore $x^*(u) \equiv y(u) \pmod{r}$ iff (1) holds. Finally, we assign the weight $r$ to the edges in $E' \setminus E$. This preserves the congruences $x^*(v) \equiv y(v) \pmod{r}$, $v \in V$. ◻

Observe that if $y(v) \neq y(w)$ for $v, w \in V$, then $x^*(v) \neq x^*(w)$ too. The key idea of the congruence method is to find a $y$ that obeys (1) and uses only few congruence classes of $\mathbb{Z}_r$ more than once.

COROLLARY 2.2. *Let $G$, $F$, and $y$ be as in Lemma 2.1. If it is impossible that $x^*(v) = x^*(w)$ for every two vertices $v, w \in V$ with $y(v) = y(w)$, then $s(G) \leq r$.* ◻

Since in Theorem 1.2 the aim is to show $s(G) \leq n - 1$, we shall decompose $\mathbb{Z}_{n-1}$ into pairs and triples that can be used to define $y$ incrementally, preserving (1) in each step.

DEFINITION 2.3. *Let $r \in \mathbb{N}$. A pair $(\kappa, r - \kappa)$, where $\kappa \in \{1, \ldots, \lfloor \frac{r-1}{2} \rfloor\}$, is called an $r$-*complementary pair*, or, shorter, a* c-pair *if the choice of $r$ is clear from* *the context. A triple* $(\alpha, \beta, \gamma) \in \{1, \dots, r\}^3$ *is called an* $r$-complementary triple*, respectively, a* c-triple *if* $\alpha + \beta \equiv \gamma \pmod r$.

We do not distinguish explicitly between natural numbers and congruence classes (mod $r$) in what follows. For instance, $\kappa$ may denote both an element of $\mathbb{Z}_r$ and its representative in $\{1, \dots, r\}$. With this in mind, we call $\kappa \in \mathbb{Z}_r$ *small* if $\kappa \le \left\lfloor \frac{r-1}{2} \right\rfloor$.

The following lemma by Skolem and O'Keefe [8, 7] is used to decompose $\mathbb{Z}_r$ into almost every desirable number of c-pairs and c-triples.

LEMMA 2.4 (Skolem, O'Keefe [8, 7]). *Let* $t \in \mathbb{N}$. *For* $z = 3t$ *or* $z = 3t + 1$ *the set* $\{1, \dots, 3t + 1\} \setminus \{z\}$ *can be partitioned into* $t$ *triples* $(\alpha_i, \beta_i, \gamma_i)$*,* $i \in \{1, \dots, t\}$ *such that* $\alpha_i + \beta_i = \gamma_i$ *for all* $i$. ∎

COROLLARY 2.5. *Let* $n, t \in \mathbb{N}$*,* $r := n - 1$*, and assume that* $\left\lfloor \frac{r-1}{2} \right\rfloor \ge 3 \left\lfloor \frac{t}{2} \right\rfloor + 1$. *Let* $p := \left\lfloor \frac{r-1}{2} \right\rfloor - \left\lfloor \frac{3t}{2} \right\rfloor$. *Then* $\mathbb{Z}_r$ *can be partitioned into* $t$ *c-triples,* $p$ *c-pairs, and a set* $M \subset \{\frac{r}{2}, r\}$*, where* $\frac{r}{2} \in M$ *iff* $r$ *is even and* $r \in M$ *iff* $t$ *is even. If* $t \ge 1$*, then at least one of the c-triples contains a small congruence class in the first place.*

*Proof.* Notice that $(1, r-1), \dots, (\left\lfloor \frac{r-1}{2} \right\rfloor, \left\lceil \frac{r+1}{2} \right\rceil)$ is a partition of $\mathbb{Z}_r \setminus \{\frac{r}{2}, r\}$ into $\left\lfloor \frac{r-1}{2} \right\rfloor$ c-pairs. The c-triples are constructed out of $\left\lfloor \frac{3t}{2} \right\rfloor$ of these c-pairs:

Pack triples $(\alpha_i, \beta_i, \gamma_i)$, $i \in \{1, \dots, \left\lfloor \frac{t}{2} \right\rfloor\}$ into $\{1, \dots, 3\left\lfloor \frac{t}{2} \right\rfloor + 1\}$ as in Lemma 2.4. These triples are c-triples and, since $3\left\lfloor \frac{t}{2} \right\rfloor + 1 \le \left\lfloor \frac{r-1}{2} \right\rfloor$, they consist of small congruence classes only. The next $\left\lfloor \frac{t}{2} \right\rfloor$ c-triples are chosen as $(r - \alpha_i, r - \beta_i, r - \gamma_i)$, $i \in \{1, \dots, \left\lfloor \frac{t}{2} \right\rfloor\}$, using the nonsmall rest of the used c-pairs. In this way, $2\left\lfloor \frac{t}{2} \right\rfloor$ c-triples have been constructed out of $3\left\lfloor \frac{t}{2} \right\rfloor$ c-pairs. If $t$ is odd, the last c-triple is chosen as $(\kappa, r - \kappa, r)$, where $(\kappa, r - \kappa)$ is one of the $\left\lfloor \frac{r-1}{2} \right\rfloor - 3\left\lfloor \frac{t}{2} \right\rfloor \ge 1$ remaining c-pairs.

Since only the second portion of c-triples does not contain a small congruence class in the first place, the corollary follows. ∎

In the rest of this section the refined congruence method is described. It uses c-pairs and c-triples to find irregular assignments via Corollary 2.2.

DEFINITION 2.6 (block, feasibility). *Let* $H = (V, E')$ *be a component of a graph. A* block *for* $H$ *is a spanning tree* $T = (V, E)$ *of* $H$ *and a vertex subset* $B \subset V$*, called the* blocked vertices.

*Let* $R$ *and* $S$ *be the color classes of* $T$ *and let* $r' := |R \setminus B|$ *and* $s' := |S \setminus B|$. *Call* $(T, B)$ *a* feasible block *for* $H$ *if* $r'$ *is even unless* $s' \ge 2$ *and vice versa. Let*

$$t(H, T, B) := \begin{cases} 0 & \textit{if } r' \textit{ and } s' \textit{ are even,} \\ 2 & \textit{if } r' \textit{ and } s' \textit{ are odd,} \\ 1 & \textit{otherwise,} \end{cases}$$

*and* $p(H, T, B) := \frac{r' + s' - 3t(H,T,B)}{2}$.

LEMMA 2.7 (filling with block). *Let* $H = (V, E')$ *be a component of a graph,* $(T, B)$ *a feasible block for* $H$*, and* $r \in \mathbb{N}$. *Given a disjoint union* $M \subset \mathbb{Z}_r$ *of* $t := t(H, T, B)$ $r$*-complementary triples and* $p := p(H, T, B)$ $r$*-complementary pairs, there is an injective map* $y_{\bar{B}} : V \setminus B \to M$ *for which* (1) *holds.*

*If* $t = 0$ *or* $M$ *contains a c-triple with a small congruence class in the first place, and there are two unblocked vertices* $u, v$*, say, in the same color class of* $T$*, then one may furthermore assume that* $y(u)$ *is small.*

*Proof.* Equation (1) holds for any map that is assembled from partial maps for which it holds. In the following, partial maps for subsets of two or three vertices of $V \setminus B$ are given such that (1) holds. The map $y_{\bar{B}}$ is then assembled from these partial maps.

Let $R$, $S$, $r'$, and $s'$ be defined as in Definition 2.6. If one of $r'$ and $s'$ is odd, say $r'$, then $s' \geq 2$ and $M$ contains a c-triple $(\alpha, \beta, \gamma)$. Let $u \in R \setminus B$ and $w_1, w_2 \in S \setminus B$. Let $y_{\bar{B}}(u) := \gamma$, $y_{\bar{B}}(w_1) := \alpha$, and $y_{\bar{B}}(w_2) := \beta$. Since the triple is $r$-complementary, (1) holds for this part of $y_{\bar{B}}$. If $s'$ is also odd, then there are another vertex in $S \setminus B$ and two other vertices in $R \setminus B$. Perform the same with these vertices and the other c-triple in $M$.

As a result, in each color class there is an even number of unblocked vertices left. Let $v_1$ and $v_2$ be vertices in one color class. Choose a c-pair $(\kappa, r-\kappa) \subset M$ that has not been used before and let $y_{\bar{B}}(v_1) := \kappa$ and $y_{\bar{B}}(v_2) := r - \kappa$. This step also preserves (1) and can be repeated until $y_{\bar{B}}$ has been extended to $V \setminus B$.

Notice that it is irrelevant how the c-pairs and c-triples are distributed. Assume that $u \neq v \in V \setminus B$ are in the same color class of $T$. If $t = 0$, then $M$ contains only c-pairs and the first step might be $y_{\bar{B}}(u) := \kappa$ and $y_{\bar{B}}(v) := r - \kappa$, where $(\kappa, r-\kappa) \subset M$. If $M$ contains a c-triple $(\alpha, \beta, \gamma)$ with small $\alpha$, then the first step might be $y_{\bar{B}}(u) := \alpha$, $y_{\bar{B}}(v) := \beta$, and $y_{\bar{B}}(w) := \gamma$ for some $w$ in the other color class. In each case $y_{\bar{B}}(u)$ is small.

It is easy to verify that this procedure needs precisely $t$ c-triples and $p$ c-pairs and that $y_{\bar{B}}$ is injective. ◻

In the following we refer to this process of assigning sufficiently many c-pairs and c-triples to subsets of color classes by saying that they are *spread over* the respective vertices.

COROLLARY 2.8 (filling without block). *Let $H = (V, E')$ be a $k$-vertex component of a graph and assume $k \geq 3$, $r \in \mathbb{N}$. Let*

$$t := t(H) := \begin{cases} 1 & \text{if } k \text{ is odd,} \\ 0 & \text{if } H \text{ has a spanning tree with even color classes,} \\ 0 & \text{if } k \text{ is even, and } H = K_{1,k-1}, \\ 2 & \text{otherwise.} \end{cases}$$

*Given a disjoint union $M$ of $t$ $r$-complementary triples and $p := p(H) := \frac{k-3t}{2}$ $r$-complementary pairs, there is a map $y : V \rightarrow M$ for which* (1) *holds and every realization of which is irregular.*

*Proof.* Let $B := \emptyset$. In the following cases the block $(T, B)$ is feasible and $t = t(H, T, B)$ and $p = p(H, T, B)$:

(i) if $k$ is odd and $T$ is any spanning tree of $H$,

(ii) if $k$ is even and $H$ has a spanning tree $T$ with even color classes,

(iii) if $k$ is even and $H$ has a spanning tree $T$ with at least three vertices in each color class.

Therefore Lemma 2.7 gives an injective map $y : V \rightarrow M$ for which (1) holds and Corollary 2.2 implies the result.

Otherwise $k$ is even and $T := K_{1,k-1}$ is a spanning tree, but $H$ has no spanning tree with even color classes. Then $H$ has no edge between the leaves of $T$ and thus $H = T = K_{1,k-1}$. Let $c$ be the center vertex and $l$ be one of the leaves. Choose $B := \{c, l\}$. Then $p(H, T, B) = p - 1$ and Lemma 2.7 gives an injective map $y : V \backslash B \rightarrow M \backslash \{\kappa, r-\kappa\}$, where $(\kappa, r-\kappa)$ is one of the c-pairs of $M$. Set $y(c) := y(l) := \kappa$. Then $y$ satisfies (1). To apply Corollary 2.2 it remains to verify that $x^*(c) \neq x^*(l)$ for any $x$ realizing $y$.

Notice that in general for a leaf $l$ and its neighbor $c$

$$x^*(l) = x(\{l,c\}) < \sum_{e\in E: c\in e} x(e) = x^*(c), \tag{2}$$

as long as $\{l,c\}$ is not an isolated edge. ◻

DEFINITION 2.9 (blocked graphs, notation). *Let $G = (V,E)$ be an $n$-vertex graph with $m'$ components, each of order at least three. Call $G$ a* blocked graph *if there are blocks for $m$, say, of the components. Define the following notation for blocked graphs:*

*Set $r := n - 1$. Denote by $H_i$, $i \in \{1,\dots,m'\}$ the components, where $H_i$, $i \in \{1,\dots,m\}$ are those with blocks. Denote these blocks by $(T_i, B_i)$ and let $p_i := p(H_i,T_i,B_i)$ and $t_i := t(H_i,T_i,B_i)$. For $m < i \le m'$ let $p_i := p(H_i)$ and $t_i := t(H_i)$.*

*Call $B := \bigcup_{i=1}^m B_i$ the* block of $G$. *Let $b := |B|$, $p := \sum_{i=1}^{m'} p_i$, $t := \sum_{i=1}^{m'} t_i$, and $p_B := \lfloor \frac{b}{2} \rfloor - 1$.*

DEFINITION 2.10 (good block). *Let $G$ be a blocked graph and use notation of Definition 2.9. The block $B$ is called a* good block *for $G$ if the $(T_i,B_i)$ are feasible,*

$$b \ge \begin{cases} 4 & \text{if } n \text{ is even and } p = 0,\\ 2 & \text{otherwise,}\end{cases} \tag{3}$$

*and $b$ is odd only if $n$ is odd and $b \ge 5$.*

DEFINITION 2.11 (block assignment scheme (b.a.s.)). *Let $G$ be a blocked graph and use notation of Definition 2.9. Let $M := \{\kappa_1, \bar\kappa_1, \dots, \kappa_{p_B}, \bar\kappa_{p_B}\} \cup M'$, where $M' \subset \{\frac{r}{2}, r\}$ such that $r \in M'$ iff $n$ is even or $b$ is odd and $\frac{r}{2} \in M'$ iff $n$ is odd. The $\kappa_i, \bar\kappa_i$ are literals.*

*An* initialization *of $M$ is a replacement of the $\kappa_i, \bar\kappa_i$ by $p_B$ pairwise disjoint $r$-complementary pairs such that $\kappa_1 < \cdots < \kappa_{p_B}$ and $\bar\kappa_i = r - \kappa_i$.*

*Let $y_B : B \to M$ and assume that (1) holds for every initialization of $y_B$, restricted to $B_i$, $i \in \{1,\dots,m\}$. Let $y : V \to \mathbb{Z}_r$ such that $y|_B$ is an initialization of $y_B$, and let $x$ be a realization of $y$. If it is impossible that $x^*(v) = x^*(w)$ whenever $y(v) = y(w)$, then $y_B$ is called a* b.a.s.

*The property that $x^*(v) \ne x^*(w)$ if $y(v) = y(w)$ may be conditional on the fact that in one of the blocked components, say $H_1$, $y(u)$ is small, where $u$ belongs to a color class of $T_1$ that contains at least two unblocked vertices. In this case $y_B$ is called* conditional.

PROPOSITION 2.12. *Let $G$ be a blocked graph and use notation of Definition 2.9. If $B$ is a good block, then $\mathbb{Z}_r$ can be decomposed into sets $M_i$, $i \in \{1,\dots,m'\}$, $M'$ as in Definition 2.11, and $p_B$ extra c-pairs. The $M_i$ are disjoint unions of $t_i$ c-triples and $p_i$ c-pairs, and if $t_1 \ge 1$, then $M_1$ contains a c-triple with a small first place.*

*Proof.* Observe that $n = 3t + 2p + b$. Assume first that $b$ is even. Then $\lfloor \frac{r-1}{2} \rfloor = \lfloor \frac{3t+2p+b-2}{2} \rfloor = \lfloor \frac{3t}{2} \rfloor + \frac{2p+b}{2} - 1$ and $t \equiv n \pmod 2$. If $n$ is even, then $\lfloor \frac{3t}{2} \rfloor = 3 \lfloor \frac{t}{2} \rfloor$ and, by (3), $\frac{2p+b}{2} \ge 2$. In total, $\lfloor \frac{r-1}{2} \rfloor \ge 3 \lfloor \frac{t}{2} \rfloor + 1$. If $n$ is odd, then $\lfloor \frac{3t}{2} \rfloor = 3 \lfloor \frac{t}{2} \rfloor + 1$ and, by (3), $\frac{2p+b}{2} \ge 1$. So, also in this case $\lfloor \frac{r-1}{2} \rfloor \ge 3 \lfloor \frac{t}{2} \rfloor + 1$. Therefore, by Corollary 2.5, $\mathbb{Z}_r$ can be decomposed into $t$ c-triples, $\lfloor \frac{r-1}{2} \rfloor - \lfloor \frac{3t}{2} \rfloor = p + p_B$ c-pairs, and $M' \subset \{\frac{r}{2}, r\}$ as in Definition 2.11.

Assume now that $b$ is odd. Then, by assumption, $n$ is odd, $t$ is even, and $b \ge 5$. Let $b' := b - 3$ and $t' := t + 1$. Perform the same as in the case where $b$ is even, but with $b'$ and $t'$ instead of $b$ and $t$. Then $\mathbb{Z}_r$ is decomposed into $p + \frac{b'}{2} - 1$ c-pairs, $t'$ c-triples, and $M' = \{\frac{r}{2}\}$. Since $t'$ is odd, one of the c-triples is of the form $(\kappa, r - \kappa, r)$

(cf. the proof of Corollary 2.5). Split up this c-triple and the result is a decomposition of $\mathbb{Z}_r$ into $t'-1=t$ c-triples, $p+\frac{b'}{2}-1+1=p+p_B$ c-pairs, and $M'=\{\frac{r}{2},r\}$, as required.

Since $p=\sum_i p_i$ and $t=\sum_i t_i$ the c-pairs and c-triples just need to be distributed to the $M_i$. If $t_1\geq 1$, choose one c-triple for $M_1$ with a small first place. ☐

LEMMA 2.13 (refined congruence method). *Let $G=(V,E)$ be a blocked graph with a good block. Use the notation of Definition* 2.11. *If there is a block assignment scheme $y_B : B \ \rightarrow\ M$, then $s(G)\leq r=n-1$.*

*Proof.* Decompose $\mathbb{Z}_r$ as in Proposition 2.12. Use the $p_B$ extra c-pairs to instanciate $M$. For $i\leq m$, apply Lemma 2.7 to $H_i$ with $(T_i,B_i)$ and $M_i$. If $y_B$ is conditional, make sure that $y(u)$ is small. For $m<i\leq m'$, apply Corollary 2.8 to $H_i$ with $M_i$. This produces $y_i : V_i\setminus B \ \rightarrow\ M_i$ for all $i$. Let $y_{\bar{B}} : V\setminus B \ \rightarrow\ \bigcup_i M_i$ be their union.

Since $y_B$ is a b.a.s., (1) holds for the union $y$ of $y_{\bar{B}}$ with the initialization of $y_B$. Let $x$ be the realization of $y$. Then $x^*$ is injective on each of the $V_i\setminus B_i$ and $B$, by Lemma 2.7, by Corollary 2.8, and by the fact that $y_B$ is a b.a.s. Since the $y$-images of these sets are pairwise disjoint, $x$ is irregular. ☐

*Remark* 2.14. Let $H=(V,E)$ be a component of a graph $G$. If $B:=V$ is a good block[1] and there exists an irregular assignment of $H$ such that the weighted degrees are contained in an arbitrary initialization of $M$ (notation as in Lemma 2.13), then $s(G)\leq n-1$.

To see this, the proof of Lemma 2.13 needs to be varied only slightly. In this case $H$ is blocked and $y_{\bar{B}} : V\setminus B \ \rightarrow\ \mathbb{Z}_r\setminus M$ is constructed as in Lemma 2.13. Since $B$ is the vertex set of a whole component, by Corollary 2.8 there is a realization of $y_{\bar{B}}$. Together with the irregular assignment for $H$ this is an irregular assignment for $G$ and thus $s(G)\leq n-1$.

**3. Proof of Theorem 1.2.** The proof of Theorem 1.2 consists of a case analysis that is organized as follows. Assume that $G$ is an $n$-vertex graph without isolated edges and with at most one isolated vertex, but $s(G)\geq n$. Let $r:=n-1$. Several arguments impose conditions on $G$, which in the end imply that $G=K_3$. The first of those conditions holds without loss of generality: if $G$ contained an isolated vertex $v$, then every irregular assignment of $G-v$ would also be one of $G$.

CONDITION 3.1. *Every component of $G$ has at least three vertices.* ☐

CONDITION 3.2. *Suppose that $G$ contains a $k$-vertex component $H\neq C_k$ with $k\geq 5$. Then there are a spanning tree $T$ of $H$, two vertices $u$, $v$ in different color classes of $T$, and a b.a.s. $y_B$ for the block $B:=\{u,v\}$.*

*Proof.* The spanning tree $T$ and $u,v\in V$ are given for different situations. Let $\rho:=\frac{r}{2}$ if $n$ is odd and $\rho:=r$ otherwise. Then $M=\{\rho\}$ in Definition 2.11 and $y_B$ is defined by $y_B(u):=y_B(v):=\rho$. Trivially, (1) holds for $y_B$. It remains to check in each case that

$$x^*(u)\neq x^*(v) \tag{4}$$

for any $x$ realizing a $y$ with $y|_B=y_B$.

1. If $H$ contains a leaf, then let $T$ be any spanning tree and let $u$ and $v$ be the leaf and its neighbor. Here, (4) holds for the same reason as (2). Therefore, one may assume that $H$ contains no leaves.

[1]This is abuse of notation, since no spanning tree of $H$ is involved. However, blocking all vertices of a component is always feasible. Whether a block is good depends mainly on the other components.

2. If there are two adjacent noncut vertices $u$ and $v$ with $d(u) > d(v)$, say, then choose a spanning tree of $H - u$. Attach $u$ to this tree as a leaf adjacent to $v$, thus obtaining $T$. There are at least $d(v)$ nontree edges adjacent to $u$. Since all of those are assigned the weight $r$, $u$ gets a larger weighted degree than $v$. This verifies (4). Assume therefore that all noncut vertices in the same 2-connected component have the same degree.

3. If there are cut vertices in $H$, then there exist two 2-connected components $B_1$ and $B_2$, say, which contain only one cut vertex. Because of Case 1, both $B_1$ and $B_2$ have at least two noncut vertices.

Let $d_1$ and $d_2$ be the degrees of the noncut vertices in $B_1$ and $B_2$, respectively, where $d_1 \le d_2$, say. Let $w$ be the vertex that cuts off $B_2$. Now take two adjacent vertices $v$ and $v'$ out of $B_1$ and let $T'$ be a spanning tree of $H - B_2$, with $v'$ as a leaf and $v$ as neighbor of $v'$. Add $w$ to $T'$ and let $\pi \in \{0,1\}$ denote the parity of the length of the $v$-$w$-path in $T'$.

Since $B_2$ is 2-connected, there are $u \in B_2$ and a spanning tree $T''$ of $B_2$ in which $u$ is a leaf whose distance from $w$ has parity $1 - \pi$. Then let $T$ be $T'$ and $T''$, with their copies of $w$ identified. Notice that $u$ and $v$ are in different color classes of $T$ because the $u$-$v$-path has an odd length.

Since the color class of $v'$ contains at least one further vertex (another neighbor of $v$, for instance), $y(v')$ may be assumed to be small. Then

$$x^*(v) \le y(v') + (d_1 - 1) \cdot r < \rho + (d_2 - 1) \cdot r = x^*(u). \tag{5}$$

Assume therefore that $H$ contains no cut vertices.

4. The remaining case is that $H$ is $d$-regular and 2-connected but not a cycle. Then $H$ contains two vertices $u$ and $w$ with a common neighbor $v$ such that $H - \{u, w\}$ still is connected [5]. Let $T$ be a spanning tree of $H - \{u, w\}$ with $u$ and $w$ attached as leaves to $v$.

Since $v$ has a third neighbor, $y(w)$ may be assumed to be small. A similar calculation as (5) verifies (4) in this last case. □

Call a tree 2-class if one of its color classes contains two vertices and the other an even number which is at least four. If a component of $G$ contained a 2-class spanning tree, then $s(G)$ would be at most $n - 1$.

CONDITION 3.3. *No component of $G$ has a 2-class spanning tree.*

*Proof.* Assume some component $H = (V, E)$ of $G$ has a 2-class spanning tree $T$, with color classes $R$ and $S$. Let $B := V$ and observe that $b \ge 6$ is even. Therefore the block $\{(T, B)\}$ is good for $G$. By Lemma 2.13 and using the notation of Definition 2.11, the condition holds if there exists a b.a.s. $y_B : B \to M$.

Observe that $M$ is the disjoint union of some $\{\rho\}$ and $\frac{b}{2} - 1$ c-pairs. In the following, two vertices $z_1$ and $z_2$ are selected for different situations. Unless stated otherwise, let $y_B : z_1, z_2 \mapsto \rho$ and spread the c-pairs over the rest of the vertices. It is easy to check that $|R \setminus \{z_1, z_2\}|$ and $|S \setminus \{z_1, z_2\}|$ are always even. Since $2\rho \equiv r \pmod{r}$, (1) holds and it remains to check that

$$x^*(z_1) \ne x^*(z_2) \tag{6}$$

must hold for every realization $x$ of $y_B$.

Assume that $R = \{u, v\}$ and $S = \{u_1, \ldots, u_k, w, v_1, \ldots, v_l\}$ as in Figure 1. As $|H|$ is even, $k + l$ is odd. Say $k$ is even and $l$ is odd.

1. Choose $z_1, z_2 \in S \setminus \{w\}$ with $d(z_1) > d(z_2)$ if possible. Then $x^*(z_1) - x^*(z_2) = (d(z_1) - d(z_2))r > 0$. Otherwise, all vertices in $S \setminus \{w\}$ have the same degree $d$ in $H$.

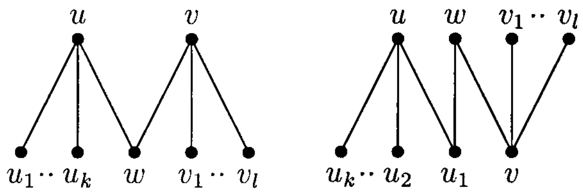


FIG. 1. *A 2-class tree and the modification to be performed in Case 2 of the proof of Condition 3.3.*

2. If one of the $u_i$, say $u_1$, is adjacent to $w$, then modify $T$ as in Figure 1. Let $z_1 := u_1$ and $z_2 := v_1$. The c-pairs can be spread over the other vertices such that $y_B : u_2 \mapsto \kappa_1$; $u \mapsto \kappa_2$; $v \mapsto r - \kappa_1$; $w \mapsto r - \kappa_2$. Then the assignment $x$, as produced by Lemma 2.1, has the following values:

| $e$ | $\{u_2, u\}$ | $\{u, u_1\}$ | $\{u_1, w\}$ | $\{w, v\}$ | $\{v, v_1\}$ |
|---|---|---|---|---|---|
| $x(e)$ | $\kappa_1$ | $\kappa_2 - \kappa_1$ | $\rho - \kappa_2 + \kappa_1$ | $\rho - \kappa_1$ | $\rho$ |

.

Therefore $x^*(u_1) - x^*(v_1) = r > 0$. The case that one of the $v_i$ is adjacent to $w$ can be treated similarly. Assume therefore that $w$ has no neighbor within $S$.

3. If $d = 1$ there are two subcases: $k > 0$ and $k = 0$. In the latter subcase let $z_1 := u$ and $z_2 := v$. Then $x^*(u) = \rho < \rho + r \leq x^*(v)$, since $|S| \geq 4$.

In the subcase where $k > 0$ one proceeds somewhat different. Let $y_B : u, u_1 \mapsto \kappa_1$ and $y_B : v, v_1 \mapsto r - \kappa_1$, and spread the rest of the c-pairs over the other vertices. Then $x^*(u) \neq x^*(u_1)$ and $x^*(v) \neq x^*(v_1)$ by the same argument as for (2). In this case $\rho$ is not used. Assume therefore that $d$ is at least 2.

4. Choose $z_1 := w$ and $z_2 := v_1$. Then, assuming $y_B(u) = \kappa_1$ and $y_B(v) = r - \kappa_1$, Lemma 2.1 produces $x : \{u, w\} \mapsto \kappa_1$; $\{w, v\} \mapsto \rho - \kappa_1$; $\{v, v_1\} \mapsto \rho$. While $v_1$ is incident to $d - 1 \geq 1$ nontree edges, $w$ is only incident to tree edges. Therefore $x^*(w) = \rho < \rho + r \leq x^*(v_1)$. ∎

CONDITION 3.4. *Suppose $H$, $T$, and $B$ are as in Condition 3.2 and let $l = \frac{k-1}{2}$. If $(T, B)$ is not a feasible block for $H$, then $n$ is even and either $H = K_{1,2l}$ or $k = 5$.*

*Proof.* Let $R$, $S$, $r'$, and $s'$ be as in Definition 2.6. If $(T, B)$ is not feasible, then $r' \leq 1$ and $s'$ is odd, say.

If $r' = 1$, then $|R| = 2$ and $|S| \geq 4$ is even. Then $T$ would be 2-class, a contradiction to Condition 3.3. Hence we may assume that $r' = 0$, i.e., $T = K_{1,2l}$.

If $n$ is odd, then $\rho = \frac{r}{2}$. Extend the block $B$ by three vertices of $S$ and define the b.a.s. $y_B$ on these vertices using $r$, $\kappa_1$, and $\bar{\kappa}_1$. Then $s(G) \leq n - 1$ by Lemma 2.13.

Assume that $n$ is even, $H \neq T = K_{1,2l}$ and $l \geq 3$. Distinguish two cases:

1. Two leaves of $T$ have different degree in $H$. Denote these two leaves by $u$ and $v$ and change $B$ to $B := \{u, v\}$. Then $M = M' = \{r\}$ in Definition 2.11 and $y_B \equiv r$ remains a b.a.s., since for any realization $x$, $x^*(u) = d(u) \cdot r \neq d(v) \cdot r = x^*(v)$. The block is feasible, because there is an even number of unblocked leaves, and it is good, because $p \geq p(H) = l - 1 \geq 1$. Therefore $s(G) \leq n - 1$ by Lemma 2.13.

2. All leaves of $T$ have the same degree $d$ in $H$, where $d \geq 2$, since $H \neq T$. Modify $T$ by making one leaf $u$, say, adjacent to another leaf $v$, say. Let $w$ be a third leaf and $c$ be the center vertex of $T$. Then $B := \{c, u, v, w\}$ and $y_B : c \mapsto \kappa_1$; $u \mapsto \bar{\kappa}_1$; $v, w \mapsto r$. This block is feasible for $H$ and good for $G$, and since $x^*(v) = (d-1) \cdot r < dr = x^*(w)$ for any $x$ realizing $y_B$, $s(G) \leq n - 1$ by Lemma 2.13. ∎

CONDITION 3.5. *Suppose $H$, $T$, $B$, and $k$ are as in Condition 3.2. If $(T, B)$ is feasible for $H$, but $\{(T, B)\}$ is not good for $G$, then $n$ is even and $k = 5$.*

*Proof.* Use notation of Definition 2.10. Since $(T, B)$ is feasible and $b$ is even, $\{(T, B)\}$ would be good if (3) held. Thus, $n$ is even and $p = 0$. Since $p(H, T, B) \leq p = 0$, $r' = 3$, and $s' \in \{0, 3\}$ in the notation of Definition 2.6.

If $s' = 0$, then $k = 5$ and the condition holds. If $s' = 3$, then $k = 8$ and each color class of $T$ has four vertices. There exist exactly nine such trees.

Eight of them are presented in Figure 2. If $T$ is one of them, then let $B := V(H)$ and let $u$, $v$, $x_1$, and $x_2$ be as in the description of Figure 2. If $v$ is incident to fewer nontree edges than $u$, then choose the assignment $x_2$, otherwise choose $x_1$. Assign $r$ to the nontree edges. The resulting assignment is irregular and the weighted degrees are contained in $M$ (as in Definition 2.11). Thus $s(G) \leq n - 1$ by Lemma 2.13 and Remark 2.14.

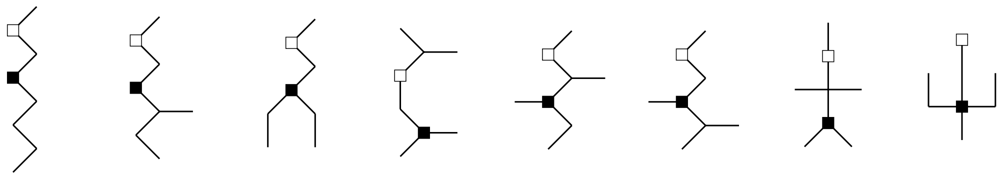


FIG. 2. *Eight trees with four vertices in each color class. Each has a white vertex $u$ and a black vertex $v$. One can check that each tree has two edge assignments $x_1$ and $x_2$, where $x_1^*(u) = r$, $x_1^*(v) = 2r$, and $x_2^*(u) = x_2^*(v) = r$. Both use the congruence class $r$ and three arbitrary c-pairs.*

Assume therefore that $H$ has none of the trees of Figure 2 as a spanning tree and $T$ consists of two inner vertices, each with three adjacent leaves. The color classes of all spanning trees of $H$ contain at least three vertices because otherwise there would be a 2-class spanning tree of $H$, contradicting Condition 3.3. Using this, one can check that $H$ must have at least two leaves $u_1$ and $u_2$ with different neighbors $v_1$ and $v_2$. Choose $B := \{u_1, u_2, v_1, v_2\}$ and let $y_B(u_1) := y_B(v_1) := \kappa_1$ and $y_B(u_2) := y_B(v_2) := \bar{\kappa}_1$. Then $y_B$ is a b.a.s., and therefore $s(G) \leq n - 1$ by Lemma 2.13. ▯

CONDITION 3.6. *If $H = C_k$ is a component of $G$ with $k \geq 5$, then $n$ is even and $k = 5$.*

*Proof.* If $k \geq 7$, then let $T$ be a spanning path of $H$ and let $B = \{v_1, v_2, v_3, v_4\}$ be four subsequent vertices such that $v_1$ is a leaf. This is a feasible block for $H$ and good for $G$. In Definition 2.11, $M = \{\rho, \kappa_1, \bar{\kappa}_1\}$, where $2\rho \equiv r \pmod{r}$.

Let $y_B(v_1) := \rho$, $y_B(v_2) := \kappa_1$, $y_B(v_3) := \rho$, and $y_B(v_4) := \bar{\kappa}_1$. This is a b.a.s., since $x^*(v_1) = r + \rho > \rho = x^*(v_3)$, and thus $s(G) \leq n - 1$ by Lemma 2.13.

For the cases "$k = 6$" and "$k = 5$, $n$ odd," irregular assignments of $H$ whose weighted degrees are contained in the appropriate $M$ are given in Figure 3. Then $s(G) \leq n - 1$ by Lemma 2.13 and Remark 2.14. ▯

CONDITION 3.7. *Every component of $G$ has order at most four.*

*Proof.* Assume that $G$ has a component $H_1$ of order $k \geq 5$. If $H_1$ is not a cycle, then, by Condition 3.2, there is a block $(T, B)$ of that component and an appropriate b.a.s. That block is not feasible for $H_1$ or it is not good for $G$, because otherwise $s(G) \leq n - 1$ by Lemma 2.13. In either case, by Conditions 3.4 and 3.5, $n$ is even and either $H_1 = K_{1,2l}$ or $k = 5$, where $l = \frac{k-1}{2}$. Condition 3.6 implies the same if $H_1$ is a cycle.

Since $k$ is odd but $n$ is even, there must be another odd component $H_2$. If $H_2$ has more than three vertices, then, by the same reasoning as above, $H_2$ is a star with an even number of leaves or has order five. The plan in this situation is to define a feasible block consisting of odd vertex subsets $B_1$ and $B_2$ in $H_1$ and $H_2$, respectively.

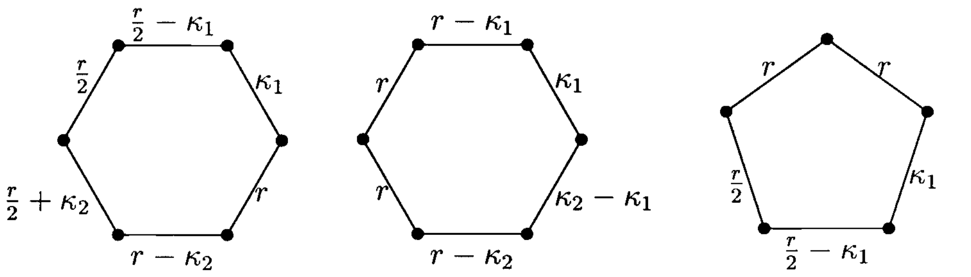


FIG. 3. *Irregular assignments for cycles of length* 6 *and, if n is odd, of length* 5. *The weighted degrees are contained in the appropriate set M.*

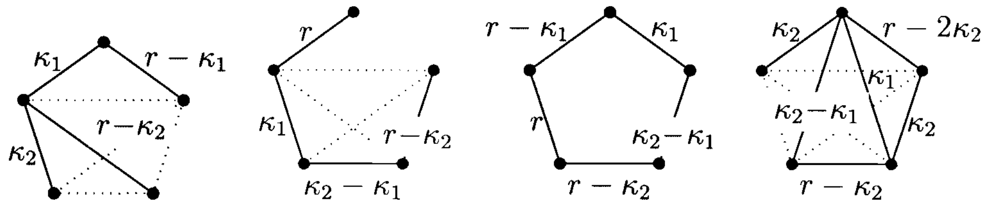


FIG. 4. *Irregular assignments for components of order* 5. *The dotted edges may exist or not, if they do exist they are assigned the value r. The weighted degrees are contained in the appropriate set M. The weighted degree in the congruence class of r is r itself.*

Then the set $M$ consists of the congruence class $r$ and $\frac{|B_1|+|B_2|}{2}-1$ c-pairs. Split this into $M_i$, consisting of $r$ and $\frac{|B_i|-1}{2}$ c-pairs, $i=1,2$.

In the following, either a b.a.s. $y_{B_i}: B_i \longrightarrow M_i$ is given, or an irregular assignment, having weighted degrees only in $M_i$. In a realization of $y_{B_1}$ the weighted degree $r$ does not occur, only multiples of $r$. In a relization of $y_{B_2}$ no multiple occurs, only $r$ itself. Therefore both together are a b.a.s. for $B_1 \cup B_2$. It is left to the reader to check this in each case, as well as that the proposed block is good and that $y_{B_1}$ and $y_{B_2}$ are block assignment schemes. The condition then follows from Lemma 2.13 or Remark 2.14.

If $H_1$ is a star, then let $B_1 := \{c_1, l_1, l_1'\}$, where $c_1$ is the center vertex and $l_1$, $l_2$ are two leaves. Then let $y_{B_1}(c_1) := r$, $y_{B_1}(l_1) := \kappa_1$, and $y_{B_1}(l_1') := \bar{\kappa}_1$. Observe that, in a realization $x$, $x^*(c_1) \geq 2r$. If $k=5$ and $H_1 \neq K_{1,4}$, then let $B_1 := V(H_1)$. It is straightforward to construct irregular assignments for components of order 5 such that the weighted degrees are contained in the congruence class $r$ and two c-pairs but are not equal to $r$ itself.

If $H_2$ is a star, then let $B_2 := \{c_2, l_2, l_2'\}$, similarly as for $B_1$. But in this case let $y_{B_2}(c_2) := y_{B_2}(l_2) := \kappa_1$, and $y_{B_2}(l_2') := r$. As in (2), $x^*(c_2) \neq x^*(l_2)$. Otherwise let $B_2 := V(H_2)$. If $|H_2| = 5$ and $H_2 \neq K_{1,4}$, then $y_{B_2}$ is given in Figure 4. If $|H_2| = 3$, then the unique choice for $y_{B_2}$ will work. $\Box$

CONDITION 3.8. *G contains no component of order four.*

*Proof.* Assume that $H$ is a component of $G$ with $|H| = 4$. Then let $B := V(H)$ and $M = \{\rho, \kappa_1, \bar{\kappa}_1\}$ as in Definition 2.11. In Figure 5 irregular assignments of $H$, whose weighted degrees are contained in an arbitrary initialization of $M$ are given. Therefore $s(G) \leq n-1$ by Lemma 2.13 and Remark 2.14. $\Box$

CONDITION 3.9. *If G contains only components of order three, then $G = K_3$.*

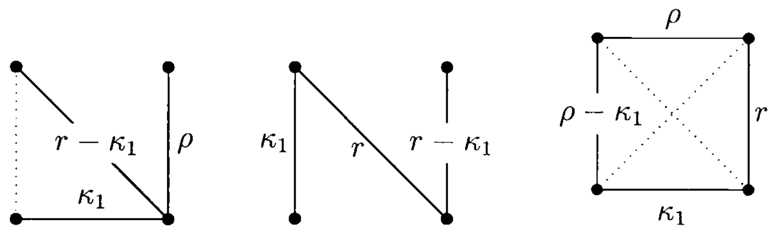


FIG. 5. *Irregular assignments for components of order 4. The dotted edges may exist or not, if they do exist they are assigned the value $r$. The weighted degrees are contained in the set* $\{\rho, \kappa_1, r-\kappa_1\}$.

*Proof.* This final condition is proved by direct construction. There are two possible components of order three: $K_3$ and $P_3$. Assume that $G = s \cdot K_3 + t \cdot P_3$, i.e., $G$ is the disjoint union of $s$ copies of $K_3$ and $t$ copies of $P_3$. In the following it is shown that $s(G) \leq 3(s+t)-1$, except for the case $s=1$ and $t=0$, when $G=K_3$.

The irregularity strength of $tP_3$ was already determined in [1]. The exact value is rather complicated. Here only the upper bound $s(tP_3) \leq \lfloor \frac{5t}{2} \rfloor$ is needed. Since $\lfloor \frac{5t}{2} \rfloor \leq 3t-1$ for $t \geq 1$, the condition holds if $s=0$.

Assume now that $s>0$, and that there is an irregular assignment $x : E(tP_3) \to \{1, \dots, \lfloor \frac{5t}{2} \rfloor\}$. Denote the edges of the $i$th copy of $K_3$ by $e_i$, $e_i'$, and $e_i''$. Let $x : e_i \mapsto \lfloor \frac{5t}{2} \rfloor + 2i - 1$; $e_i' \mapsto \lfloor \frac{5t}{2} \rfloor + 2i$; $e_i'' \mapsto \lfloor \frac{5t}{2} \rfloor + 2i + 1$. Then $x|_{sK_3}$ is irregular.

It is easy to verify that $\max_{v \in tP_3} x^*(v) < \min_{v \in sK_3} x^*(v)$. The assignment $x$ is therefore irregular and the maximal edge weight is $\lfloor \frac{5t}{2} \rfloor + 2s + 1$. Notice that $\lfloor \frac{5t}{2} \rfloor + 2s + 1 \leq 3(s+t)-1$, unless $s=1$ and $t=0$. $\square$

Conditions 3.1, 3.7, 3.8, and 3.9 imply Theorem 1.2.

**4. Acknowledgment.** Theorem 1.2 is the main result of my diploma thesis [6]. I am very grateful to Prof. Dr. E. Triesch for suggesting this topic to me, for supervision, and especially for valuable conversations (not only on this topic).

## REFERENCES

[1] M. AIGNER AND E. TRIESCH, *Irregular assignments of trees and forests*, SIAM J. Discrete Math., 3 (1990), pp. 439–449.
[2] G. CHARTRAND, M. S. JACOBSON, J. LEHEL, O. R. OELLERMANN, S. RUIZ, AND F. SABA, *Irregular networks*, Congr. Numer., 64 (1988), pp. 197–210.
[3] M. S. JACOBSON AND J. LEHEL, *Degree Irregularity*, available online at http://athena.louisville.edu/~msjaco01/irregbib.html
[4] J. LEHEL, *Facts and quests on degree irregular assignments*, in Graph Theory, Combinatorics and Applications, vol. 2, Y. Alavi, G. Chartrand, O. R. Oellermann, and A. J. Schwenk, eds., John Wiley, New York, 1988, pp. 765–781.
[5] L. LOVÁSZ, *Three short proofs in graph theory*, J. Combin. Theory Ser. B, 19 (1975), pp. 269–271.
[6] T. NIERHOFF, *Irreguläres Gewichten von Bäumen und Wäldern*, Diplomarbeit, Universität Bonn, 1995 (in German).
[7] E. S. O'KEEFE, *Verification of a conjecture of Th. Skolem*, Math. Scand., 9 (1961), pp. 80–82.
[8] T. SKOLEM, *On certain distributions of integers in pairs with given differences*, Math. Scand., 5 (1957), pp. 57–68.

SIAM J. DISCRETE MATH.
Vol. 13, No. 3, pp. 324–345

# COMPUTING FUNCTIONS OF A SHARED SECRET[*]

AMOS BEIMEL[†], MIKE BURMESTER[‡], YVO DESMEDT[§], AND EYAL KUSHILEVITZ[¶]

**Abstract.** In this work we introduce and study threshold ($t$-out-of-$n$) secret sharing schemes *for families of functions* $\mathcal{F}$. Such schemes allow any set of at least $t$ parties to compute privately the value $f(s)$ of a (previously distributed) secret $s$, for any $f \in \mathcal{F}$. Smaller sets of players get no more information about the secret than what follows from the value $f(s)$. The goal is to make the shares as short as possible. Results are obtained for two different settings: we study the case when the evaluation is done on a *broadcast channel without interaction*, and we examine what can be gained by allowing evaluations to be done *interactively via private channels*.

**Key words.** secret sharing, broadcast channel, private channels, interaction, private computations

**AMS subject classifications.** 68Q99, 68R05, 94A60, 94A62

**PII.** S0895480195288819

## 1. Introduction.

Suppose, for example, that we are interested in sharing a secret file among $n$ parties in a way that will later allow any $t$ of the parties to test whether a particular string (not known in the sharing stage) appears in this file. This test should be done without revealing the content of the whole file or giving any other information about the file. This problem can be viewed as an extension of the traditional definition of threshold ($t$-out-of-$n$) secret sharing schemes. In the traditional definition, sets of at least $t$ parties are allowed to reconstruct a (previously distributed) secret $s$, while any smaller set gets no information about the secret. We introduce a more general definition of *$t$-out-of-$n$ secret sharing schemes for a family of functions $\mathcal{F}$*. These schemes allow authorized sets of parties to compute some information about the secret and not necessarily the secret itself. More precisely, sets $B$ of size at least $t$ can evaluate $f(s)$ for any function $f \in \mathcal{F}$; any set $C$ of size less than $t$ knows nothing (in the information theoretic sense) about the secret prior to the evaluation of the function and, in addition, *after* $f(s)$ is computed by a set $B$ no set $C$ of size less than $t$ knows anything more about the secret $s$ than what follows from $f(s)$. In other words, the parties in $C$ might know the value $f(s)$, but they know nothing more than that, although they might have heard part of the communication during the evaluation of $f(s)$.

Clearly, if we consider a family $\mathcal{F}$ that includes only the identity function $f(s) = s$, then we get the traditional notion of secret sharing schemes. These schemes, which were introduced by Blakley [8] and Shamir [34], were the subject of a considerable

[*]Received by the editors July 7, 1995; accepted for publication (in revised form) January 20, 2000; published electronically May 15, 2000.
http://www.siam.org/journals/sidma/13-3/28881.html

[†]Department of Computer Science, Ben-Gurion University, Beer Sheva 84105, Israel (beimel@cs.bgu.ac.il, http://www.cs.bgu.ac.il/∼beimel). The research of this author was done when the author was a Ph.D. student in the Department of Computer Science, Technion.

[‡]Department of Mathematics, Royal Holloway, University of London, Egham, Surrey TW20 OEX, UK (m.burmester@rhbnc.ac.uk, http://hp.ma.rhbnc.ac.uk/∼burmester).

[§]Department of Computer Science, Florida State University, PO Box 4530, 206 Love Building, Tallahassee, FL 32306-4530 (desmedt@cs.fsu.edu). The research of this author was partially supported by NSF grants NCR-9106327, INT-9123464, and CCR-9508528.

[¶]Department of Computer Science, Technion, Haifa 32000, Israel (eyalk@cs.technion.ac.il, http://www.cs.technion.ac.il/∼eyalk).

amount of work (e.g., [30, 26, 28, 6, 35, 20]). They were used in many applications (e.g., [31, 5, 15, 19]) and were generalized in various ways [22, 7, 36]. Surveys are given in [35, 37]. The question of sharing many secrets *simultaneously* was considered (with some differences in the definitions) by several researchers [30, 26, 21, 11, 23, 10, 24]. Simultaneous sharing of many secrets is also a special case of our setting.[1] Other similar scenarios in which sharing is viewed as a form of encryption and the security is computationally bounded have been considered in [32, 3, 1].

Threshold cryptography [19, 18, 17] is also a special case of secret sharing for a family of functions.[2] A typical scenario of threshold cryptography is the following: Every set $B$ of $t$ parties should be able to sign any document such that any coalition $C$ of less than $t$ parties cannot sign any other document (even if the coalition $C$ knows the signatures of some documents). To achieve this goal, the key is shared in such a way that every set of $t$ parties can generate a signature from their shares without revealing any information on the key except the signature. Specifically, assume we have a signature function $\mathrm{SIGN} : M \times K \to O$, where $M$ is the domain of messages, $K$ is the domain of keys, and $O$ is the domain of signatures. For every $m \in M$ define $f_m : K \to O$ as $f_m(k) = \mathrm{SIGN}(m, k)$. The previous scenario is simply sharing the key for the family $\{f_m \ : \ m \in M\}$. These examples show that secret sharing for a family of functions is a natural primitive.

Obviously, one possible solution to the problem of sharing a secret for a family $\mathcal{F}$ is by sharing *separately* each of the values $f(s)$ (for any $f \in \mathcal{F}$) using known threshold schemes. While this solution is valid, it is very inefficient, in particular when the size of $\mathcal{F}$ is large. Therefore, an important goal is to realize such schemes while using "small" shares. For example, to share a single bit among $n$ parties, the average (over the parties) length of shares is at least $\log(n - t + 2)$ (for $2 \leq t \leq n - 1$) [27] and $\log n$ bits are sufficient [34]. The obvious solution for sharing $\ell$ bits simultaneously will require $\ell \log n$ bits. By [26], it can be shown that shares of at least $\ell$ bits are necessary. We shall show that $\ell$-bit shares are also sufficient if *interactive* evaluation on private channels is allowed, and $O(\ell)$-bit shares are sufficient if noninteractive evaluation (on a broadcast channel) is used.[3]

We present an interactive scheme in which $\ell$-bit secrets are distributed using $\ell$-bit shares; this scheme allows the computation of every linear combination of the bits (and not only computation of the bits themselves). We use this scheme to construct schemes for other families of functions. The length of the shares in these schemes can be much longer than the length of the secret. An interesting family of functions that we shall consider is the family $\mathcal{ALL}$ of *all* functions of the secret. For this family, we construct a scheme in which the length of the shares is $\approx 2^{\ell} \log n$ (where $\ell$ is the length of the secret and $2^{\ell}$ is the length of the description of a function which is evaluated). In this scheme we can use a broadcast channel with no interaction during the computation. If we allow interaction on private channels during the computation, then the length of the shares can be reduced to $2^{\ell}$. Note that the obvious solution of sharing each bit separately requires, in this case, $2^{2^{\ell}}$ bits.

[1]Let $s_1, s_2, \ldots, s_\ell$ be the secrets we want to share *simultaneously*. Construct the concatenated secret $s = s_1 \circ s_2 \circ \cdots \circ s_\ell$ and the functions which can be evaluated are the functions $f_i(s) = s_i$. The secrets $s_i$ may be dependent: our model allows some information to leak provided it is no more than what follows from the evaluations of $f(s)$.

[2]The functions considered in [19, 18, 17] are, however, very limited and the scenario in [17] is restricted to computational security.

[3]In fact, both results require $\ell$ to be "sufficiently large": $\ell \geq \log n$ in the interactive case and $\ell \geq \log n \log \log n$ in the broadcast model.

Our work deals mainly with two models: the *private channels model* in which the computation might require a few rounds of communication, and the *broadcast channel model* in which the computation is noninteractive. On one hand, the broadcast model does not require secure private channels and synchronization; hence, the computation is more efficient. On the other hand, in the private channels model, a coalition $C$ that does not intersect the evaluating set $B$ will know nothing about $s$; not even the value $f(s)$. Interaction seems to be useful in the computation. It enables us to reduce the length of the shares by a factor of $\log n$ in our basic scheme for linear functions.

We also study *ideal* threshold schemes. These are schemes in which the size of the shares equals the size of the secrets[4] (see, e.g., [12, 13, 4, 33, 25]). We deal with the characterization of the families of functions $\mathcal{F}$ which can be evaluated by an ideal threshold scheme. For the interactive private channels model, we prove that every boolean function that can be evaluated is a *linear* function. For the broadcast model, we prove that $\mathcal{F}$ cannot contain any boolean function (for every family that contains the identity function).

*An example.* To motivate our approach and to clarify the notion of secret sharing for a family of functions we consider a simple example, which we discuss informally. Suppose that a dealer shares a secret $s$ among $n$ parties and that at some time later a set of at least $t$ parties would like to answer the question: "Is the shared secret equal to the string $a$?" That is, they want to evaluate the boolean function $f_a(s)$ which is 1 if and only if $s = a$. The string $a$ is not known when the dealer shares the secret. We describe a simple scheme which makes it possible for the parties to answer such questions. Let $s = s_1 s_2 \dots s_\ell$ be the secret bit-string. The dealer chooses $2\ell$ random strings from $\{0,1\}^{2\ell}$, denoted by $b_0, b_1, \dots, b_{2\ell-1}$, and shares each of these strings using Shamir's $t$-out-of-$n$ threshold scheme [34]. Let $b_{i,j}$ be the share of the secret $b_i$ given to party $P_j$. The dealer computes the sum (i.e., bitwise exclusive-or) $\alpha = \sum_{i=1}^{\ell} b_{2i+s_i}$ (that is, $s_i$, the value of the $i$th bit of the secret, selects whether we take $b_{2i}$ or $b_{2i+1}$) and gives $\alpha$ to each party. When a subset $B$ of (at least) $t$ parties wants to check if $s = a$ for some $a = a_1 a_2 \dots a_\ell$, each $P_j \in B$ computes a "new" share $\sum_{i=1}^{\ell} b_{(2i+a_i),j}$. Then, the parties in $B$ apply Shamir's secret reconstruction procedure to compute a "new" secret from these "new" shares. Since this procedure involves only computing a linear combination of the shares, the "new" secret is simply $\sum_{i=1}^{\ell} b_{2i+a_i}$. If the "new" secret differs from $\sum_{i=1}^{\ell} b_{2i+s_i}$, then the parties learn that $s \neq a$. However, they get no more information on the secret $s$. If the "new" secret is equal to $\sum_{i=1}^{\ell} b_{2i+s_i}$, then it is easy to see that with high probability (over the choice of $b_1, \dots, b_\ell$) we must have $s = a$. In this case, the parties learn the secret and there is no additional information which should be hidden from them. Observe that in both cases the parties learn no more than what is revealed by answering the question.

*Connection to private computations.* In our schemes we require that authorized sets of parties can compute a function of the secret without leaking any other information on the secret. This resembles the requirement of $(n,k)$-private protocols [5, 15] in which the set of all $n$ parties can evaluate a function of their inputs in a way that no set of less than $k$ parties will gain any additional information about the inputs of the other parties. Indeed, in some of our schemes a set $B$ of size $t$ uses a $(t,t)$-private protocol. However, it is not necessary to use private protocols in the computation, since the parties are allowed to leak information about their inputs (the shares) as long as they do not leak any additional information on the secret. Moreover, using

[4]The size of each share is always at least the size of the secret [26].

private protocols does not solve the problem of efficiently sharing a secret for all functions, since not all functions can be computed $(t,t)$ privately [5, 16]. Furthermore, the parties cannot use the $(t, \lfloor (t-1)/2 \rfloor)$-private protocols of [5] or [15] since this means that coalitions of size greater than $t/2$ (but still smaller than $t$) gain information.

*Organization.* The rest of this paper is organized as follows. In section 2 we discuss our model for secret sharing for families of functions and provide the definitions—definitions which are more delicate than in the case of traditional threshold schemes. In section 3 we present interactive and broadcast secret sharing schemes for the family $\mathcal{LIN}$ of linear functions. In section 4 we use these schemes to construct schemes for other families. In section 5 we describe the broadcast scheme for the bit functions $\mathcal{BIT}$. In section 6 we characterize ideal schemes in the various models. Finally, in section 7 we discuss possible extensions of our work. For completeness we give some background results in three appendices.

## 2. Definitions.

This section contains a formal definition of secret sharing for a family of functions. We start by defining the model. We consider a system with $n$ parties $U = \{P_1, P_2, \ldots, P_n\}$. In addition to the parties, there is a dealer who has a secret input $s$. A *distribution scheme* is a probabilistic mapping, which the dealer applies to the secret to generate $n$ pieces of information $s_1, \ldots, s_n$ which are referred to as *the shares*. The dealer gives the share $s_i$ to party $P_i$. Formally, we have the following.

Definition 2.1 (distribution schemes). *Let $S$ be a set of secrets, $S_1, S_2, \ldots, S_n$ sets of shares, and $R$ a set of random inputs. Let $\mu : R \to [0,1]$ be a probability distribution on $R$. A* distribution scheme $\Pi$ *over $S$ is a mapping $\Pi : S \times R \to S_1 \times S_2 \times \cdots \times S_n$. The ith coordinate $s_i$ of $\Pi(s,r)$ is the share of $P_i$. The pair $(\Pi, \mu)$ can be thought of as a dealer who shares a secret $s \in S$ by choosing at random $r \in R$ (according to the distribution $\mu$) and then gives each party $P_i$ as a share the corresponding component of $\Pi(s,r)$.*

In the scenario we consider, the dealer is active only during the initialization of the system. After this stage the parties can communicate. We next define our communication models.

Definition 2.2 (communication models). *We consider the following two models of communication:*

1. *The* private channels *model in which the parties communicate via a complete synchronous network of secure and reliable point-to-point communication channels. In this model a set of parties has access only to the messages sent to the parties in the set.*
2. *The* broadcast channel *model in which the parties communicate via a (public) broadcast channel. In this model a set of parties can obtain all the messages exchanged by the communicating parties.*

A subset of the parties may communicate in order to compute a function of their shares. We now define how they evaluate this function.

Definition 2.3 (function evaluation). *A set of parties $B \subseteq U$ executes a protocol $F_{B,f}$ to evaluate a function $f$.*

*At the beginning of the execution, each party $P_i$ in $B$ has an input $s_i$ and chooses a random input $r_i$. To evaluate $f$, the parties in $B$ exchange messages as prescribed by the protocol $F_{B,f}$. In each round every party in $B$ sends messages to every other party in $B$. A message sent by $P_i$ is determined by $s_i$, $r_i$, the messages received by $P_i$ so far, and the identity of the receiver. We say that a* protocol $F_{B,f}$ evaluates $f$ (or computes $f(s)$) *if each party in $B$ can always evaluate $f(s)$ from its input and the* *communication it obtained.*

*We denote by $M_C(\langle s_i \rangle_B, \langle r_i \rangle_B)$ the messages that an eavesdropping coalition $C$ can obtain during the communication between the parties in $B$ (this depends on the communication model). Here $\langle s_i \rangle_B$ is the vector of shares of $B$ and $\langle r_i \rangle_B$ the vector of random inputs of $B$.*

*A protocol is* noninteractive *if the messages sent by each party $P_i$ depend only on its input (and not on the messages received during the execution of the protocol). Noninteractive protocols have only one round of communication. An* interactive *protocol might have more than one round. In this case we require that the protocol terminates after a finite number of rounds (that is, we do not allow infinite runs).*

*The parties in the system are honest, that is, they send messages according to the protocol.*

A coalition $C$ which eavesdrops on the execution of a protocol $F_{B,f}$ by the parties in $B$ *gains no additional information on the secret $s$* if any information that $C$ may get is *independent* of the old information. In our model we shall allow $C$ to gain some information, but no more than what follows from the evaluation of $f$. The information that $C$ gains is determined by the view of $C$.

Definition 2.4 (the view of a coalition). *Let $C \subset U$ be a coalition. The* view *of $C$, denoted $VIEW_C$, after the execution of a protocol consists of the information that $C$ gains. In a distribution scheme, $VIEW_C = \langle s_i \rangle_C$, the shares of $C$. In the evaluation of $f(s)$ by a set of parties $B \subseteq U$ the view of $C$ consists of the inputs $\langle s_i \rangle_C$, the local random inputs $\langle r_i \rangle_{C \cap B}$ (only the parties in $C \cap B$ are involved in the computation), and the messages that $C$ obtains from the communication channel during the evaluation by $B$, i.e., $M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B)$. That is,*

$$VIEW_C = \langle s_i \rangle_C \circ \langle r_i \rangle_{P_i \in B\cap C} \circ M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B).$$

We now define $t$-out-of-$n$ secret sharing schemes *for a family of functions $\mathcal{F}$*. Such schemes allow any set $B$ of at least $t$ parties to evaluate $f(s)$, for any $f \in \mathcal{F}$, where $s$ is a previously distributed secret, while any set $C$ of less than $t$ parties gains no more information about the secret than the information inferred from $f(s)$. We distinguish between three types of schemes depending on the way that the value $f(s)$ is computed by $B$ and the channels used: (1) *interactive private channels* schemes, where the parties in $B$ engage in a protocol via private channels which computes $f(s)$; (2) *noninteractive private channels* schemes, where the parties in $B$ engage in a noninteractive protocol via private channels to compute $f(s)$; and (3) *broadcast noninteractive* schemes, where each party in $B$ broadcasts a single message (which depends only on its share) on a broadcast channel (we do not consider interactive broadcast schemes in this paper).

The computation of $f(s)$ must be *secure*. There are two requirements to consider: (1) the usual requirement for threshold schemes; that is, before the evaluation any set $C$ of size less than $t$ knows nothing about the secret by viewing their shares; (2) after the evaluation of $f(s)$ by a set $B$, any set $C$ of size less than $t$ gains no information about $s$ that is not implied by $f(s)$. Formally, we have the following.

Definition 2.5 (secret sharing schemes for a family). *A $t$-out-of-$n$* secret sharing scheme for a family of functions $\mathcal{F}$ *is a distribution scheme* $(\Pi, \mu)$ *which satisfies the following two conditions:*

Evaluation. *For any set $B \subseteq U$ of size at least $t$ and any function $f \in \mathcal{F}$ the parties in $B$ can evaluate $f(s)$. That is, there is a protocol $F_{B,f}$ which, given the shares of $B$ as inputs, will always output the correct value of $f(s)$. The scheme* *is called* noninteractive *if $F_{B,f}$ is noninteractive, and* interactive *otherwise. Depending on the communication model, we use either a broadcast channel or private channels.*

Security. *Let $X$ be a random variable on the set of secrets $S$ and $C \subset U$ be any coalition of size less than $t$.*

Prior to evaluation (after the distribution of shares). *For any two secrets $s, s' \in S$ and any shares $\langle s_i \rangle_C$ (from $\Pi(s, r)$):*

$$\Pr[\,\mathit{VIEW}_C = \langle s_i \rangle_C \mid X = s\,] \quad = \quad \Pr[\,\mathit{VIEW}_C = \langle s_i \rangle_C \mid X = s'\,]\,.$$

After evaluation. *For any $f \in \mathcal{F}$, any $B \subseteq U$ of size at least $t$, any secrets $s, s' \in S$ with $f(s) = f(s')$, any shares $\langle s_i \rangle_C$, any inputs $\langle r_i \rangle_{B\cap C}$, and any messages $M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B)$ from the computation of $f(s)$ by $B$:*

$$\begin{aligned}&\Pr[\,\mathit{VIEW}_C = \langle s_i \rangle_C \circ \langle r_i \rangle_{B\cap C} \circ M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) \mid X = s\,] \\ &\quad = \Pr[\,\mathit{VIEW}_C = \langle s_i \rangle_C \circ \langle r_i \rangle_{B\cap C} \circ M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) \mid X = s'\,]\,.\end{aligned}$$

*We say that $C$* gains no information that is not implied by the function $f$ (or simply: gains no additional information) *if in the computation of $f(s)$ we have security after evaluation.*

A (traditional) $t$-out-of-$n$ secret sharing scheme is a secret sharing scheme for the family that includes only the identity function $f(s) = s$ and its renamings. (From Observation 2.1 below it follows that if a function $f$ can be evaluated securely, then so can all its renamings.)

*Remark* 2.1. We require only that one function $f \in \mathcal{F}$ can be evaluated securely. A more desirable property is that the scheme is reusable: any number of functions can be evaluated securely (possibly by different sets). All our schemes, except for the scheme in section 5, are reusable.

*Remark* 2.2. In Definition 2.5 we required that all coalitions $C$ of size less than $t$ should gain no additional information on the secret. Clearly, it is sufficient to require this only for coalitions $C$ of size $t-1$, since any smaller coalition has less information on the secret and, hence, will gain no additional information.[5]

*Remark* 2.3. The security in Definition 2.5 is based on the requirement that the view of a coalition is the same for secrets which have the same evaluation. Alternatively we could require $\Pr[X = s \mid \mathrm{VIEW}_C = \mathit{view}] = \Pr[X = s \mid f(s) = v]$. That is, the probability that the secret is $s$, given the view of the coalition $C$, equals the probability of $s$, given the value $v = f(s)$. In Appendix A we show that the two definitions are equivalent (see [14] for the analogue claim with respect to traditional secret sharing schemes).

Obviously, any broadcast scheme can be transformed into a scheme in which each message is sent on private channels. On the other hand, if we take a noninteractive private channels scheme and broadcast the messages, then the security of the computation may be violated.

A function $f'$ is a *renaming* of $f$ if $f(x) = f(y)$ if and only if $f'(x) = f'(y)$. Suppose that $f'$ is a renaming of $f$ and that $f$ can be evaluated securely; then $f'$ can also be evaluated securely. Indeed, to evaluate $f'(s)$ securely we first evaluate $f(s)$ securely; after this evaluation $s$ might not be known. Nevertheless, we can find an input $y$, such that $f(y) = f(s)$, and output $f'(y)$.

[5]This is different from what happens in private multiparty computation; there, a smaller set has less inputs and hence is expected to learn even less.

Observation 2.1. *A secret sharing scheme enables the secure evaluation of a* renaming *of $f$ if and only if it enables the secure evaluation of $f$.*

Therefore, we ignore renamings of functions.

We next define certain classes of functions over $\mathrm{GF}(2^\ell)$ which are of particular interest. Recall the structure of $\mathrm{GF}(2^\ell)$. Each element $a \in \mathrm{GF}(2^\ell)$ is an $\ell$-bit string $a_{\ell-1} \ldots a_1 a_0$ which may be represented by the polynomial $a_{\ell-1}x^{\ell-1} + \cdots + a_1 x^1 + a_0$. Addition and multiplication are as for polynomials, using the structure of $\mathrm{GF}(2)$ for the coefficients and reducing products modulo a polynomial $p(x)$ of degree $\ell$, which is irreducible over $\mathrm{GF}(2)$. A function $f : \mathrm{GF}(2^\ell) \to \mathrm{GF}(2^\ell)$ is *linear* over $\mathrm{GF}(2)$ if $f(x+y) = f(x) + f(y)$ for every $x, y \in \mathrm{GF}(2^\ell)$. Notice that $f(0) = f(0) + f(0) = 0$ for every linear function $f$. Thus, we have the following.

Observation 2.2. *For every linear function $f$, for every $x \in \mathrm{GF}(2^\ell)$, and for every $a \in \mathrm{GF}(2)$ it holds that $f(ax) = af(x)$.*

Notice that we *do not* require that $f(ax) = af(x)$ for $a \in \mathrm{GF}(2^\ell)$. We define the family $\mathcal{LIN}_\ell$ to be the family of all linear functions of $\mathrm{GF}(2^\ell)$ over $\mathrm{GF}(2)$ (so $\mathcal{LIN}_\ell$ is the family of *additive* functions). Let $e_i : \mathrm{GF}(2^\ell) \to \mathrm{GF}(2)$ be the function that returns the $i$th bit of $x$. Then, $e_i(x+y) = e_i(x) + e_i(y)$ and, hence, $e_i$ is linear. We call the family $\{e_1, \ldots, e_\ell\}$ the *bit functions* and denote it by $\mathcal{BIT}_\ell$. We also consider the family $\mathcal{ALL}_\ell$ of all possible functions of the secret (by Observation 2.1, without loss of generality, the range of the functions in $\mathcal{ALL}_\ell$ is in $\mathrm{GF}(2^\ell)$). We stress that $\mathcal{BIT}_\ell$ contains only boolean functions while both $\mathcal{ALL}_\ell$ and $\mathcal{LIN}_\ell$ contain nonboolean functions as well.

We will need the following claim concerning linear functions.

Claim 2.1. *For every linear function $f$, there are constants $c_1, \ldots, c_\ell$ such that $f(b_1 b_2 \ldots b_\ell) = \sum_{j=1}^{\ell} c_j b_j$ for every $\ell$-bit string $b_1 b_2 \ldots b_\ell$.*

*Proof.* Notice that $b_1 b_2 \ldots b_n = \sum_{j=1}^{n} b_j e_j$. Thus, since $f$ is linear, and by Observation 2.2,

$$f(b_1 b_2 \ldots b_n) = f\left(\sum_{j=1}^{n} b_j e_j\right) = \sum_{j=1}^{n} f(b_j e_j) = \sum_{j=1}^{n} f(e_j)\, b_j.$$

Now, define $c_i = f(e_i)$, and the claim follows. □

## 3. Schemes for the linear functions.

**3.1. An interactive scheme.** In this section we show that Shamir's scheme over $\mathrm{GF}(2^\ell)$ [34] (described in Appendix B) is also a secret sharing scheme for the family $\mathcal{LIN}_\ell$—the family of linear functions.

Theorem 3.1 (basic interactive scheme). *For every $\ell \geq 1$ there exists an interactive private channels $t$-out-of-$n$ secret sharing scheme for the family $\mathcal{LIN}_\ell$ in which the secrets have length $\ell$ and the shares have length $\ell' \triangleq \max\{\ell, \lceil \log(n+1) \rceil\}$.*

*Proof.* The dealer uses Shamir's $t$-out-of-$n$ scheme over $\mathrm{GF}(2^{\ell'})$ to distribute the shares. We show how every function $f \in \mathcal{LIN}_\ell$ can be evaluated securely. Consider a set $B$ of $t' \geq t$ parties with shares $\langle s_i \rangle_B$, where $s_i \in \mathrm{GF}(2^{\ell'})$, that wishes to compute $f(s)$. By the reconstruction procedure of Shamir's scheme, for each $P_j \in B$ there exist coefficients $\delta_j$ such that $s = \sum_{P_j \in B} \delta_j s_j$ (see Appendix B). Since $f$ is linear, $f(s) = f(\sum_{P_j \in B} \delta_j s_j) = \sum_{P_j \in B} f(\delta_j s_j)$. Let $x_j \triangleq f(\delta_j s_j)$. Then, $f(s)$ is simply the sum of the $x_j$'s. Computing this sum is done using the following interactive protocol of Benaloh [6] (for a more detailed description of this protocol see Appendix C). For convenience of notation, we assume that $B = \{P_1, P_2, \ldots, P_{t'}\}$. In the first step of the protocol each party $P_i \in B$ chooses $t'-1$ random inputs $r_{i,1}, \ldots, r_{i,t'-1}$ in $\mathrm{GF}(2^{\ell'})$, computes $r_{i,t'} \triangleq x_i - \sum_{j=1}^{t'-1} r_{i,j}$, and sends $r_{i,j}$ to $P_j \in B$. Then, each party $P_j$ computes $y_j = \sum_{P_i \in B} r_{i,j}$ and sends $y_j$ to $P_1$. Party $P_1$ now computes $\sum_{j=1}^{t'} y_j$ and sends it to all the other parties. By the choice of $r_{i,t'}$, the sum $\sum_{j=1}^{t'} y_j$ equals $\sum_{P_i \in B} x_i = f(s)$; thus, each party in $B$ computes $f(s)$ correctly.

Let us now prove the security of this scheme. The security prior to evaluation follows from the security of Shamir's threshold scheme. For the security after the evaluation it is sufficient to consider a coalition $C$ of size $t-1$ (by Remark 2.2). Since each $x_j$ is computed locally, and since $f(s)$ is computed using a private protocol [6] (for a formal definition of privacy see Appendix C), the parties in $C$ gain no information about the shares of the other parties that is not implied by $f(s)$. That is, for any shares $\langle s_i \rangle_B$ and $\langle s'_i \rangle_B$ with $s_i = s'_i$ for all $P_i \in B \cap C$ and $\sum_{P_j \in B} f(\delta_j s_j) = \sum_{P_j \in B} f(\delta_j s'_j)$, we have

$$\begin{aligned} &\Pr[\ \langle r_i \rangle_{B\cap C} \circ M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) \mid \langle s_i \rangle_B] \\ (1) \qquad &\qquad = \Pr[\ \langle r_i \rangle_{B\cap C} \circ M_{B\cap C}(\langle s'_i \rangle_B, \langle r_i \rangle_B) \mid \langle s'_i \rangle_B]. \end{aligned}$$

Now for a secret $s$ and shares $\langle s_i \rangle_C$ with $|C| = t-1$, there are unique shares $\langle s_i \rangle_{B \setminus C}$ of Shamir's scheme that are consistent with $s$ and $\langle s_i \rangle_C$. Therefore,

$$\begin{aligned} \Pr[\ \mathrm{view}_C \mid s\ ] &= \Pr[\ \langle s_i \rangle_C \circ \langle r_i \rangle_{B\cap C} \circ M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) \mid s\ ] \\ &= \Pr[\ \langle s_i \rangle_B \circ \langle r_i \rangle_{B\cap C} \circ M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) \mid s\ ] \\ &= \Pr[\ \langle r_i \rangle_{B\cap C} \circ M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) \mid \langle s_i \rangle_B \circ s\ ] \cdot \Pr[\ \langle s_i \rangle_B \mid s\ ] \\ (2) \qquad &= \Pr[\ \langle r_i \rangle_{B\cap C} \circ M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) \mid \langle s_i \rangle_B] \cdot \Pr[\ \langle s_i \rangle_B \mid s\ ]\ . \end{aligned}$$

The last equation holds because the shares $\langle s_i \rangle_B$ determine the secret $s$. By (1), the probability $\Pr[\ \langle r_i \rangle_{B\cap C} \circ M_{B\cap C}(\langle s_i \rangle_B, \langle r_i \rangle_B) \mid \langle s_i \rangle_B]$ is independent of $\langle s_i \rangle_B$ and therefore of $s$, and by the properties of Shamir's scheme the probability $\Pr[\ \langle s_i \rangle_B \mid s\ ]$ is independent of $\langle s_i \rangle_B$ and $s$. Therefore, by (2), for every $s' \in S$ such that $f(s) = f(s')$,

$$\Pr[\mathrm{view}_C \mid s] = \Pr[\mathrm{view}_C \mid s'],$$

and the coalition $C$ gains no information about $s$ that is not implied by $f(s)$. $\square$

*Remark* 3.1. In the special cases when $t = 2$ or $t = n$, the parties can compute $f(s)$ without interaction by simply sending the messages $x_j$. For $t = 2$ the messages must be sent via private channels. In this case, only coalitions $C \subseteq B$ can get messages of the evaluating set $B$. Let $C = \{P_i\}$, $B = \{P_i, P_j\}$. Then, party $P_i$ gains no additional information from the message $x_j$ of $P_j$, because $P_i$ knows $f(s) = x_i + x_j$ and its input $x_i$, and therefore can compute $x_j$. For $t = n$ the messages may be sent via a public broadcast channel. In this case, any coalition $C$ is a subset of $B$, and if $C$ has size $n-1$, then $B = C \cup \{P_j\}$ for some $P_j$. Again $C$ gains no additional information from $x_j$, since $C$ can compute $x_j$ from $f(s)$ and $\sum_{P_i \in C} x_i$.

*Remark* 3.2. When $3 \leq t \leq n-1$, the parties evaluate $f(s)$ using an interactive private channels scheme. It can be shown that for this particular scheme interactive evaluation is essential. Otherwise, some coalitions $C$ that intersect the evaluating set $B$, but are not contained in $B$, will gain additional information on $s$ when they obtain the values of $x_j$.

This scheme has the advantage that it is *reusable*—sets $B_1, \ldots, B_m$ can compute values $f_1(s), \ldots, f_m(s)$ while any coalition $C$ does not gain any information on the secret beside the values $f_i(s)$ for every $i$ with $C \cap B_i \neq \emptyset$. This is because the evaluation is done using a private protocol. The private computation ensures that $C$ cannot distinguish between two vectors of shares with the same value $f_i(s)$. Hence, even if $C$ knows some evaluations $f_i(s)$, an evaluation of another function will not reveal extra information.

**3.2. A noninteractive broadcast scheme.** Next, we present a broadcast scheme for the family $\mathcal{LIN}_\ell$ of linear functions, in which the length of the shares is $\ell \log n + \ell$.

THEOREM 3.2 (basic broadcast scheme). *For every $\ell \geq 1$ there exists a noninteractive broadcast $t$-out-of-$n$ secret sharing scheme for the family $\mathcal{LIN}_\ell$ in which the secrets have length $\ell$ and the shares have length $\ell \log n + \ell$.*

*Proof.* Let $q \triangleq \lceil \log(n+1) \rceil$. The dealer shares each bit of the secret independently using Shamir's scheme over GF($2^q$) (this is the smallest possible field of characteristic two for Shamir's scheme). Let $s = b_1 b_2 \ldots b_\ell$ be the secret bit-string and let $s_{i,j} \in$ GF($2^q$) be the share of $b_j$ given to party $P_i$. Then, the total length of the share of each party is $\ell \cdot q = \ell \lceil \log(n+1) \rceil \leq \ell \log n + \ell$.

Clearly, every bit of the secret can be securely reconstructed. To evaluate other linear functions of the secret, we use the homomorphic property of Shamir's scheme, as observed by Benaloh [6], which enables the evaluation of linear combinations of a shared secret without revealing other information on the secret. To explain how this property is used we first observe that it is sufficient to show the result for a boolean linear function, since each coordinate of $f(s)$, where $f \in \mathcal{LIN}_\ell$, is also a linear function (we use the property that this scheme is reusable).

Let $f$ be a boolean linear function and let $B$ be a set of (at least) $t$ parties with shares $\langle s_{i,j} \rangle_{P_i \in B, 1 \leq j \leq \ell}$ that wishes to compute the bit $f(s)$. By Claim 2.1, there are constants $c_1, \ldots, c_\ell$ such that $f(b_1 b_2 \ldots b_\ell) = \sum_{j=1}^{\ell} c_j b_j$. Recall that for Shamir's reconstruction procedure the secret bit $b_j$ is computed as $b_j = \sum_{P_i \in B} \delta_i s_{i,j}$, where $\delta_i \in$ GF($2^q$) are appropriate coefficients. Let $x_i \triangleq \sum_{j=1}^{\ell} c_j s_{i,j}$ be a "new" share of $P_i$. Then, $f(s) = \sum_{j=1}^{\ell} c_j b_j = \sum_{j=1}^{\ell} c_j \sum_{P_i \in B} \delta_i s_{i,j} = \sum_{P_i \in B} \delta_i \sum_{j=1}^{\ell} c_j s_{i,j} = \sum_{P_i \in B} \delta_i x_i$.

To evaluate $f(s)$ each party $P_i \in B$ computes locally the "new" share $x_i$ and then broadcasts it. The "new" secret $f(s)$ is just the linear combination of these "new" shares. We claim that a coalition $C$ of size $t-1$ gains no information that is not implied by $f(s)$ during this process. Indeed, the probability of the view of $C$ is

$$\begin{aligned} \Pr[\langle s_{i,j} \rangle_{P_i \in C, 1 \leq j \leq \ell} \circ \langle x_i \rangle_{P_i \in B} \mid s] &= \Pr[\langle s_{i,j} \rangle_{P_i \in C, 1 \leq j \leq \ell} \circ f(s) \mid s] \\ &= \Pr[\langle s_{i,j} \rangle_{P_i \in C, 1 \leq j \leq \ell} \mid s\,]. \end{aligned}$$

The first equality holds because the shares of $C$ and $f(s)$ determine the "new" shares of $B$ and vice-versa. The second equality holds because $s$ determines $f(s)$. Now $\Pr[\langle s_{i,j} \rangle_{P_i \in C, 1 \leq j \leq \ell} \mid s\,]$ is independent of $s$ by the security requirement prior to evaluation. So, we get the security requirement after evaluation. Furthermore, if this process is repeated more than once and different linear combinations are evaluated, then $C$ will gain no information that is not implied by these linear combinations, and the parties in $B$ can evaluate each bit of a nonboolean linear function independently. $\square$

*Remark* 3.3. Observe that in the broadcast scheme there is no need for the set $B$ to be given as input for the evaluation: each party $P_i$ which is available just outputs an appropriate linear combination of its shares, and its identity $P_i$. Then, a function can be evaluated so long as there are at least $t$ parties that agree to evaluate the function. These parties do not need to be available at the same time or to know in advance which parties would be available.

**4. Schemes for other families of functions.** We now show how to use the basic scheme (for linear functions) to construct schemes for other families of functions. Given a family of functions, we shall construct a longer secret such that every function in the family can be evaluated as a linear function of the longer secret. Observe that any boolean function $f : \mathrm{GF}(2^\ell) \to \mathrm{GF}(2)$ can be represented as a binary vector over $\mathrm{GF}(2)$ of length $2^\ell$ whose $i$th coordinate is $f(i)$. Similarly, any function $f : \mathrm{GF}(2^\ell) \to \mathrm{GF}(2^{\ell'})$ can be represented as an array of $\ell'$ binary vectors of length $2^\ell$ in which the $j$th vector corresponds to the $j$th bit function of $f(x)$. The *rank* of a family of functions $\mathcal{F}$ is the smallest $k$ for which there exist boolean functions $f_1, \ldots, f_k$ such that for every function $f \in \mathcal{F}$ there exists a renaming of it $f'$, for which each of the $\ell'$ vectors representing $f'$ is a linear combination of $f_1, \ldots, f_k$. The rank of a family does not change if we add renamings of functions. So, we can assume that $\mathcal{F}$ contains all renamings of its functions.

THEOREM 4.1. *Let $\mathcal{F}$ be any family of functions.*

- *There exists a $t$-out-of-$n$ interactive private channels secret sharing scheme for $\mathcal{F}$ with shares of length* $\max\{\mathrm{rank}(\mathcal{F}), \log n + 1\}$.
- *There exists a $t$-out-of-$n$ noninteractive broadcast secret sharing scheme for $\mathcal{F}$ with shares of length* $\mathrm{rank}(\mathcal{F})(\log n + 1)$.

*Proof.* Let $f_1, f_2, \ldots, f_k$ be a basis for the vector space spanned by the functions in $\mathcal{F}$. To share a secret $s$ the dealer generates a new secret

$$\mathrm{E}(s) = f_1(s) \circ f_2(s) \circ \cdots \circ f_k(s)$$

of length $k$ (where $\circ$ denotes concatenation of strings). The dealer now shares the secret $\mathrm{E}(s)$ using the basic scheme (either the interactive or broadcast version, depending on the communication model). Then, $f_i(s) = e_i(\mathrm{E}(s))$ for $i = 1, 2, \ldots, k$. Let $f$ be the function in $\mathcal{F}$ to be evaluated. Without loss of generality, we may assume that $f : \mathrm{GF}(2^\ell) \to \mathrm{GF}(2^{\ell'})$ is such that the $\ell'$ vectors representing it are spanned by $f_1, \ldots, f_k$. Every bit of $f(s)$ is a linear combination of $f_1, \ldots, f_k$ and therefore of the bits of $\mathrm{E}(s)$ (i.e., $e_i(\mathrm{E}(s))$). Since addition in $\mathrm{GF}(2^k)$ is bitwise, concatenation of linear functions is a linear function too. Therefore $f(s)$ can be computed by evaluating a linear function of $\mathrm{E}(s)$. Thus Theorem 4.1 follows by Theorems 3.1 and 3.2. $\square$

We demonstrate the applicability of the above construction by considering the family $\mathcal{ALL}$ of all possible functions of the secret (boolean and nonboolean).

COROLLARY 4.2.

- *There exists a $t$-out-of-$n$ interactive private channels secret sharing scheme for $\mathcal{ALL}_\ell$ with shares of length* $\max\left\{2^\ell - 1, \log n + 1\right\}$.
- *There exists a $t$-out-of-$n$ noninteractive broadcast secret sharing scheme for $\mathcal{ALL}_\ell$ with shares of length* $(2^\ell - 1)(\log n + 1)$.

*Proof.* By Theorem 4.1 it is sufficient to show that the rank of $\mathcal{ALL}_\ell$ is $2^\ell - 1$. Let $d = 2^\ell - 1$. Consider the $d$ boolean functions $f_1, \ldots, f_d$, where $f_i(x) = 1 \Leftrightarrow x = i$. To evaluate a boolean function $f$ we notice that the functions $f(x)$ and $f(x) + 1$ are renamings of each other, so we can assume that $f(0) = 0$. Hence, $f(s) = \sum_{1 \le i \le d} f_i(s) f(i)$, and therefore $f_1, \ldots, f_d$ form a basis for $\mathcal{ALL}_\ell$. In this case, the secret $s$ is encoded as the vector $\mathrm{E}(s)$ of length $d$ in which the $s$th coordinate is 1 and all the other coordinates are zero (with the exception of $\mathrm{E}(0) = 0$). $\square$

Notice that the length of the secret is $\ell$, while the length of the shares is $(2^\ell - 1)$. However, there are $2^{2^\ell - 1}$ different boolean functions with domain of cardinality $2^\ell$ such that every function is not a renaming of another function (leaving aside nonboolean functions). Therefore, our scheme is significantly better than the naive scheme in which we share every function (up to renaming) separately. Also, the length of $E(s)$ for $\mathcal{ALL}_\ell$ must be at least $\log 2^{2^\ell-1} = 2^\ell - 1$, so the representation for $E(s)$ that we use is the best possible for the family $\mathcal{ALL}_\ell$ in this particular scheme. It remains an interesting open question whether there exists a better scheme for this family or whether one can prove that this scheme is optimal.

## 5. Noninteractive broadcast scheme for the bit family.

In this section we present a noninteractive broadcast scheme for the family of bit functions, whose shares are of length $O(\ell)$ (compared to $O(\ell \log n)$ of the scheme from Theorem 3.2.

THEOREM 5.1. *Let $\ell = \Omega(\log n \log\log n)$. There exists a noninteractive broadcast $t$-out-of-$n$ secret sharing scheme for the family $\mathcal{BIT}_\ell$ in which the length of the secrets is $\ell$ and the length of the shares is $O(\ell)$.*

We first present a meta scheme (section 5.1). Then we show a possible implementation of the meta scheme that satisfies the conditions of the theorem.

### 5.1. A meta scheme.

Let $k$ and $h$ be integers (to be fixed later) such that $h \geq \lceil \log(n+1) \rceil$ and the secret $s$ has length $\ell \leq kh$. We view the secret as a binary matrix with $k$ rows and $h$ columns and denote its $(i,j)$ entry by $s[i,j]$. We construct, in a way to be specified below, a new binary matrix $H$ with $3k$ rows and $h$ columns. Then we share every row of $H$ using Shamir's $t$-out-of-$n$ scheme over $\mathrm{GF}(2^h)$. Since the length of every row is $h \geq \lceil \log(n+1) \rceil$, Shamir's scheme is ideal and every party gets $3k$ shares of length $h$. The distribution stage of this scheme is illustrated in Figure 5.1. When a set $B$ of cardinality (at least) $t$ wants to reconstruct the bit $s[i,j]$ of the secret, the parties in $B$ reconstruct a subset $T_{i,j}$ of rows (which depends only on $i,j$ and not on $B$). We will guarantee that these rows do not give any additional information on the secret.

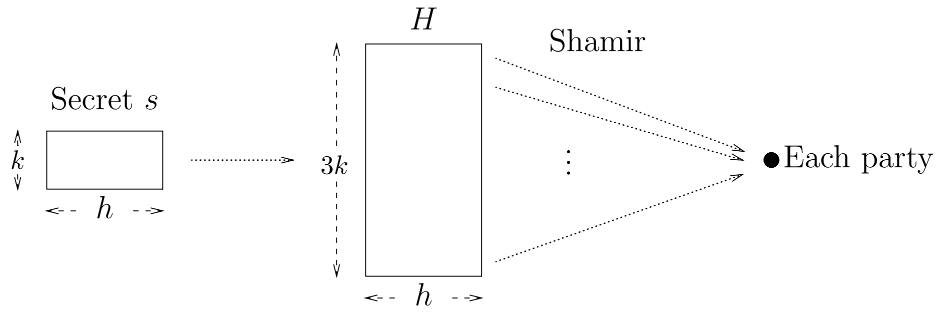


FIG. 5.1. *An illustration of the meta scheme of the noninteractive broadcast scheme for the bit family.*

More specifically, for every $1 \leq i \leq k$ and $1 \leq j \leq h$, we fix a set $T_{i,j} \subseteq \{1, \ldots, 3k\}$ (independently of the secret). The $j$th column of $H$ is constructed independently and depends only on the $j$th column of the secret. It is chosen uniformly at random among the column vectors such that

$$\forall\ 1 \leq i \leq k\ :\ \sum_{a \in T_{i,j}} H[a,j] = s[i,j]\ . \tag{3}$$

To reconstruct $s[i,j]$ it is enough to reconstruct the $T_{i,j}$-rows of $H$ and compute the sum (modulo 2) of the $j$th bit of the reconstructed rows. (Actually, every party can sum the shares of the $T_{i,j}$-rows of $H$, and then the parties reconstruct the secret from these shares. The $j$th bit of the reconstructed secret equals $s[i,j]$.) That is, the message of a party in a reconstructing set consists of the shares corresponding to the $T_{i,j}$-rows of $H$. The existence of a matrix $H$ satisfying (3), and the security requirement after reconstruction, depend on the choice of the sets $T_{i,j}$. On the other hand, independently of the choice of the sets $T_{i,j}$, any coalition of size less than $t$ prior to any reconstruction does not have any information on the rows of $H$ (by the properties of Shamir's scheme) and, hence, does not have any information on the secret. That is, the meta scheme is secure prior to the evaluation.

**5.2. Implementing the meta scheme.** We show how to construct sets $T_{i,j}$ such that the security requirement after reconstruction will hold. Let $R_1, R_2, \ldots, R_h \subset \{k+1, \ldots, 3k\}$ be a collection of different sets of size $k$. In particular, no set is contained in another set ($R_1, R_2, \ldots, R_h$ is a Sperner family [2]). For $i = 1, \ldots, k$, let $T_{i,j} \triangleq R_j \cup \{i\}$.

The number of sets of cardinality $k$ which are contained in $\{k+1, \ldots, 3k\}$ is $\binom{2k}{k} \geq 2^{2k-o(k)} \geq 2^k$. Thus, fixing $k \triangleq \lceil \log h \rceil$ suffices. We require that $h \lceil \log h \rceil = hk \geq \ell$, and we choose the smallest possible $h$. The length of the shares (i.e., $3h \lceil \log h \rceil$) is $\Theta(\ell)$. The following claims are useful for proving the security of a reconstruction.

CLAIM 5.1. *For every secret $s$, the number of matrices $H$ that satisfies* (3) *for all $j$ ($1 \leq j \leq h$) is at least* 1 *and is independent of $s$.*

*Proof.* Set $H[i,j] = s[i,j]$ for $1 \leq i \leq k$, and 0 otherwise. This matrix $H$ satisfies (3). To show that the number of matrices satisfying (3) is independent of $s$, notice that $H$ is a solution of a nonhomogeneous system of linear equations. Since the system has at least one solution, the number of such matrices is the number of solutions to the homogeneous linear system of equations, where every $s[i,j] = 0$. □

To show that no coalition gains any additional information, we first consider the case in which the coalition knows only the $T_{i,j}$-rows of $H$. In this case, the coalition can reconstruct $s[i,j]$. We prove that the coalition does not gain any information on the other bits of the secret (i.e., bits $s[i',j']$ for $i' \neq i$ or $j' \neq j$).

CLAIM 5.2. *Let $s$ be a secret and fix the $T_{i,j}$-rows of $H$ such that*

$$\sum_{a \in T_{i,j}} H[a,j] = s[i,j].$$

*The number of matrices $H$ that satisfies* (3) *for all $j$ ($1 \leq j \leq h$) is at least* 1 *and is independent of $s$.*

*Proof.* We first prove that there exists a matrix $H$ satisfying the requirements. Since the columns of $H$ are constructed independently, it is enough to prove the existence of every column $j'$. Only the $j'$th column of the $T_{i,j}$-rows of $H$, and the $j'$th column of $s$, influence the $j'$th column of $H$; hence, while considering the $j'$th column we can ignore the rest of the columns.

We first consider the $j$th column of $H$ (i.e., $j' = j$). Since $T_{i',j} = R_j \cup \{i'\}$, we can assign $H[i',j]$ a value as follows: For every $i'$ such that $k+1 \leq i' \leq 3k$ and $i' \notin R_j$, set $H[i',j] = 0$ (this is arbitrary); for every $i'$ such that $1 \leq i' \leq k$, set $H[i',j] = \sum_{a \in R_j} H[a,j] + s[i',j]$.

The case $j' \neq j$ is similar except that we have to be careful about $s[i,j']$. Since $R_{j'} \not\subseteq R_j$, there is an element $p \in T_{i,j'} \setminus T_{i,j}$. We can assign $H[i',j']$ a value as follows: For every $i'$ such that $k+1 \leq i' \leq 3k$, $i' \notin R_j$, and $i' \neq p$, set $H[i',j'] = 0$; $H[p,j'] = \sum_{a \in T_{i,j'} \setminus \{p\}} H[a,j'] + s[i,j']$. For every $i'$, $1 \leq i' \leq k$, and $i' \neq i$, set $H[i',j'] = \sum_{a \in R_{j'}} H[a,j'] + s[i',j']$.

We have shown that there exists a matrix as required. By the same arguments as in the proof of Claim 5.1, the number of possible matrices, given a secret $s$, is independent of $s$. ∎

We are now ready to prove the security requirement (note that this scheme is *not* reusable).

CLAIM 5.3. *The above implementation of the meta scheme is secure.*

*Proof.* Let $B$ be any set that evaluates a bit $s[i,j]$ of the secret and $C$ be any coalition. The security requirement before reconstruction is easy (this was discussed at the end of section 5.1). Suppose that $C$ has obtained the messages sent by the parties in $B$. That is, the parties in $C$ know the shares of the parties of $B$ corresponding to the $T_{i,j}$-rows of $H$. Therefore, the only information that they gain is the $T_{i,j}$-rows of $H$. Then, given the shares of the coalition and all the messages that were sent, every matrix $H$ that agrees with these rows is possible. We need to prove that, given two secrets in which the $(i,j)$th bit is the same, and a matrix $H$ in which only the $T_{i,j}$-rows are fixed, the probability that $H$ was constructed from each secret is equally likely. By Claims 5.1 and 5.2 this probability depends only on $s[i,j]$. ∎

## 6. Characterization of families with ideal schemes.

In this section we consider *ideal* secret sharing schemes for a family $\mathcal{F}$. We say that two secrets $s, s'$ are *distinguishable* by $\mathcal{F}$ if there exists a function $f \in \mathcal{F}$ for which $f(s) \neq f(s')$. A secret sharing scheme for $\mathcal{F}$ is *ideal* if the cardinality of the domain of shares equals the cardinality of the domain of distinguishable secrets. These are the shortest shares possible by [26]. The following theorem gives several impossibility results for ideal schemes for a family $\mathcal{F}$ which contains the bit functions.

THEOREM 6.1 (characterization theorem).

1. *Let* $2 \leq t \leq n$. *Suppose that there is an ideal* interactive *$t$-out-of-$n$ secret sharing scheme for a family of functions $\mathcal{F}$ such that $\mathcal{BIT}_\ell \subseteq \mathcal{F}$. Then, any* boolean *function $f \in \mathcal{F}$ is linear.*
2. *Let $t \in \{2, n\}$. Suppose that there is an ideal* noninteractive *$t$-out-of-$n$ secret sharing scheme for a family of functions $\mathcal{F}$ such that $\mathcal{BIT}_\ell \subseteq \mathcal{F}$. Then, $\mathcal{F} \subseteq \mathcal{LIN}_\ell$.*
3. *Let $3 \leq t \leq n-1$. In an ideal $t$-out-of-$n$ secret sharing the evaluation of every nonconstant boolean function requires interaction on private channels.*
4. *Let $t = 2$. In an ideal noninteractive 2-out-of-$n$ secret sharing every nonconstant boolean function cannot be evaluated via a broadcast channel.*

It follows that for families $\mathcal{F}$, with $\mathcal{BIT}_\ell \subseteq \mathcal{F}$, if interaction is allowed then every boolean function in $\mathcal{F}$ is linear. If interaction is not allowed, then either $t = 2$ or $t = n$, and all the functions in $\mathcal{F}$ are linear. In particular, when $t = 2$ private channels must be used.

In section 6.1 we will discuss some basic properties of ideal schemes that will be needed for the proof of this theorem. In section 6.2 we prove items 1 and 2 and in section 6.3 we prove items 3 and 4.

### 6.1. Properties of ideal schemes.

In this section we discuss some simple properties of ideal schemes which are useful in what follows.

PROPOSITION 6.2 (see [26]). *Fix the shares of any $t-1$ parties in an ideal $t$-out-of-$n$ secret sharing scheme. Then, the share of any other party $P$ is a permutation of* *the secret. In particular, all shares are possible for $P$.*

In our proof, we use the following proposition regarding private functions, that is, functions that can be computed without revealing any other information on the inputs. (For a formal definition of privacy see Appendix C.) Bivariate boolean private functions were characterized by Chor and Kushilevitz [16].

PROPOSITION 6.3 (see [16]). *Let $A_1, A_2$ be nonempty sets and $f : A_1 \times A_2 \rightarrow \{0,1\}$ be an arbitrary boolean function. Then, $f$ can be computed privately if and only if there exist boolean functions $f_1 : A_1 \rightarrow \{0,1\}$, $f_2 : A_2 \rightarrow \{0,1\}$ such that for every $x \in A_1,\ y \in A_2$ it holds that $f(x,y) = f_1(x) + f_2(y)$.*

CLAIM 6.1. *Let $f$ be a boolean function that can be evaluated securely in an ideal 2-out-of-2 secret sharing scheme. Then, there exist boolean functions $f_1$ and $f_2$ such that $f(s) = f_1(x) + f_2(y)$, where $x$ and $y$ are the shares of $P_1$ and $P_2$, respectively.*

*Proof.* The function $f$ is evaluated from $x$ and $y$, i.e., there is some function $f'$ such that $f(s) = f'(x,y)$. By Proposition 6.2 any information that a party gets on the share of the other party is translated into information on the secret. That is, the parties must compute $f'(x,y)$ in such a way that each party receives only information implied by $f'(x,y)$. In other words, they compute the boolean function $f'$ privately. By Proposition 6.3 there exist $f_1$ and $f_2$ such that $f(s) = f'(x,y) = f_1(x) + f_2(y)$. ∎

Consider an ideal $t$-out-of-$n$ secret sharing scheme for $\mathcal{F}$. Without loss of generality, we may assume that all the secrets are distinguishable by $\mathcal{F}$. So, the parties $P_1, P_2, \ldots, P_t$ can reconstruct the secret from their shares. Denote the reconstructing function by $h(s_1, s_2, \ldots, s_t)$. We next prove an important property of the messages sent in a noninteractive protocol that evaluates $f$.

CLAIM 6.2. *Let $f$ be a function that can be securely evaluated in an ideal $t$-out-of-$n$ secret sharing scheme without interaction. Without loss of generality, we may assume that all the messages sent by $P_i$, while holding the share $s_i$, to the other parties are identical and equal to*

$$f(h(\underbrace{0,\ldots,0}_{i-1}, s_i, \underbrace{0,\ldots,0}_{t-i})).$$

*Proof.* It is sufficient to prove the claim for $P_1$. Suppose that the shares of the parties $P_2, \ldots, P_t$ are $s_i = 0$. Consider a possible message of $P_1$ while the set $\{P_1, P_2, \ldots, P_t\}$ computes the value $f(s)$. Party $P_1$ might toss coins, so if there are several messages choose the first one lexicographically. When $P_1$ holds two shares $s_1$ and $s_1'$ such that $f(h(s_1, 0, \ldots, 0)) \neq f(h(s_1', 0, \ldots, 0))$, the messages of $P_1$ to $P_i$ have to be different; otherwise $P_i$ will compute an incorrect value in one of the cases. On the other hand, for two shares $s_1$ and $s_1'$ such that $f(h(s_1, 0, \ldots, 0)) = f(h(s_1', 0, \ldots, 0))$ the messages sent by $P_1$ to the parties in $\{P_2, \ldots, P_t\}$ have to be the same in both cases; otherwise the coalition $\{P_2, \ldots, P_t\}$ can distinguish between the secrets $s = h(s_1, 0, \ldots, 0)$ and $s' = h(s_1', 0, \ldots, 0)$, although $f(s) = f(s')$. Thus, without loss of generality, the messages sent by $P_1$ while holding the share $s_1$ are identical and equal to $f(h(s_1, 0, \ldots, 0))$. ∎

**6.2. Proofs of items 1 and 2 of Theorem 6.1.** The proofs of items 1 and 2 are similar and have two stages. We consider the following two-party distribution scheme over GF($2^\ell$), denoted XOR. Given a secret $s$, the dealer chooses at random $x \in$ GF($2^\ell$). The share of the first party is $x$, and the share of the second party is $y = s + x$. In the first stage (section 6.2.1) we characterize the functions that can be evaluated from the shares of XOR. In the second stage (section 6.2.2) we show that if there exists an ideal $t$-out-of-$n$ secret sharing scheme for $\mathcal{F}$, then XOR is a secret sharing scheme for $\mathcal{F}$. The combination of the two stages implies items 1 and 2.

**6.2.1. Characterizing the functions which can be evaluated with XOR.** We first characterize the functions that can be evaluated with XOR *without interaction.* It can be seen that every linear function of the secret can be evaluated with XOR (the details are the same as in Remark 3.1). We prove that no other functions can be evaluated securely without interaction. Then we characterize the boolean functions that can be evaluated with XOR *with interaction* and show that these functions are exactly the boolean linear functions.

To prove the characterization without interaction, we will prove that $f(s)$ can be computed from $f(x)$ and $f(y)$ (where $x$ and $y$ are the shares). In the next claim we prove that every function with this property is a renaming of a linear function.

CLAIM 6.3. *Let $f : \mathrm{GF}(2^\ell) \to \mathrm{GF}(2^\ell)$ be any function. If there exists a function $g$ such that $f(x+y) = g(f(x), f(y))$ for every $x, y \in \mathrm{GF}(2^\ell)$, then $f$ is a renaming of a linear function.*

*Proof.* Let $X_0 \triangleq \{x : f(x) = f(0)\}$. We first prove that

$$f(x_1) = f(x_2) \text{ if and only if } f(x_1 + x_2) = f(0). \tag{4}$$

The "only if" direction follows from $f(x_1+x_2) = g(f(x_1), f(x_2)) = g(f(x_1), f(x_1)) = f(x_1 + x_1) = f(0)$. Similarly, the "if" direction follows from the following simple equations:

$$f(x_1) = f(x_1+0) = g(f(x_1), f(0)) = g(f(x_1), f(x_1+x_2)) = f(x_1+x_1+x_2) = f(x_2).$$

That is, if $x_1, x_2 \in X_0$, then $x_1+x_2 \in X_0$. In other words, the set $X_0$ is a linear space over the field GF(2). Now consider a linear transformation $f' : \mathrm{GF}(2^\ell) \to \mathrm{GF}(2^\ell)$ whose null space is $X_0$ (that is, $f'(x) = 0$ if and only if $x \in X_0$). Since $X_0$ is a linear space, such a linear transformation exists. We claim that $f$ is a renaming of $f'$, i.e., $f(x) = f(y)$ if and only if $f'(x) = f'(y)$. By (4), $f(x) = f(y)$ if and only if $x+y \in X_0$. Furthermore, $x + y \in X_0$ if and only if $f'(x + y) = 0$ (by the definition of $f'$). Since $f'$ is linear, $f'(x) + f'(y) = f'(x + y)$. Hence, $f(x) = f(y)$ if and only if $f'(x) = f'(y)$. ☐

CLAIM 6.4. *Let $f$ be any function that can be evaluated without interaction with the scheme* XOR. *Then, $f$ is a renaming of a linear function.*

*Proof.* Assume, without loss of generality, that $f : \mathrm{GF}(2^\ell) \to \mathrm{GF}(2^\ell)$ (otherwise consider the function of $x$ defined as $\min\{y : f(y) = f(x)\}$, which is a renaming of $f$). By Claim 6.3 it suffices to prove that $f(x + y)$ can be computed from $f(x)$ and $f(y)$.

Denote the share of $P_1$ by $x$ and the share of $P_2$ by $y$. Recall that $s = x + y$. By Claim 6.2 the message sent by $P_2$ while holding a share $y$ to $P_1$ is $f(y)$, and the message sent by $P_1$ while holding a share $x$ to $P_2$ is $f(x)$. Hence, $P_1$ can compute $f(s) = f(x + y)$ from $x$ and $f(y)$. Moreover, for every two shares $x_1$ and $x_2$ held by $P_1$ such that $f(x_1) = f(x_2)$ and every share $y$ held by $P_2$, party $P_1$ must compute the same value of $f(s)$, since $P_2$ receives the same message from $P_1$ in both cases and therefore computes the same value of $f(s)$. Hence, $P_1$ can compute $f(x + y)$ from $f(x)$ and $f(y)$. The computed function is the desired function $g$ with $f(x + y) = g(f(x), f(y))$. By Claim 6.3 this implies that $f$ is a renaming of a linear function. ☐

We now prove a similar claim for interactive evaluations. However, this claim applies only to boolean functions.

Claim 6.5. *Let $f$ be a* boolean *function that can be evaluated interactively with the scheme* XOR. *Then, $f$ is a renaming of a linear function.*

*Proof.* Since XOR is an ideal scheme, Claim 6.1 implies that there exist boolean functions $f_1, f_2$ such that $f(s) = f(x+y) = f_1(x)+f_2(y)$. In particular, $f(x) = f(x+0) = f_1(x) + f_2(0)$. Similarly $f(y) = f_1(0) + f_2(y)$. So, there exists a bit $b$ such that $f(x + y) = f(x) + f(y) + b$. Then, $f'(x) = f(x) + b$ is a linear function which is a renaming of $f$. □

The exact family of functions which can be evaluated interactively with XOR consists of the two-argument functions for which $f(s) = f(x + y)$ can be computed privately from $x$ and $y$ (as characterized in [29]).

**6.2.2. Reduction to XOR.** In this section we prove that if there exists an ideal $t$-out-of-$n$ secret sharing scheme for a family $\mathcal{F}$, then there exists an ideal 2-out-of-2 secret sharing scheme for $\mathcal{F}$. We then prove that this implies that XOR is a secret sharing scheme for $\mathcal{F}$.

Claim 6.6. *If there exists an interactive (respectively, noninteractive) ideal $t$-out-of-$n$ secret sharing scheme for $\mathcal{F}$, then there exists an interactive (respectively, noninteractive) ideal 2-out-of-2 secret sharing scheme for $\mathcal{F}$.*

*Proof.* Let $B = \{P_1, P_2, \ldots, P_t\}$ be a set of parties. We ignore the shares distributed to parties not in $B$. Therefore, we have an ideal $t$-out-of-$t$ secret sharing scheme for $\mathcal{F}$. Let $\langle s_1, s_2, s_3, \ldots, s_t\rangle$ be any (fixed) vector of shares that is dealt to $B$ with positive probability. To share a secret $s$ the dealer now generates a random vector of shares of $s$ (according to the scheme) that agrees with $s_3, \ldots, s_t$, respectively, and gives the first two components of this vector to $P_1$ and $P_2$. Parties $P_1$ and $P_2$ know the shares of the other parties (as they are fixed) and therefore $P_1$, say, can simulate the parties $P_3, \ldots, P_t$ in the protocol which evaluates the function $f \in \mathcal{F}$. Hence, $P_1$ and $P_2$ can evaluate $f(s)$ from their shares and the messages they exchange. On the other hand, $P_1$ has no more information than the information known to the coalition $\{P_1, P_3, \ldots, P_t\}$ of cardinality $t-1$ in the $t$-out-of-$t$ scheme. So, $P_1$ does not gain additional information on $s$. Similar arguments hold for $P_2$. This implies that this scheme is a 2-out-of-2 secret sharing scheme for $\mathcal{F}$. □

Claim 6.7. *Let $\mathcal{BIT}_\ell \subseteq \mathcal{F}$. If there exists an interactive (respectively, noninteractive) ideal $t$-out-of-$n$ secret sharing scheme for $\mathcal{F}$, then* XOR *is an interactive (respectively, noninteractive) secret sharing scheme for $\mathcal{F}$.*

*Proof.* By Claim 6.6 we can assume that there exists an ideal 2-out-of-2 secret sharing scheme for $\mathcal{F}$, say $\Pi$. We transform $\Pi$ into XOR in a way that every function that can be evaluated in $\Pi$ can also be evaluated in XOR.

Claim 6.1 implies that for every $j$, where $1 \le j \le \ell$, there exist boolean functions $f_1^j, f_2^j$ such that $e_j(s) = f_1^j(x)+f_2^j(y)$. For $i = 1, 2$, define $\overline{m}_i(x)$ as the concatenation of the values of these functions, that is, $\overline{m}_i(x) \overset{\triangle}{=} f_i^1(x) \circ f_i^2(x) \circ \cdots \circ f_i^\ell(x)$. Then, for every secret $s$ and random input $r$ of the dealer, $s = \overline{m}_1(\Pi_1(s,r)) + \overline{m}_2(\Pi_2(s,r))$. Therefore, the scheme $\Pi'$ defined by $\Pi'(s,r) = \langle \overline{m}_1(\Pi_1(s,r)), \overline{m}_2(\Pi_2(s,r))\rangle$ is a distribution scheme equivalent to XOR. We still have to show that the two parties can evaluate securely every function in $\mathcal{F}$ with $\Pi'$.

We first prove that $\overline{m}_1$ is invertible. Clearly, the two parties can reconstruct the secret in $\Pi'$, while every single party knows nothing about the secret. Therefore, $\Pi'$ is a 2-out-of-2 secret sharing scheme and, by [26], the cardinality of the domain of shares is at least the cardinality of the domain of secrets. Thus, the cardinality of the range of $\overline{m}_1$ is at least as large as $2^\ell$. The domain of $\overline{m}_1$, which is the set of shares of $P_1$, has cardinality $2^\ell$. Therefore, $\overline{m}_1$ is a bijection and, hence, invertible. So, $P_1$

can reconstruct the share $x$ from $\overline{\mathrm{m}}_1(x)$. Similarly, $P_2$ can reconstruct the share $y$ from $\overline{\mathrm{m}}_2(y)$. This implies that the parties $P_1$ and $P_2$, while holding $\overline{\mathrm{m}}_1(x)$ and $\overline{\mathrm{m}}_2(y)$, respectively, can evaluate every function $f \in \mathcal{F}$. We have proved that every function $f \in \mathcal{F}$ can be evaluated with $\Pi'$ which is equivalent to XOR. Furthermore, if the evaluation of the original scheme required no interaction, then also the evaluation with the XOR scheme requires no interaction. $\square$

**6.3. Proofs of items 3 and 4 of Theorem 6.1.** Next we prove item 4. Namely, in any ideal noninteractive $t$-out-of-$n$ secret sharing scheme ($2 \leq t \leq n-1$), evaluation of a boolean function requires private channels. The proof is by contradiction. Suppose that there is an ideal noninteractive $t$-out-of-$n$ scheme in which the evaluation can be held on a broadcast channel. Then, by the same arguments as in Claim 6.6, since $t \leq n-1$, there is an ideal 2-out-of-3 scheme with the same property. Our first claim is that, without loss of generality, in the evaluation of $f(s)$ every party sends a one-bit message such that the sum of the messages is $f(s)$.

Claim 6.8. *Assume there exists an ideal 2-out-of-3 secret sharing scheme in which a nonconstant boolean function $f$ can be securely evaluated without interaction via a broadcast channel. Then parties $P_1$ and $P_2$ can securely compute $f(s)$ by sending one-bit messages $m_1$ and $m_2$, such that $m_1 + m_2 = f(s)$.*

*Proof.* By Claim 6.2 we can assume that the message of $P_1$ while holding the share $s_1$ is the bit $m_1 = f(h(s_1, 0))$. Similarly, we assume that the message of $P_2$ while holding the share $s_2$ is $m_2 = f(h(0, s_2))$. Furthermore, the value of $f(s)$ is determined by these two messages. There are two values for $f(s)$ and two values for each message. Since each party knows nothing about $f(s)$, a change of any message will result in a change of $f(s)$. The only boolean functions with two binary variables satisfying this requirement are $m_1 + m_2 + b$ with either $b = 0$ or $b = 1$. So, without loss of generality, assume that $P_1$ sends the message $m_1 + b$, and the claim follows. $\square$

Claim 6.9. *Consider an ideal noninteractive 2-out-of-3 secret sharing scheme for a family $\mathcal{F}$ which contains a nonconstant boolean function $f$. Then, $f$ cannot be evaluated via a broadcast channel.*

*Proof.* Fix the share of $P_3$ to be $z = 0$. By Proposition 6.2, the share $x$ of $P_1$ is a permutation of the secret. Assume, without loss of generality, that this share is equal to the secret. (Of course $P_1$ does not know that $z = 0$ and does not know the secret.) Since $P_3$ does not know $f(s)$ although $z = 0$, the value of $f(s)$ is not fixed, and the message that $P_1$ sends to $P_2$ while $P_1$ and $P_2$ evaluate $f(s)$ cannot be constant (otherwise $P_2$ will not be able to evaluate $f(s)$). Furthermore, for any two shares $x, x'$ held by $P_1$ such that $f(x) = f(x')$, the messages of $P_1$ to $P_2$ should be the same (otherwise $P_3$ will distinguish between the two secrets). Without loss of generality, the message $m_1$ which $P_1$ sends to $P_2$ is $f(x)$. By Claim 6.8, $f(s) = m_1 + m_2$. Thus, the message that $P_2$ sends to $P_1$ is constant and $P_1$ will not be able to evaluate $f(s)$, which leads to a contradiction. Hence, $f(s)$ cannot be evaluated on a broadcast channel. $\square$

We conclude this section by proving item 3. That is, for $3 \leq t \leq n-1$, the evaluation requires interaction via private channels. Again, we assume for the contradiction that there exists an ideal noninteractive $t$-out-of-$n$ scheme for the family $\mathcal{F}$, and we construct an ideal noninteractive 3-out-of-4 scheme for the family $\mathcal{F}$. We show that this contradicts Claim 6.9.

Claim 6.10. *Assume there is an ideal 3-out-of-4 secret sharing scheme for a family $\mathcal{F}$ in which a function $f \in \mathcal{F}$ can be evaluated via private channels without* *interaction. Then, there is an ideal* 2*-out-of-*3 *secret sharing scheme in which $f$ can be evaluated via a broadcast channel.*

*Proof.* By Claim 6.2 we may assume that the messages sent during the evaluation by one party to the other two parties are identical. In particular, if $\{P_i, P_j, P_k\}$ evaluates a function, then $P_i$ knows the messages that $P_j, P_k$ exchange. To construct a 2-out-of-3 secret sharing scheme, we fix a share $u$ for $P_4$ and share the secret $s$ with a random vector of shares such that the share of $P_4$ is $u$. When $P_i$ and $P_j$ want to evaluate $f(s)$, where $1 \leq i < j \leq 3$, then $P_i$ simulates $P_4$. All messages are exchanged on a broadcast channel. After this evaluation, every party $P_\ell$, where $\ell \in \{1, 2, 3\}$, will have the same information as the coalition $\{P_\ell, P_4\}$ had in the original scheme. Hence, $P_\ell$ does not gain any information that is not implied by $f(s)$. That is, we get a 2-out-of-3 secret sharing scheme for $\mathcal{F}$. ∎

**7. Discussion.** In this section we consider possible extensions of our work. So far, we considered only *threshold* secret sharing schemes. In [22] a general notion of secret sharing schemes for arbitrary collections of reconstructing sets is defined. That is, we are given a collection $\mathcal{A}$ of sets of parties called an *access structure* and require that any set in $\mathcal{A}$ can reconstruct the secret, while any set not in $\mathcal{A}$ does not know anything about the secret. Clearly, secret sharing schemes exist only for monotone collections. Conversely, it is known that for every monotone collection there exists a secret sharing scheme [22]. More efficient schemes for general access structures were presented (e.g., in [7, 36]). However, the length of the shares in these schemes can be exponential in the number of parties (i.e., of length $\ell 2^{\Theta(n)}$, where $n$ is the number of parties in the system and $\ell$ is the length of the secret). Our definition of secret sharing for a family $\mathcal{F}$ of functions can be naturally generalized to an arbitrary access structure. To construct such schemes, observe that the schemes of [22, 7, 36] are "linear": the share of each party is a vector of elements over some field, and every set in $\mathcal{A}$ reconstructs the secret using a linear combination of elements in their shares. Thus, if we share every bit of the secret independently, we can evaluate every linear function of the secret without any interaction (the details are as in Theorem 3.2). This implies that for every access structure, there exists a scheme for the family $\mathcal{ALL}_\ell$ in which the length of the shares is $O(2^\ell 2^n)$. However, if the access structure has a more efficient linear scheme for sharing a single bit, then the length of the shares can be shorter (but at least $2^\ell$).

Another possible extension of the definition is by allowing a weaker "nonperfect" definition of security. Such extensions in the context of traditional secret sharing schemes were considered, e.g., in [9]. As remarked earlier there are several related works on *simultaneous* secret sharing. For example, in [23] each set of cardinality at least $t$ should be able to reconstruct some of the secrets (but not all). In [11] each set $B$ of cardinality at least $t$ is able to reconstruct any of the distributed secrets.[6] The security requirement considered in [11, 24, 10] is somewhat weak: after revealing one of the secrets, some limited information about other secrets may be leaked.

Our scheme for the family of functions $\mathcal{ALL}_\ell$ uses shares of size $2^\ell$. We can construct an efficient scheme if we relax the security requirement as follows: Every set of size at least $t$ can evaluate any function $f(s)$ in a way that every coalition of at most $\lfloor (t-1)/2 \rfloor$ parties gains no information on the secret (but larger coalitions may get some information). In this case, we can use Shamir's $t$-out-of-$n$ scheme to distribute the secret and the interactive protocol of [5] or [15] to securely compute

[6]In fact [11] deals with more general *access structures* and not only threshold schemes.

$f(s)$. In this scenario, the length of the share is the same as the length of the secret. However, for most functions $f$, the length of the communication during the evaluation of $f$ is exponential in $\ell$.

**Appendix A. Equivalence of definitions of security.** An alternative definition to our definition of security of secret sharing for $\mathcal{F}$ requires that after evaluating $f(s)$, the probability of the secret given the view of the coalition is the same as the probability of the secret given only the value $f(s)$. We prove that these two definitions are equivalent for private channels schemes.

CLAIM A.1. *Let $X$ be any random variable on the set of secrets $S$ such that $\Pr[\,X = s\,] > 0$ for every $s$. A private channels secret sharing scheme for the family $\mathcal{F}$ is secure (according to Definition* 2.5*) if and only if, for any value $v$ in the range of $f$, any secret $s \in S$ such that $f(s) = v$, any coalition $C$ of size less than $t$ such that $B \cap C \neq \emptyset$, and any value* view *of $VIEW_C$ such that* $\Pr[\,VIEW_C = \text{view}\,] > 0$,

$$\Pr[\,X = s \mid VIEW_C = \text{view}\,] \quad = \quad \Pr[\,X = s \mid f(X) = v\,]\,, \tag{5}$$

*where the probability is taken over the distribution of the secrets, the random input of the dealer, and the random inputs of the parties.*[7]

*Proof.* Assume that the scheme is secure according to Definition 2.5, that is, for every $s, s' \in S$ such that $f(s) = f(s')$: $\Pr[\text{VIEW}_C = \text{view} \mid X = s] = \Pr[\text{VIEW}_C = \text{view} \mid X = s']$. Then, $\Pr[\text{VIEW}_C = \text{view} \mid X = s] = \Pr[\text{VIEW}_C = \text{view} \mid f(X) = v]$. Using Bayes' rule and the previous equation, we get

$$\begin{aligned}
&\Pr[X = s \mid \text{VIEW}_C = \text{view}]\\
&\qquad = \frac{\Pr[\text{VIEW}_C = \text{view} \mid X = s] \cdot \Pr[X = s]}{\Pr[\text{VIEW}_C = \text{view}]}\\
&\qquad = \frac{\Pr[\text{VIEW}_C = \text{view} \mid f(X) = v] \cdot \Pr[X = s]}{\Pr[\text{VIEW}_C = \text{view}]}\\
&\qquad = \frac{\Pr[f(X) = v \mid \text{VIEW}_C = \text{view}] \cdot \Pr[X = s]}{\Pr[f(X) = v]}\\
&\qquad = \Pr[f(X) = v \mid \text{VIEW}_C = \text{view}] \cdot \Pr[X = s \mid f(X) = v]\ .
\end{aligned}$$

For the last equality recall that $f(s) = v$, i.e., $\Pr[X = s] = \Pr[(X = s) \wedge (f(X) = v)]$. Furthermore, each party in $B$ can evaluate $f(X)$ from the information it has. Since $B \cap C \neq \emptyset$, the probability $\Pr[f(X) = v \mid \text{VIEW}_C = \text{view}]$ is 1. Hence, the security according to Definition 2.5 implies (5).

The proof in the other direction is similar. Assume that (5) holds. Using Bayes' rule we see that for every $s$ such that $f(s) = v$,

$$\begin{aligned}
\Pr[\text{VIEW}_C = \text{view} \mid X = s] &= \frac{\Pr[f(X) = v \mid X = s] \cdot \Pr[\text{VIEW}_C = \text{view}]}{\Pr[f(X) = v]}\\
&= \frac{\Pr[\text{VIEW}_C = \text{view}]}{\Pr[f(X) = v]}\ .
\end{aligned}$$

For the last equality notice that $\Pr[f(X) = v \mid X = s] = 1$ (since $f(s) = v$). Thus, $\Pr[\text{VIEW}_C = \text{view} \mid X = s]$ is independent of $s$, which implies the security of the scheme. ∎

[7]For broadcast schemes we have to deal also with coalitions $C$ such that $B \cap C = \emptyset$. For these coalitions (5) does not necessarily hold, as they might not learn the value $f(s)$ (but will learn some information on $f(s)$).

**Appendix B. Shamir's $t$-out-of-$n$ threshold scheme.** For the sake of completeness we describe the $t$-out-of-$n$ secret sharing scheme of Shamir [34]. In this scheme, the set of secrets is identified with the elements of the field $\mathrm{GF}(q)$, where $q$ is a prime power larger than $n$, the number of parties in the system. To share a secret $s \in \mathrm{GF}(q)$ the dealer chooses independently with uniform distribution $t-1$ random elements $r_1, \ldots, r_{t-1}$ in $\mathrm{GF}(q)$. These elements and the secret $s$ define the polynomial $p(x) = r_{t-1}x^{t-1} + r_{t-2}x^{t-2} + \cdots + r_1 x + s$ , for which $p(0) = s$. The dealer gives to each party $P_j$ the share $s_j = p(j)$. Any set $B$ of at least $t$ parties can now reconstruct $p(x)$ by interpolation,

$$s = p(0) = \sum_{P_j \in B} s_j \prod_{P_i \in B;\, i \neq j} \frac{i}{i-j}.$$

Observe that the secret $s$ is a linear combination of the shares $s_j$ of the parties $P_j \in B$, for which the coefficient of $s_j$ is

$$\delta_{B,j} \triangleq \prod_{P_i \in B;\, i \neq j} \frac{i}{i-j}.$$

**Appendix C. Definition of private computations.** We define *private protocols*, which are used for evaluating functions of the secrets securely. The model of computation is the private channels model described in Definitions 2.2 and 2.3. We say that a coalition $C$ *gains no additional information* from the execution of a randomized protocol $F$ which computes $f$ if the view of $C$ does not imply any information other than what follows from its input and the function value. That is, for every two input vectors $\langle x_i \rangle, \langle x'_i \rangle$ of $F$ for which $\langle x_i \rangle_C = \langle x'_i \rangle_C$ (i.e., which agree in their $C$ entries) and $f(\langle x_i \rangle) = f(\langle x'_i \rangle)$, for every choice of random inputs $\langle r_i \rangle_C$ for $C$, and every value of the messages $M_C(\langle x_i \rangle, \langle r_i \rangle)$ and $M_C(\langle x'_i \rangle, \langle r_i \rangle)$ received by $C$,

$$\Pr[\; M_C(\langle x_i \rangle, \langle r_i \rangle) \circ \langle r_i \rangle_C \;\mid\; \langle x_i \rangle \;] \;=\; \Pr[\; M_C(\langle x'_i \rangle, \langle r_i \rangle) \circ \langle r_i \rangle_C \;\mid\; \langle x'_i \rangle \;] \;.$$

A protocol $F$ computing $f$ is *private* if any coalition $C$ of size at most $n-1$ gains no additional information from the execution of the protocol. A function $f$ is *private* if there is a private protocol which computes it. It follows that if a private protocol is used to reconstruct the secret, then the reconstruction is secure.

In our interactive secret sharing scheme for the linear functions we use the private protocol of Benaloh [6] to compute the sum over $\mathrm{GF}(q)$ of the shares. For the sake of completeness we describe this protocol below:

**Private addition over $\mathrm{GF}(q)$: $x = \sum_{i=1}^{n} x_i$, $x_i \in \mathrm{GF}(q)$.**

```
Each $P_i$ with input $x_i$, where $i = 1, \ldots, n$:
    chooses $n-1$ random inputs from $\mathrm{GF}(q)$ denoted $r_{i,1}, \ldots, r_{i,n-1}$.
    computes $r_{i,n} \triangleq x_i - \sum_{j=1}^{n-1} r_{i,j}$.
    sends $r_{i,j}$ to $P_j$ (for every $j$).
Each $P_j$ computes $y_j = \sum_{i=1}^{n} r_{i,j}$ and sends it to $P_1$.
Party $P_1$ computes $\sum_{j=1}^{n} y_j = x$ and sends it to the other parties.
```

**Acknowledgments.** The authors would like to thank the referees for several helpful comments.

## REFERENCES

[1] M. Abadi, J. Feigenbaum, and J. Kilian, *On hiding information from an oracle*, J. Comput. System Sci., 39 (1989), pp. 21–50.

[2] M. Aigner, *Combinatorial Theory*, Springer-Verlag, Berlin, 1979.

[3] D. Beaver and J. Feigenbaum, *Hiding instances in multioracle queries*, in STACS '90, Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science, C. Choffrut and T. Lengauer, eds., Lecture Notes in Comput. Sci. 415, Springer-Verlag, New York, 1990, pp. 37–48.

[4] A. Beimel and B. Chor, *Universally ideal secret sharing schemes*, IEEE Trans. Inform. Theory, 40 (1994), pp. 786–794.

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson, *Completeness theorems for noncryptographic fault-tolerant distributed computations*, in Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, ACM, New York, 1988, pp. 1–10.

[6] J. Benaloh, *Secret sharing homomorphisms: Keeping shares of a secret secret*, in Advances in Cryptology - CRYPTO '86, A. M. Odlyzko, ed., Lecture Notes in Computer Sci. 263, Springer-Verlag, New York, 1987, pp. 251–260.

[7] J. Benaloh and J. Leichter, *Generalized secret sharing and monotone functions*, in Advances in Cryptology - CRYPTO '88, S. Goldwasser, ed., Lecture Notes in Comput. Sci. 403, Springer-Verlag, New York, 1990, pp. 27–35.

[8] G. R. Blakley, *Safeguarding cryptographic keys*, in Proceedings of the 1979 AFIPS National Computer Conference, R. E. Merwin, J. T. Zanca, and M. Smith, AFIPS Conference Proceedings 48, AFIPS Press, Montvale, NJ, 1979, pp. 313–317.

[9] G. R. Blakley and C. Meadows, *The security of ramp schemes*, in Advances in Cryptology - CRYPTO '84, G. R. Blakley and D. Chaum, eds., Lecture Notes in Comput. Sci. 196, Springer-Verlag, New York, 1985, pp. 242–268.

[10] C. Blundo, A. De Santis, G. Di Crescenzo, A. Giorgio Gaggia, and U. Vaccaro, *Multi-secret sharing schemes*, in Advances in Cryptology - CRYPTO '94, Y. Desmedt, ed., Lecture Notes in Comput. Sci. 839, Springer-Verlag, New York, 1994, pp. 150–163.

[11] C. Blundo, A. De Santis, and U. Vaccaro, *Efficient sharing of many secrets*, in STACS '93, P. Enjalbert, A. Finkel, and K. W. Wagner, eds., Lecture Notes in Comput. Sci. 665, Springer-Verlag, New York, 1993, pp. 692–703.

[12] E. F. Brickell, *Some ideal secret sharing schemes*, J. Combin. Math. Combin. Comput., 6 (1989), pp. 105–113.

[13] E. F. Brickell and D. M. Davenport, *On the classification of ideal secret sharing schemes*, J. Cryptology, 4 (1991), pp. 123–134.

[14] E. F. Brickell and D. R. Stinson, *Some improved bounds on the information rate of perfect secret sharing schemes*, J. Cryptology, 5 (1992), pp. 153–166.

[15] D. Chaum, C. Crépeau, and I. Damgard, *Multiparty unconditionally secure protocols*, in Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, ACM, New York, 1988, pp. 11–19.

[16] B. Chor and E. Kushilevitz, *A zero-one law for Boolean privacy*, SIAM J. Discrete Math., 4 (1991), pp. 36–47.

[17] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, *How to share a function securely*, in Proceedings of the 26th Annual ACM Symposium on the Theory of Computing, ACM, New York, 1994, pp. 522–533.

[18] Y. Desmedt, *Threshold cryptosystems*, in Advances in Cryptology - AUSCRYPT '92, J. Seberry and Y. Zheng, eds., Lecture Notes in Comput. Sci. 718, Springer-Verlag, New York, 1993, pp. 5–14.

[19] Y. Desmedt and Y. Frankel, *Shared generation of authenticators and signatures*, in Advances in Cryptology - CRYPTO '91, J. Feigenbaum, ed., Lecture Notes in Comput. Sci. 576, Springer-Verlag, New York, 1992, pp. 457–469.

[20] Y. G. Desmedt and Y. Frankel, *Homomorphic zero-knowledge threshold schemes over any finite abelian group*, SIAM J. Discrete Math., 7 (1994), pp. 667–679.

[21] M. K. Franklin and M. Yung, *Communication complexity of secure computation*, in Proceedings of the 24th Annual ACM Symposium on the Theory of Computing, ACM, New York, 1992, pp. 699–710.

[22] M. Ito, A. Saito, and T. Nishizeki, *Secret sharing schemes realizing general access structure*,

in Proceedings of the IEEE Global Telecommunication Conference, Globecom '87, IEEE Press, Piscataway, NJ, 1987, pp. 99–102. Journal version: *Multiple assignment scheme for sharing secret*, J. Cryptology, 6 (1993), pp. 15–20.

[23] W. Jackson, K. M. Martin, and C. M. O'Keefe, *Multisecret threshold schemes*, in Advances in Cryptology - CRYPTO '93, D. R. Stinson, ed., Lecture Notes in Comput. Sci. 773, Springer-Verlag, New York, 1994, pp. 126–135.

[24] W. Jackson, K. M. Martin, and C. M. O'Keefe, *On sharing many secrets*, in ASIACRYPT '94, J. Pieprzyk and R. Safavi-Naini, eds., Lecture Notes in Comput. Sci. 917, Springer-Verlag, New York, 1995, pp. 42–54.

[25] W. Jackson, K. M. Martin, and C. M. O'Keefe, *Ideal secret sharing schemes with multiple secrets*, J. Cryptology, 9 (1996), pp. 233–250.

[26] E. D. Karnin, J. W. Greene, and M. E. Hellman, *On secret sharing systems*, IEEE Trans. Inform. Theory, 29 (1983), pp. 35–41.

[27] J. Kilian and N. Nisan, *Private communication*, 1990.

[28] S. C. Kothari, *Generalized linear threshold scheme*, in Advances in Cryptology - CRYPTO '84, G. R. Blakley and D. Chaum, eds., Lecture Notes in Comput. Sci. 196, Springer-Verlag, New York, 1985, pp. 231–241.

[29] E. Kushilevitz, *Privacy and communication complexity*, SIAM J. Discrete Math., 5 (1992), pp. 273–284.

[30] R. J. McEliece and D. V. Sarwate, *On sharing secrets and Reed-Solomon codes*, Comm. ACM, 24 (1981), pp. 583–584.

[31] M. O. Rabin, *Randomized Byzantine generals*, in Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 1983, pp. 403–409.

[32] R. L. Rivest, L. Adleman, and M. L. Dertouzos, *On data banks and privacy homomorphisms*, in Foundations of Secure Computation, R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, eds., Academic Press, New York, 1978, pp. 169–179.

[33] P. D. Seymour, *On secret-sharing matroids*, J. Combin. Theory Ser. B, 56 (1992), pp. 69–73.

[34] A. Shamir, *How to share a secret*, Comm. ACM, 22 (1979), pp. 612–613.

[35] G. J. Simmons, *An introduction to shared secret and/or shared control and their application*, in Contemporary Cryptology, The Science of Information Integrity, G. J. Simmons, ed., IEEE Press, Piscataway, NJ, 1992, pp. 441–497.

[36] G. J. Simmons, W. Jackson, and K. M. Martin, *The geometry of shared secret schemes*, Bull. Inst. Combin. Appl., 1 (1991), pp. 71–88.

[37] D. R. Stinson, *An explication of secret sharing schemes*, Des. Codes Cryptogr., 2 (1992), pp. 357–390.

SIAM J. Discrete Math. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 3, pp. 346–353

# AN OPTIMIZATION PROBLEM IN STATISTICAL DATABASES*

LJILJANA BRANKOVIĆ†, PETER HORAK‡, AND MIRKA MILLER†

**Abstract.** Let $D = \{a_1, \dots , a_n\}$ be a set of real numbers, and let $S \subset \{1, \dots , n\}$. For an interval $I \subset \{1, \dots , n\}$ we set $SUM(I) = \sum_{i \in I} a_i$. In this paper we solve the following problem which has been asked in connection with security of statistical databases: Find a largest family **B** of subintervals of $\{1, \dots , n\}$ so that knowing the value of $SUM(I)$ for all $I \in \mathbf{B}$ does not enable one to calculate any element $a_i \in D$, where $i \in S$.

**Key words.** optimization, database security, sum queries

**AMS subject classifications.** 68R05, 68P15

**PII.** S0895480198345004

## 1. Introduction.

The optimization problem studied in this paper comes from the area of database security. For the convenience of readers not closely familiar with databases we start with a description of the original problem and only later provide its mathematical formulation. At the same time, this will enable us to explain, from a real-life point of view, which generalizations of the stated problem are important for database security.

Let $D = \{a_1, \dots , a_n\}$ be a (1-dimensional) statistical database. The owner of the database wants to provide a user with as much information from $D$ as possible but, at the same time, does not want to disclose the values of some elements of $D$. For the sake of an example, let an element $a_i$, $1 \leq i \leq n$, contain the sum of salaries of all employees working for a company for $i$ years. Elements of $D$ containing the sum of salaries of one employee, more than one employee, and no employee (= nobody works for the company for that particular given number of years; then the corresponding element of $D$ is set to equal 0) are called *single elements, multielements*, and *holes*, respectively. Without loss of generality, we can assume that there are no holes in $D$, otherwise, for practical reasons, they are removed from $D$. Information about salaries of individual employees is confidential. The owner cannot allow a *compromise* of the database; i.e., the owner should not enable the user to learn, directly or indirectly, the value of any single element $a_i$ of $D$. (The owner may disclose, i.e., the user may learn, the value of any multielement, as it is not possible for the user to infer the salary of any individual employee whose income is included in $a_i$.) The user is allowed to ask only statistical types of questions. The most frequent scenario is that the user chooses a query $Q = \{a_t; t \in T \subset \{1, \dots , n\}\}$ and from the owner gets an answer $SUM(T) = \sum_{t \in T} a_t$ and the cardinality of $T$ in order to calculate the average salary of the corresponding group of employees. Of course, knowing the values of $SUM(T)$ for some family of sets $T$ may allow the user to infer the value of a single element. One possible way to prevent this is to choose in advance a family **B** of queries which the owner will answer; the owner will refuse to answer queries not in **B**. Thus the

*Received by the editors September 21, 1998; accepted for publication (in revised form) March 2, 2000; published electronically May 26, 2000.
http://www.siam.org/journals/sidma/13-3/34500.html

†Department of Computer Science and Software Engineering, The University of Newcastle, Newcastle, NSW 2308, Australia (lbrankov@cs.newcastle.edu.au, mirka@cs.newcastle.edu.au).

‡Department of Mathematics, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait (horak@KUC01.kuniv.edu.kw).

problem is to find a largest family **B** of queries which does not lead to a compromise of the database. Equivalently, let $D = \{a_1, \dots, a_n\}$ be a set of real numbers, and let $S \subset \{1, \dots, n\}$. Find a largest family **B** of subsets $T$ of $\{1, \dots, n\}$ so that knowing the value of $SUM(T)$ for all $T \in \mathbf{B}$ does not enable one to calculate any element $a_i \in D$, where $i \in S$.

The problem above has been solved in [5] for databases whose elements are all single. It has been shown there that the cardinality of a largest family of queries which does not lead to a compromise in this case equals $\binom{m}{\lfloor \frac{m}{2} \rfloor}$. The uniqueness of the family described there has been proved in [3]. As already mentioned in [1], if $D$ has $n$ elements and $|S| = m$, then the cardinality of the largest family **B** of queries not compromising $D$ equals $\binom{m}{\lfloor \frac{m}{2} \rfloor} \times 2^{n-m}$ (for the sake of completeness we provide a proof of the statement at the end of section 3). The usability of a statistical database is defined to be the ratio of the cardinality of a largest family **F** of queries that can be answered without a compromise to the number of all posable queries. Thus, the result stated above implies that usability of a database when all queries are posable equals $\binom{m}{\lfloor \frac{m}{2} \rfloor} \times 2^{-m}$; that is, it tends to 0 for $m \to \infty$ . Clearly, a user is not interested in using a database if he or she does not get an answer to almost all his or her queries. The only possible way to increase the usability is to restrict the family of posable queries. With respect to real-life problems, we consider in this paper as posable queries the so-called range queries. A query $Q = \{a_t, t \in T\}$ is a range query if $T$ is a subinterval of $\{1, \dots, n\}$. In our example, a range query reads as follows: What is the sum of salaries (the average salary) of employees working for the company for $r \in [a, b]$ years? The range queries were discussed, for example, in [2].

In this paper we find a largest family of range queries which does not lead to a compromise of a 1-dimensional database $D$ and, consequently, determine the usability of $D$.

**2. Preliminaries.** In this section we deal only with range queries, that is, in a sense, with ordered subsets. Thus, throughout this section, we will consider a database $D = (a_1, \dots, a_n)$ to be an ordered set. Then range queries are contiguous subsets of $D$.

Let $T$ be a subset of $\{1, \dots, n\}$. Then $T$ can be expressed as a union of disjoint intervals of $\{1, \dots, n\}$,

$$T = \bigcup_{j=1}^{k} [b_j, c_j],$$

if, for each $j = 1, \dots, k-1$, it is $c_j + 1 < b_{j+1}$; that is, if we have a representation of $T$ in the above form with the minimum possible number of intervals, then the representation will be called the *minimum interval representation* of $T$ (MIR of $T$).

Consider a database $D = (a_1, \dots, a_n)$. Let $S \subset \{1, \dots, n\}$ be the set of indices of those elements which are single. We recall that revealing an element $a_i$ of $D$ is a compromise if and only if $i \in S$. Let $S = \bigcup_{j=1}^{k} [b_j, c_j]$ be a MIR of $S$. We will construct a superset $S'$ of $S$ and a number $L$ as follows. First we place into $S'$ all elements of $S$. Further, for $1 \le j \le k$, if the interval $[b_j, c_j]$ is of odd length, then also $c_j + 1 \in S'$. If $n+1 \notin S'$, set $L = n+1$ and we are done. Otherwise (i.e., if the interval $[b_k, c_k]$ is of odd length and $c_k = n$), we remove $n+1$ from $S'$ and consider two cases. If $S' = \{1, \dots, n\}$, set $L = n+1$; otherwise, set $L = \max\{j \mid j \in \{1, \dots, n\} - S'\}$ and we add $L$ to $S'$.

Thus, if $S' \neq \{1, \dots, n\}$ and $L = n+1$, the set $S'$ is constructed from the set $S$ by extending intervals $I$ of odd length in MIR by an element immediately to the right of $I$. If $L < n$, then intervals $I$ of odd length in MIR lying to the right of $L$ have been extended by the element immediately to the left of $I$.

In what follows, $S$ and $S'$ will always have the same meaning as above.

Let $S' = \{s_1, \dots, s_m\}$, where $s_1 < s_2 < \cdots < s_m$. From the definition of $S'$ and $L$ one can easily see the following.

Observation 1. Let $S' \neq \{1, \dots, n\}$. Then each interval in MIR of $S'$ is of an even length. Thus, $|S'| = m$ is an even number. Further, for $j$ even, $s_{j-1} = s_j - 1$; for $j$ odd, $s_{j+1} = s_j + 1$.

Observation 2. If $L \leq n$, then $[L, n] \subset S'$.

Observation 3. If $s_j \notin S$, then

- $j$ is even if $s_j < L$ or
- $j$ is odd if $s_j \geq L$.

**3. Results and proofs.** Now we are ready to formulate the main result of the paper.

Theorem 3. *The maximum number of range queries that do not lead to a compromise of a database $D$ equals the number of range queries which contain an even number of elements $a_i$, $i \in S'$.*

*Proof.* We shall refer to a range query that contains an even number of elements $a_i$, $i \in S'$, as an even range query; otherwise the query will be called odd. First we claim the following.

Claim 1. *The set $E$ of all even range queries does not lead to a compromise.*

In this part of the proof we employ techniques from linear algebra. Consider the $n$-dimensional vector space over the field of real numbers, where each vector is an ordered $n$-tuple. To each range query $Q$ we assign a vector so that the $i$th coordinate of the vector equals 1 if and only if $a_i \in Q$. By $\mathbf{a}_i$ we denote the vector assigned to the range query $Q = (a_i)$. Further, if $T$ is a set of range queries, then by $L(T)$ we denote the linear hull of the set of vectors corresponding to range queries in $T$. The following statement plays a crucial role in the proof of Claim 1.

Claim 2. *Let $Q$ be a set of range queries. Then $Q$ leads to a compromise if and only if $L(Q)$ contains one of the vectors $\mathbf{a}_i, i \in S$.*

*Proof.* The sufficiency of the condition is obvious. To show the necessity suppose that $Q$ leads to a compromise. Consider a system of linear equations so that to each query $B$ from $Q$ we assign an equation $\sum_{i=1}^{n} \lambda_i a_i = \mathrm{SUM}(B)$, where $\lambda_i = 1$ if $a_i \in B$; otherwise $\lambda_i = 0$. Just for this moment we view $a_i's$ as unknowns of the system. As $Q$ leads to a compromise it has to be possible to infer from $\mathrm{SUM}(B)$, $B \in Q$, at least one of the values $a_i, i \in S$. This means that in the solution of the system of linear equations the corresponding unknown is uniquely determined, although the system can have infinitely many solutions. In turn this implies that the vector $\mathbf{a}_i$ can be expressed as a linear combination of $\mathrm{SUM}(B)$, $B \in Q$, that is, $\mathbf{a}_i$ belongs to $L(Q)$. The proof of Claim 2 is complete.

Hence, in order to prove Claim 1, it suffices to show the following.

Claim 3. *$L(E)$ does not contain any vector $\mathbf{a}_i$, $i \in S$.*

*Proof.* Let, as before, $S' = \{s_1 < \cdots < s_m\}$. Set $\mathcal{W} = \{\mathbf{a}_t \mid t \in \{1, \dots, n\} - S'\} \bigcup \{\mathbf{a}_{s_j} + \mathbf{a}_{s_{j+1}}, 1 \leq j \leq m-1\}$ . Any even range query is easily seen to be a linear combination of vectors from $\mathcal{W}$. Therefore, $L(E) \subset L(\mathcal{W})$. To finish the proof we show that $L(\mathcal{W})$ does not contain any $\mathbf{a}_i, i \in S$. Consider a matrix $A$ of size $(n-1) \times n$

whose rows are vectors of $\mathcal{W}$. Thus, after a suitable permutation of columns,

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 \\ 0 & 1 & 0 & ... & 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 0 & 1 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 \\ 0 & 0 & 0 & ... & 0 & 0 & 1 & 1 & 0 & 0 & ... & 0 & 0 & 0 \\ 0 & 0 & 0 & ... & 0 & 0 & 0 & 1 & 1 & 0 & ... & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 1 & 1 \end{pmatrix};$$

i.e., all elements on the main diagonal equal 1, the elements in the last $m-1$ rows which are immediately to the right of the elements on the main diagonal equal 1 as well, and all other elements of $A$ equal 0.

**A** can easily be seen to be row equivalent to the matrix whose first $(n-1)$ columns form the $(n-1)\times(n-1)$ identity matrix, and the last column **B** is a column vector where the first $n-|S'|$ coordinates equal 0:

$$\mathbf{B}^T = (0, 0, \dots, 0, (-1)^{m-2}, (-1)^{m-3}, \dots, (-1)^1, (-1)^0).$$

This implies that none of the vectors $\mathbf{a}_i$, $i \in S$, is a linear combination of vectors from $\mathcal{W}$; that is, none of the vectors $\mathbf{a}_i$, $i \in S$, is in the span of $\mathcal{W}$. The proof of Claim 3, and thus also the proof of Claim 1, is complete.

To show that $E$ has the largest cardinality among all sets of range queries which do not lead to a compromise of $D$, we use methods of graph theory. Consider a graph $G$ whose set of vertices consists of all range queries from the database $D$ except for range queries containing exactly one element which is single. The edge set of $G$ is defined so that there are no edges connecting two vertices corresponding to range queries of the same parity. Further, each vertex corresponding to an odd range query $q$ is adjacent to exactly one vertex corresponding to an even range query $p$. The query $p$ will be chosen so that the symmetric difference of $p$ and $q$ contains only one element of the database $D$, and this element is single. Let $q = (a_i, \dots, a_j)$; then $p$ is given by the following cases:

1. $i < L$
   (a) if $i \notin S'$,
      $p = (a_i, \dots, a_{j-1})$ if $j < L$, or
      $p = (a_i, \dots, a_{j+1})$ if $j \geq L$;
   (b) if $i \in S'$ and $i = s_k$,
      $p = (a_{i+1}, \dots, a_j)$ if $k$ is odd, or
      $p = (a_{i-1}, \dots, a_j)$ if $k$ is even;
2. $i \geq L$ and $j = s_k$,
   $p = (a_i, \dots, a_{j-1})$ if $k$ is even, or
   $p = (a_i, \dots, a_{j+1})$ if $k$ is odd.

In what follows we will make use of Claim 4 below.

CLAIM 4. *The unique element $a_t$ in the symmetric difference of $p$ and $q$ is single; i.e., $t \in S$.*

*Proof.* If $i < L$ and $i \notin S'$, then $j \in S'$. Further, $j$ is odd, and by Observation 3, $j \in S$ for $j < L$, and $j+1 \in S$ for $j \geq L$. The other cases follow directly from Observation 3.

CLAIM 5. *Each vertex corresponding to an even range query is of degree at most 1 in $G$.*

*Proof.* Let $p' = (a_i, \dots, a_j)$ be an even range query. Suppose first that $i \notin S'$. By the definition of the edge set of $G$, if an odd query $q = (a_r, \dots, a_s)$ is adjacent to $p'$, then $r = i$ (note that in cases 1(b) and (2) the leftmost element of $p$ is from $S'$) and $s = j-1$ or $j+1$ with respect to $L$. Thus $p'$ is of degree at most 1. If $i \in S'$ and $i < L$, then an edge incident to $p'$, if any, has to be given by case 1(b) of the definition of $G$, i.e., there is again at most one such edge. Similarly, for $i > L$, only case 2 may apply. Hence, $p'$ is of degree at most 1 in $G$. The proof is complete.

In view of Claim 4, $|\mathrm{SUM}(p) - \mathrm{SUM}(q)| = a_t$, where $t \in S$, thus revealing that both $p$ and $q$ give a compromise. Therefore, if a set of range queries does not lead to a compromise, then the corresponding set of vertices is an independent set of vertices of $G$.

Let $E$ be a set of vertices of $G$ corresponding to even range queries. Obviously, $E$ is an independent set of $G$. Claim 5 implies that $E$ is a largest independent set of $G$, that is, a largest set of range queries which does not lead to a compromise. ∎

*Remark.* Theorem 3 describes a largest family of range queries which does not lead to a compromise. In general, for databases containing multielements, this family is not determined in a unique way. To see this, it suffices to define the set $S'$ in a more general way by the following recursive procedure:

1. Set $S' = S$.

2. If there is an interval $[a, b]$ of an odd length in a MIR of $S'$, add to $S'$ one of the numbers $a-1, b+1$ which is from $\{1, \dots, n\}$. Otherwise stop.

3. If $S' \neq \{1, \dots, n\}$, go to 2. Otherwise stop.

Thus, if there is at least one interval of odd length in MIR of $S$, which gives a choice in step 2, we would obtain at least two distinct sets $S'$. It is not difficult to see that the proof of Theorem 3 could be adjusted to show that the statement is valid for all sets $S'$ obtained by the above recursive procedure.

To calculate the usability of a database with posable queries being the range queries, we first determine the cardinality of the family of range queries given in Theorem 3.

Let $n, m$ be natural numbers. We set, for $n = m$ being an odd number, $K(n,m) = \frac{n^2-1}{4}$; otherwise $K(n,m) = \frac{m}{4}(2n-m) + \binom{n-m+1}{2}$.

Theorem 4. *Let $D$ be a database with $n$ elements and $|S'| = m$. Then the number of even range queries of $D$ equals $K(n,m)$.*

*Proof.* We denote by $D(n,m) = (a_1, \dots, a_n)$ a database with $n$ elements and $|S'| = m$ such that $S' = \{n-m+1, \dots, n\}$.

Claim 1. *The total number of even range queries of $D(n,m)$ equals $K(n,m)$.*

*Proof of Claim* 1. If $n = m$ is an odd number, then $S' = \{1, \dots, n\}$. Thus, there are $n-1$ range queries with two elements, $n-3$ range queries with four elements, ... , four range queries with $n-3$ elements, and two range queries with $n-1$ elements. In total, there are $K(n,m) = \frac{n^2-1}{4}$ even range queries. Otherwise, $m = |S'|$ is an even number. Let $Q = (a_i, \dots, a_j)$ be an even range query. Then, either

(a) $j \leq n-m$. In this case $Q$ contains no elements with indices from $S'$. Then $i$ can be any number from the interval $[1, j]$. Thus, in this case, there are $\binom{n-m+1}{2}$ even range queries; or

(b) $i \leq n-m < j$. As $Q$ is an even range query, then to an arbitrary choice of $i \leq n-m$ we have to take $j$ so that $j-(n-m)$ is an even number. Therefore, there are $(n-m) \times \frac{m}{2}$ even range queries of this type; or

(c) $n-m < i$. Then there are $m-1$ range queries with two elements, $m-3$ range queries with four elements, ... , three range queries with $m-2$ elements, and one range query with $m$ elements. Together, in this case, we have $\frac{m^2}{4}$ even range queries.

Summing up the numbers obtained in (a)–(c) gives the desired result.

By moving a multielement of a database $D \neq D(n,m)$ we mean an operation which takes the leftmost (= with the minimum index) element $x = a_k$ of $D$, $k \notin S'$, with the property that $k-1 \in S'$, and places it immediately before the first element $a_f$ with its index in $S'$(i.e., $f = \min(S')$). The obtained database will be denoted by $D_x$. For example, let $D = (t,t,t,s,s,s,s,t,s,s,t,t,s,s)$ be a database where $t$ are elements with indices off $S'$ and $s$ are elements with indices in $S'$. Then the element which is going to be moved is the fourth $t$ from the left and $D_x = (t,t,t,t,s,s,s,s,s,s,t,t,s,s)$.

CLAIM 2. *The cardinalities of the sets of even range queries of $D$ and of $D_x$ are the same.*

*Proof of Claim* 2. Let $D = (a_1, \dots, a_f, \dots, a_{k-1}, a_k, a_{k+1}, \dots, a_n)$, where $f = \min(S'), [f, k-1] \subset S'$, and $k \notin S'$. Then the operation of moving a multielement will move the element $a_k$; thus $D_x = (a_1, \dots, a_{f-1}, a_k, a_f, \dots, a_{k-1}, a_{k+1}, \dots, a_n)$. Consider a mapping $F$ which assigns to an even range query $Q = (a_i, a_j)$ of $D$ an even range query $Q_x$ of $D_x$ such that

(a) $Q_x = (a_i, a_j)$ for $k \neq j$ or $k = j$ and $i < f$,

(b) $Q_x = (a_k, a_{i+1})$ for $k = j, f \leq i < k$.

Note that the range queries $Q = (a_i, a_j)$ of $D$ and $Q_x = (a_i, a_j)$ of $D_x$ do not have the same number of elements although "visually" they appear to be the same. It is a matter of routine to check that the mapping $F$ defined above is a bijection on the sets of all even range queries of $D$ and $D_x$. This completes the proof of Claim 2.

Now we are ready to prove Theorem 4. Consider a database $D$ with $n$ elements and $|S'| = m$. By repeatedly applying the operation of moving a multielement of $D$ we obtain the database $D(n,m)$. By Claim 2, the number of even range queries in $D$ equals the number of even range queries in $D(n,m)$ and, in turn, by Claim 1, this equals $K(n,m)$.

Combining Theorems 3 and 4 we get the following corollary.

COROLLARY. *Let $D$ be a database with $n$ elements, $|S| = s$, and let there be $t$ intervals of odd length in MIR of $S$. Then the maximum number of range queries that does not lead to a compromise of $D$ is equal to $K(n,m)$, where $m = s + t$.*

*Remark.* Given a database $D$, let $M_D$ be the cardinality of a largest set of range queries which does not lead to a compromise of $D$. As the values of multielements could be revealed to the user, at first glance it appears that the bigger the number of multielements in $D$, the bigger the number $M_D$. However, the above statement shows that $M_D$ depends on the set of single elements in an unexpected way. For example, consider a database $D$ with the same number of multielements and single elements so that the multielements and single elements of $d$ alternate, and consider a database $D'$ having only two multielements which are at the beginning of $D'$. Then $M_{D'} > M_D$. On the other hand, whatever is the allocation of multielements $a_i, i \notin S'$, in gaps between elements with indices in $S'$, the number $M_D$ is the same for all databases.

The following theorem is an immediate consequence of the corollary.

THEOREM 5. *The usability of a 1-dimensional database with $n$ elements and posable queries being range queries is at least $\frac{n^2-1}{2(n^2+n)}$.*

*Proof.* A simple calculation shows that, for $n$ fixed, the function $K(n,m)$ is decreasing on the interval $[1,n]$. As the maximum possible value of $m$ equals $n$, we get $K(n,m) \geq K(n,n) \geq \frac{n^2-1}{4}$. Further, the total number $R$ of range queries equals the number of combinations of two elements from $n$ elements with repetitions, as each range query is determined by its first and last elements (which may coincide). Thus,

$R = \binom{n+1}{2} = \frac{n^2+n}{2}$. The proof follows.

At the end of this section we provide an outline of the proof of a statement given in the introduction.

THEOREM 6. *If $D = \{a_1, , \ldots , a_n\}$ is a (nonordered) database with $m$ single elements, then the cardinality of a largest set of answerable queries of $D$ equals $t = \binom{m}{\lfloor \frac{m}{2} \rfloor} \times 2^{n-m}$.*

*Proof.* Let $D' = \{a_1, \ldots , a_m\}$ be single elements of $D$, $D'' = D - D'$, and let $M$ be a set of answerable queries of $D'$ of cardinality $\binom{m}{\lfloor \frac{m}{2} \rfloor}$. Then the set $Q$ of queries $F$ of $D$ of the form $A \cup B$, where $A \in M$, and $B \subset D''$, has cardinality $t$. We show that $Q$ does not lead to a compromise of $D$. As in the proof of Theorem 3, consider a system of linear equations so that to each query $F$ from $Q$ we assign an equation $\sum_{i=1}^{n} \lambda_i a_i = SUM(F)$, where $\lambda_i = 1$ if $a_i \in F$; otherwise $\lambda_i = 0$. Suppose to the contrary that $Q$ leads to a compromise of $D$. Then there is a solution of the system such that at least one of the unknowns $x_j, j = 1, \ldots , m,$ is uniquely determined, although the system may have infinitely many solutions. Consider now a system where, to each query $F$, we assign an equation $\sum_{j=1}^{m} \lambda_{ij} a_j = SUM(F) - \sum_{j=m+1}^{n} \lambda_{ij} a_j = SUM(A)$, where $A = (F \cap D') \in M$. The new system also has the property, which contradicts the assumption that $M$ is a set of answerable queries of $D'$. Thus $|F| \geq t$.

On the other hand, let $Q$ be a set of answerable queries of $D$ and $B \subset D''$. Set $Q_B = \{S|\ S \in F, S = A \cup B,$ where $A \subset D'\}$. Then $|Q_B| \leq \binom{m}{\lfloor \frac{m}{2} \rfloor}$, i.e., $|Q| \leq t$; otherwise the family $\{S - B, S \in Q_B\}$ would provide a family of answerable queries of $D'$ with more than $\binom{m}{\lfloor \frac{m}{2} \rfloor}$ elements, which is a contradiction. The proof is complete. ▯

## 4. Conclusions.

Let $M_D$ be the cardinality of the largest set of posable queries of a database $D$ which does not lead to its compromise.

As mentioned before, $M_D$ has been determined in the case when all queries are posable. However, the usability of the database in this case tends to 0 when the number of elements of $D$ grows over all bounds. Thus, this model is of theoretical value only and cannot be successfully used in practice.

In [4] $M_D$ is determined in the case of a $k$-dimensional database (= $k$-dimensional matrix) $D$ whose elements are all single and the set of posable queries is the set of all range queries (= contiguous submatrices). It is proved there that the largest family of range queries which does not lead to a compromise of $D$ is uniquely determined and equals the family of all queries which has an even number of elements. This means, for large databases, that in this case the usability tends to the number $\frac{2^k-1}{2^k}$ which is definitely sufficient for practical use. However, this model is also more or less of theoretical value only, as it is unrealistic to expect that in a larger database no element is a hole. If all elements of $D$ were single or multielements, the above result would give a family of range queries which would be big enough, although not a largest one, for practical use.

Theorem 3 is a generalization of the above result from [4] for 1-dimensional databases as posable queries are also range queries but in addition we admit multielements. Some methods used for proving Theorem 3 are similar to those developed in [4]. By Theorem 5 we get that the usability of this database tends to $\frac{1}{2}$ with a growing number of the elements of $D$. With respect to this, Theorem 3 constitutes the first result in the area with practical applications, as one can easily exclude holes in this case. On the other hand, the majority of databases in real life is of dimension higher than 1. Therefore, a generalization of Theorem 3 for a higher dimension, where multielements and holes would be allowed, would be very much appreciated.

## REFERENCES

[1] L. Branković, P. Horak, M. Miller, and G. Wrightson, *Usability of compromise-free statistical databases*, in Proceedings of the 9th International Conference on Scientific and Statistical Database Management, IEEE Press, Piscataway, NJ, 1997, pp. 144–154.

[2] F. Y. Chin, P. Kossowski, and S. C. Loh, *Efficient inference control for range sum queries*, Theoret. Comput. Sci., 32 (1984), pp. 77–86.

[3] J. R. Griggs, *Concentrating subset sums at $k$ points*, Bull. Inst. Combin. Appl., 20 (1997), pp. 65–74.

[4] P. Horak, L. Branković, and M. Miller, *On a combinatorial problem in data security*, Discrete Appl. Math., 91 (1999), pp. 119–126.

[5] M. Miller, I. Roberts, and J. Simpson, *Application of symmetric chains to an optimization problem in the security of statistical databases*, Bull. Inst. Combin. Appl., 2 (1991), pp. 47–58.

SIAM J. Discrete Math.
Vol. 13, No. 3, pp. 354–369

# THE GRAPH OF LINEAR EXTENSIONS REVISITED*

MICHAEL NAATZ†

**Abstract.** The graph of linear extensions $G(P)$ of a poset $P$ has as vertices the linear extensions of $P$, and two vertices are adjacent if they differ only by an adjacent transposition. This graph has so far been investigated mainly with respect to Hamilton paths and intrinsic geodesic convexity. The work on the latter topic especially shows how the graph-theoretic properties of $G(P)$ reflect the order-theoretic structure of $P$.

The aim of this paper is to study the graph $G(P)$ with respect to topics from classical graph theory, e.g., connectivity, cycle space, isometric embeddings, and to find order-theoretic interpretations of these notions. The main theorems in this paper are the equality of the connectivity of $G(P)$ and the jump number of $P$, the existence of a certain generating system for the cycle space, and a relationship of $P$ and its subposets obtained via an embedding of the graph of linear extensions.

**Key words.** poset, linear extension, transposition, jump number, Hamilton path, cycle space, connectivity, isometric subgraph, embedding

**AMS subject classifications.** 06A06, 05C12, 05C38, 05C40, 05C45

**PII.** S0895480199352609

## 1. Introduction.

The graph of linear extensions $G(P)$ of a poset $P$ (also called the adjacent transposition graph) has as vertices the linear extensions of $P$, and two of them are adjacent if they differ only by an adjacent transposition. This graph was first defined by Pruesse and Ruskey [10] (see also [15] for related work). They mainly investigated the question, In which cases can one find a linear ordering of all linear extensions of a poset such that two consecutive linear extensions differ only by a transposition? They also considered the case where only adjacent transpositions are allowed, which is equivalent to finding a Hamilton path in $G(P)$. Hamilton paths in $G(P)$ were also investigated by Stachowiak [17] and West [19].

Björner and Wachs are the first to consider structural properties of $G(P)$ apart from Hamiltonians. In their paper [1] they prove the fundamental theorem that the lattices of all extensions of a poset $P$ is dually isomorphic to the lattice of all convex subsets (vertex sets of convex subgraphs; see also Remark 4.3) of the graph of linear extensions. This theorem was rediscovered by Reuter [13], who further investigated the intrinsic geodesic convexity in the graph of linear extensions.

Especially the papers by Björner and Wachs and by Reuter show clearly that the graph of linear extensions contains a wealth of information about the underlying poset. It is therefore natural to study the graph-theoretic properties of this graph and their order-theoretic interpretations—a line of research that is also pursued in [4]. In the present paper we focus on parameters and constructions from "classical" graph theory such as cycle space, connectivity, and others.

## 2. Preliminaries.

A poset is a pair $(P, \leq_P)$, where $P$ is a set and $\leq_P$ is a reflexive, transitive, and antisymmetric relation on $P$. By standard abuse of notation

*Received by the editors February 19, 1999; accepted for publication (in revised form) March 1, 2000; published electronically May 26, 2000. The material in this article is part of the author's diploma thesis written at the University of Hamburg, Germany. The author was partially supported by the graduate school Algorithmische Diskrete Mathematik, which is supported by the Deutsche Forschungsgemeinschaft, grant GRK 219/3.
http://www.siam.org/journals/sidma/13-3/35260.html

†Fachbereich Mathematik MA 6-1, Technische Universität Berlin, Straße des 17. Juni 136, D-10623 Berlin, Germany (naatz@math.tu-berlin.de).

we just write $P$ instead of $(P, \leq_P)$. Furthermore, $<_P$ is defined to be the irreflexive relation corresponding to $\leq_P$, i.e., $<_P := \leq_P \setminus \{(p,p) : p \in P\}$. A graph $G$ consists of a vertex set denoted by $V(G)$ and an edge set $E(G) \subseteq \{\{v,w\} : v,w \in V(G), v \neq w\}$. In accordance with this definition, all graphs are assumed to be undirected and without loops or multiple edges. When we are dealing with graphs of linear extensions rather than arbitrary graphs we denote the vertices by capital letters, beginning with $L, M, \dots$, following the notational conventions for linear extensions used in [18]. All graphs and posets are assumed to be finite. For graph-theoretic terminology we refer to [3] and for the basics of order theory to [8] and [18]. In particular, we will denote the vertex-connectivity (respectively, edge-connectivity) of a graph $G$ by $\kappa(G)$ (respectively, $\lambda(G)$) and the minimal degree by $\delta(G)$. The order dimension of a poset $P$ will be denoted by $\dim(P)$.

We will make use of the following shorthand notations: For the distance of two vertices $v$ and $w$ in a graph $G$ we will write $d(v,w)$ instead of $d_G(v,w)$, when it is clear which graph is meant—normally it will be the graph of linear extensions of a given poset. The same applies to the degree $\deg(v)$ of a vertex $v$. A path $v_1 v_2 \dots v_k$ will be called a $v_1$–$v_k$-path, and it will be called a *shortest path* if $d(v_1, v_k) = k-1$. The vertices $v_2, v_3, \dots, v_{k-1}$ are called the *inner vertices* of the path. An *$n$-cycle* is a cycle with $n$ edges, which is the same as a cycle of length $n$. We will call $v_1$ and $v_k$ the end vertices of the path $v_1 v_2 \dots v_k$. Paths $P_1, \dots, P_n$ with the same end vertices $v$ and $w$ will be called *independent paths*, if every two distinct paths $P_i$ and $P_j$ have only the vertices $v$ and $w$ in common. If $S$ is some statement, we define $\mathbf{1}_{[S]}$ to be one if $S$ is true, and zero otherwise. The symmetric difference of two sets $A$ and $B$ will be denoted by $A \bigtriangleup B$.

A word of the form $\ell_i \ell_{i+1} \dots \ell_k$ for some $1 \leq i \leq k \leq n$ will be called a *segment* of $\ell_1 \dots \ell_n$. If $pq$ is a segment of a given linear extension $L$, we say that $p$ is the *left neighbor* of $q$ (in $L$), $q$ is the *right neighbor* of $p$, and $\{p,q\}$ is a *neighboring pair* of $L$. We will mostly write linear extensions as words, but if we explicitly want to consider a linear extension $L$ as an order relation, we will denote this by

$$<_L := \mathcal{R}(L) := \operatorname{tr}\big(\{(p,q) \in P \times P : p \text{ is the left neighbor of } q \text{ in } L\}\big),$$

where tr denotes the transitive hull. It is convenient to use both symbols for readability reasons. When $L$ is a linear extension of $P$ and we consider a subposet $P'$ of $P$, we define the word $L\,|_{P'}$ consistently such that $\mathcal{R}(L\,|_{P'}) = \mathcal{R}(L) \cap (P' \times P')$. Note that here, as in the following, subposets are not necessarily induced. Comparable and incomparable pairs are always *unordered* pairs. The set of all comparable and incomparable pairs of a poset $P$ is denoted by $\operatorname{Com}(P)$ and $\operatorname{Inc}(P)$, respectively; hence we have

$$\operatorname{Com}(P) = \{\{p,q\} : p <_P q \text{ or } q <_P p\}.$$

The number $|\operatorname{Inc}(P)|$ of incomparable pairs of $P$ will be denoted by $\operatorname{inc}(P)$. If the elements of an incomparable pair $J$ are neighbors in a linear extension $L$, we call $J$ a *jump* of $L$. The jump number $\operatorname{s}(P)$ of a poset $P$ is defined to be the least number of jumps of all linear extensions of $P$. The following proposition is an easy consequence of these definitions.

PROPOSITION 2.1. *The number of jumps of a linear extension equals its degree in $G(P)$, for every poset $P$. In particular, the jump number of $P$ equals the minimal degree $\delta(G(P))$ of $G(P)$.*

When we compare two linear extensions $L$ and $L'$ that are adjacent in $G(P)$, it is clear that there is one and only one incomparable pair $\{p_1, p_2\}$ of $P$ with $p_1 <_L p_2$ and $p_2 <_{L'} p_1$; all other pairs of elements of $P$ will appear in the same order in $L$ and $L'$. So for every edge $\{L, L'\}$ we can use the corresponding incomparable pair $\{p_1, p_2\}$ as an edge label and denote it by $\psi(\{L, L'\})$. We will refer to this labeling frequently throughout the paper. When $J$ is a jump of a linear extension $L$ we will write $L \curvearrowleft J$ to denote the linear extension $L'$ constructed from $L$ by exchanging the elements of $J$. Hence $L$ and $L \curvearrowleft J$ are always adjacent in $G(P)$, and $J$ is the label of the edge joining them. We say that the pair $\{p, q\}$ is reversed in a linear extension $L'$ (with respect to a given linear extension $L$) when $p <_{L'} q$ and $q <_L p$ holds (or vice versa).

The following proposition was already stated in [13] and appeared even earlier in [1, proof of Proposition 6.2], though in a slightly different setting. We give a short proof to keep the paper self-contained.

Proposition 2.2 (see [1] and [13]). *For every two linear extensions $L$ and $M$ of a poset $P$ the following holds:*

$$d_{G(P)}(L, M) = \frac{1}{2}|\mathcal{R}(L) \triangle \mathcal{R}(M)|.$$

*Proof.* The elements of $D := \mathcal{R}(L) \triangle \mathcal{R}(M)$ are exactly the pairs $(p, q) \in P \times P$, where $p$ and $q$ are ordered differently in $L$ and $M$. Observe that for each element $(p, q)$ of $D$, the reverse pair $(q, p)$ is also in $D$.

Now consider a shortest $L$–$M$-path $W$ in $G(P)$. For every pair $(p, q) \in D$ there has to be an edge of $W$ that is labeled $\{p, q\}$, since otherwise the pair $\{p, q\}$ would appear in the same order in $M$ as in $L$. Hence $d_{G(P)}(L, M) \geq \frac{1}{2}|D|$.

We show the opposite inequality by induction on $d := \frac{1}{2}|D|$. For $d = 0$ we have $L = M$ and, consequently, $d(L, M) = 0$. Let $L$ and $M$ be vertices with $\frac{1}{2}|\mathcal{R}(L) \triangle \mathcal{R}(M)| = d+1$. Then there is a vertex $M'$ adjacent to $M$ such that $\frac{1}{2}|\mathcal{R}(L)\triangle\mathcal{R}(M')| = d$, and $d(L, M) \leq d + 1$ follows by the induction hypothesis. ∎

We state a second version of the above proposition for later reference.

Proposition 2.3. *Let $P$ be a poset.*

(i) *A path $W$ in $G(P)$ is a shortest path if and only if $\psi(e) \neq \psi(f)$ for every two distinct edges $e$ and $f$ of $W$.*

(ii) *If $W$ and $W'$ are both shortest paths from $L$ to $L'$ in $G(P)$, then*

$$\{\psi(e) : e \textit{ is an edge of } W\} = \{\psi(e) : e \textit{ is an edge of } W'\}.$$

*Proof.* (i) The first part follows from Proposition 2.2.

(ii) The second part is also easily seen to be implied by part (i) and Proposition 2.2: Just observe that, for every shortest $L$–$M$-path $W$ in $G(P)$,

$$\{\psi(e) : e \text{ is an edge of } W\} = \big\{\{p, q\} : (p, q) \in \mathcal{R}(L) \triangle \mathcal{R}(M)\big\}. \qquad ∎$$

**3. Connectivity.** The aim of this section is to prove the following theorem.

Theorem 3.1. *For every poset $P$, the edge- and vertex-connectivity of the graph of linear extensions of $P$ agrees with the jump number of $P$:*

$$\kappa(G(P)) = \lambda(G(P)) = \mathrm{s}(P).$$

As the present author learned after completing the work on this result, Savage and Zhang [16] conjectured that the vertex-connectivity of $G(P)$ is at least one less than the width $w(P)$ of $P$. This conjecture is proved by Theorem 3.1, as $w(P) - 1$ is the most well-known lower bound for the jump number of $P$.

The overall proof strategy for Theorem 3.1 is simple: We will construct at least $\delta(G(P))$ independent paths between every pair of vertices at distance two. The equality of vertex-connectivity and minimal degree (and hence the jump number of $P$) in $G(P)$ then follows directly from a surprising variant of Menger's theorem that was found by Li [7]. We will discuss that variant in the next subsection. It is then clear that the edge-connectivity has the same value because of the elementary inequality $\kappa(G) \leq \lambda(G) \leq \delta(G)$, which is true for every graph $G$.

**3.1. An application.** Before we prove Theorem 3.1, we want to illustrate its possible use with an example taken from Pruesse and Ruskey [11, p. 383]; this subsection is mainly a translation of ideas from this paper. Consider the following problem:

(∗) Generate a linear ordering of the set of all linear extensions of a given poset $P$, such that two consecutive extensions differ by *at most two* adjacent transpositions.

This is a relaxation of the problem mentioned in the introduction that allows for only one adjacent transposition.

Corollary 3.2 (see [11], p. 383). *The problem (∗) is solvable for every poset with jump number at least two (and trivially for jump number zero).*

*Proof.* The assertion is clear for jump number zero because this means that the poset is a chain. The nontrivial case follows from Theorem 3.1 and Fleischner's theorem that the square of any two-connected graph has a Hamilton cycle (see [5] and [6] and also [3] or [14] for a shorter proof). ☐

Note that Theorem 3.1 is a rather heavy tool for the proof of this corollary. Pruesse and Ruskey suggest instead to prove a lemma to the effect that $G(P)$ minus any vertices of degree one is always two-connected, which is much easier to show. This suffices for the proof of Corollary 3.2, using the elementary observation that vertices of degree one can only occur in $G(P)$ if $P$ has jump number one. Also in [11], an algorithm is constructed to generate the desired ordering in constant amortized time, i.e., the time needed is a constant multiple of the number of linear extensions of the given poset. This algorithm is proved to work for arbitrary ordered sets, hence Problem (∗) is also solvable in the case of jump number one.

**3.2. A variant of Menger's theorem.** The following variant of Menger's theorem seems to be not well known. It was stated by Li in [7]. We give a new proof here.

Theorem 3.3 (see [7]). *The following equivalence holds for every connected graph $G$ on at least $k+1$ vertices: $G$ is $k$-connected if and only if, for every two vertices $v$ and $w$ at distance 2, there are at least $k$ independent $v$–$w$-paths in $G$.*

*Proof.* The "only if" part is just a special case of Menger's theorem, and we prove the other direction by induction on $k \geq 0$.

The theorem is obviously true for $k \leq 1$. Let $G$ be a connected graph on at least $k+2$ vertices, such that there exist at least $k+1$ independent paths between every two vertices at distance two. Let $x$ and $y$ be two distinct vertices of $G$, and let $t_1, \ldots, t_k \in V(G) \backslash \{x, y\}$. We show now that $x$ and $y$ are not separated by $\{t_1, \ldots, t_k\}$. By induction, $G$ is k-connected, and from Menger's theorem we get $k$ independent $x$–$y$-paths in $G$. If one of these paths contains none of the vertices $t_1, \ldots, t_k$, we are already done. Otherwise let $P$ be one of those paths and $t$ the only element of

$\{t_1, \ldots, t_k\}$ that lies on $P$. There are just two vertices adjacent to $t$ in $P$; we call them $v_1$ and $v_2$. These two vertices are either adjacent in $G$ (then $P - \{t\}$ together with the edge $\{v_1, v_2\}$ is an $x$–$y$-path in $G - \{t_1, \ldots, t_k\}$), or $d(v_1, v_2) = 2$. In the latter case there are $k+1$ independent $v_1$–$v_2$-paths in $G$ (by the hypothesis), and obviously there cannot be a $t_i$ on every path. So there is a walk from $x$ to $y$ in $G - \{t_1, \ldots, t_k\}$, and we conclude that $G$ is $(k+1)$-connected. ∎

*Remark* 3.4. In the same way one can prove that a connected graph on at least two vertices is $k$-edge-connected if there are $k$ edge-disjoint paths between every two *adjacent* vertices.

Note that it follows from Proposition 2.2 that $G(P)$ is connected for arbitrary $P$, so Theorem 3.3 will be readily applicable to our problem.

**3.3. Order-theoretic preparations.** In this subsection we will state and prove some propositions that are needed to construct the paths necessary for the proof of Theorem 3.1. *For the rest of this section all posets are without loss of generality assumed to have a least and a greatest element denoted by $\hat{0}$ and $\hat{1}$, respectively.*

Now let $p \neq \hat{0}$ be an element of a poset $P$, and let $L$ be a linear extension of $P$. Provided $p$ is incomparable with its left neighbor in $L$, we can construct from $L$ a new linear extension by successively exchanging $p$ with its left neighbor, until we end up with a linear extension $L'$ in which $p$ is comparable with its left neighbor. We then say that $L'$ is *generated from $L$ by shifting down $p$*. We will frequently use this construction in this section.

Let us illustrate this notion with a small example: Let $P$ be the poset from Figure 3.1. Then the linear extension $\hat{0}2314\hat{1}$ is generated from $\hat{0}2143\hat{1}$ by shifting down 3 because 3 is incomparable with 4 and 1 but comparable with 2.

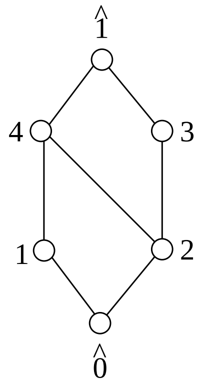


FIG. 3.1. *N-shaped poset with additional least and greatest element.*

PROPOSITION 3.5. *Let $L$ and $L'$ be linear extensions of a poset $P$, such that $L'$ is generated from $L$ by shifting down $p \in P \setminus \{\hat{0}\}$, and let $p$ be incomparable with its left neighbor in $L$. Then the degree of $L'$ in $G(P)$ is given by*

$$\deg(L') = \deg(L) - \mathbf{1}_{[l' \| r']} - \mathbf{1}_{[p \| r]} + \mathbf{1}_{[l \| r]},$$

*where $l, r \in P$ and $lpr$ and $l'pr'$ are segments of $L$ and of $L'$, respectively.*

*Proof.* Moving from $L$ to $L'$ just means replacing the segments $lpr$ by $lr$ and $l'r'$ by $l'pr'$, respectively. As the degree of a linear extension in the graph $G(P)$ is just the number of jumps in that extension (Proposition 2.1), the result follows by simply counting the number of jumps before and after those replacements. ∎

LEMMA 3.6. *Let $L = \ell_1\ell_2\ldots\ell_n$ be a linear extension of a poset $P$, and let $I \subseteq \{1, 2, \ldots, n-2\}$ be a set of indices with the following properties.*

(i) *$\ell_i \parallel_P \ell_{i+1}$, $\ell_{i+1} \parallel_P \ell_{i+2}$ and $\ell_i <_P \ell_{i+2}$ for all $i \in I$.*

(ii) *$i \in I$ implies that $i+2$ is not in $I$.*

*Then* $\deg(L) - |I|$ *is an upper bound for* $\delta(G(P))$.

*Proof.* The lemma is trivial for $I = \emptyset$. For $I \neq \emptyset$, partition $I$ into

$$I^+ := \{i \in I : i+1 \in I\} \text{ and } I^- := \{i \in I : i+1 \notin I\}.$$

The following easy algorithm constructs a linear extension $L''$ of $P$ with $\deg(L'') \leq \deg(L) - |I^+| - |I^-| = \deg(L) - |I|$.

*Step* 1. Let $i := \min I$ and $L' := L$.

*Step* 2, *Case* 1. $i$ is in $I^+$. In this case, construct from $L' = \ell'_1\ell'_2\ldots\ell'_n$ a new linear extension $L''$ by exchanging $\ell'_{i+1}$ and $\ell'_{i+2}$, which are incomparable by condition (i). If $i+1 = \max I$, the construction is finished. Otherwise, assign the following new values: $i := \min(I \setminus \{1, 2, \ldots, i+1\})$ and $L' := L''$. Afterward go back to Step 2.

*Step* 2, *Case* 2. $i$ is in $I^-$. In this case, construct from $L' = \ell'_1\ell'_2\ldots\ell'_n$ a new linear extension by shifting down $\ell'_{i+1}$. If $i = \max I$, the construction is finished. Otherwise, assign the following new values: $i := \min(I \setminus \{1, 2, \ldots, i\})$ and $L' := L''$. Afterward go back to Step 2.

This algorithm obviously terminates after finitely many steps because the value of $i \in I$ increases every time Step 2 is executed. To see that the algorithm is also correct, the crucial observation is that $\ell'_k = \ell_k$ holds for all $k \geq i$ at each moment during the execution; this follows from Property (ii) of $I$. For Case 1 of Step 2 this implies $\deg(L'') = \deg(L') - 2$ because of the hypotheses, and for Case 1 we get $\deg(L'') \leq \deg(L') - 1$ by Proposition 3.5. □

**3.4. Proof of the main theorem.** The edge labeling defined in the preliminaries can be used as a means of specifying paths in $G(P)$: When we know an end vertex $v$ of a path $W$ and the edge labels in the order in which they appear when we move from $v$ to the other end vertex of $W$, then $W$ is uniquely determined. We call $v$ the *start vertex* in this context. But not every pair consisting of a vertex $K$ and a sequence $E_1, E_2, \ldots, E_m$ of edge labels corresponds to a path in $G(P)$. For a corresponding path to exist, $E_1$ has to be a jump of $K$, $E_2$ must be a jump of $K \curvearrowleft E_1$, $E_3$ must a jump of $(K \curvearrowleft E_1) \curvearrowleft E_2$, and so on. It is clear that this is a necessary and sufficient condition. Often it is easy—though perhaps tedious—to check this condition for a given vertex and sequence, in which case we will leave this task to the reader without explicitly mentioning it.

*Proof of Theorem* 3.1. The graph of linear extensions is always connected by Proposition 2.2. We will now construct at least $\delta(G(P))$ independent paths between every two vertices at distance two. The equality of vertex connectivity and minimal degree (and hence the jump number of $P$) in $G(P)$ then follows directly from Theorem 3.3. The edge connectivity has the same value because of the elementary inequality $\kappa(G) \leq \lambda(G) \leq \delta(G)$ for every graph $G$.

Let $L = \ell_1\ldots\ell_n$ and $M$ be vertices of $G(P)$ with $d(L, M) = 2$ and $\deg(L) \leq \deg(M)$. This means that there are exactly two incomparable pairs $J_1$ and $J_2$ of $P$ that appear in $L$ and $M$ in different order. We assume without loss of generality that $J_1$ is a jump of $L$, hence $M = (L \curvearrowleft J_1) \curvearrowleft J_2$. For every edge $e$ incident with $L$ we will give a sequence of edge labels that has $\psi(e)$ as first element. Afterwards we prove that at least $\delta(G)$ of these sequences define an $L$–$M$-path (where we use $L$ as start vertex), and that all these paths are independent. As for the independence, we will provide a criterion for each path $W$ that distinguishes the inner vertices of $W$ from the inner vertices of all previously defined paths. When we speak about reversed pairs of elements in some linear extension we always mean reversed with respect to $L$. All paths to be defined by means of label sequences in the following are supposed to have $L$ as start vertex.

*Case* 1. For the edge $e$ with $\psi(e) = J_1$, we choose the path with the edge labels $J_1$ and $J_2$, and if there is an edge $e'$ incident with $L$ such that $\psi(e') = J_2$, we go along the edges labeled $J_2$ and $J_1$. It is easily checked that these sequences correspond to paths in $G(P)$. The only inner vertex of the path belonging to $e$ (respectively, $e'$) is $L \curvearrowleft J_1$ (respectively, $L \curvearrowleft J_2$).

*Case* 2. For all edges $e$ incident with $L$ that have the additional property that $\psi(e) \cap J_1 = \emptyset = \psi(e) \cap J_2$, we choose the edge labels $\psi(e)$, $J_1$, $J_2$, and again $\psi(e)$. It is easy to check that these sequences correspond to paths in $G(P)$. In any inner vertex $K$ of such a path the pair $\psi(e)$—which is disjoint from $J_1$ and $J_2$—is reversed. From this it is clear that the paths defined in Case 2 are mutually independent and independent of the paths from Case 1.

*Case* 3. Now we consider the edges $e$ for which $\psi(e)$ has an element in common with $J_1$ or $J_2$; this part splits up into various subcases and subsubcases.

*Case* 3.1. $J_1$ and $J_2$ are disjoint. In this case let $\{o, p\} := J_1$ and $\{q, r\} := J_2$, and assume $o <_L p <_L q <_L r$ without loss of generality.

*Case* 3.1.1. There is an element of $P$ that is strictly between $p$ and $q$ with respect to $L$. Let $o'opp'$ and $q'qrr'$ be segments of $L$, where we do not exclude $p' = q'$. In Table 3.1 we define a sequence of edge labels for every jump (other than $J_1$ and $J_2$) that can possibly occur in one of these segments. For the moment, just ignore the third column and the fact that some labels are in boldface; both will be needed later.

Table 3.1
*Sequences of edge labels used in the proof of Theorem* 3.1.

| Jump $J$ | Corresponding label sequence $s_J$ | Characteristic property of inner vertices |
|---|---|---|
| $\{o', o\}$ | $\{o', o\}, \mathbf{\{o', p\}}, J_1, J_2, \{o', o\}, \{o', p\}$ | $\{o', o\}$ or $\{o', p\}$ reversed |
| $\{p, p'\}$ | $\{p, p'\}, \mathbf{\{o, p'\}}, J_1, J_2, \{p, p'\}, \{o, p'\}$ | $\{p, p'\}$ or $\{o, p'\}$ reversed |
| $\{q', q\}$ | $\{q', q\}, \mathbf{\{q', r\}}, J_2, J_1, \{q', q\}, \{q', r\}$ | $\{q', q\}$ or $\{q', r\}$ reversed |
| $\{r, r'\}$ | $\{r, r'\}, \mathbf{\{q, r'\}}, J_2, J_1, \{r, r'\}, \{q, r'\}$ | $\{r, r'\}$ or $\{q, r'\}$ reversed |

Apart from the fact that we do not know whether the labels in boldface are actually incomparable pairs, it is easy to check that each of the above label sequences defines an $L$–$M$-path. To find out how many of the bold labels are incomparable pairs, it is useful to call a comparable pair $\{x, y\}$ of $P$ *bad* if $\{x, y\}$ is a bold label of a sequence $s_J$ (cf. Table 3.1) such that $J$ is an incomparable pair. Let $I$ be defined by

$$I := \{i : \{\ell_i, \ell_{i+2}\} \text{ is a bad pair}\}.$$

Hence $|I|$ is the number of bad pairs and thus the number of edges incident with $L$ for which the corresponding label sequence does not define a path in $G(P)$. It follows from Lemma 3.6 that $|I| \leq \deg(L) - \delta(G(P))$, as desired—but there is one special case in which $I$ does not fulfill the conditions of this lemma, namely, the case that $p' = q'$ and $\{o, p'\}$ and $\{p', r\}$ are both bad. Then $o'opp'qrr'$ is a segment of $L$, and we know that $o \parallel p \parallel p' \parallel q \parallel r$ and $o <_P p' <_P r$ holds. This implies $o' <_P o$ and $r <_P r'$ because of the condition $\deg(L) \leq \deg(M)$ and Proposition 2.1. Hence $I$ has no other elements apart from $\{o, p'\}$ and $\{p', r\}$. We first shift down $p$ and then $q$

in $L$, and by Proposition 3.5 this results in a linear extension with degree at most $\deg(L) - 2$, so $2 = |I| \leq \deg(L) - \delta(G(P))$ holds also in this case.

It remains to check that the paths just defined do not have any inner vertices in common with previously constructed paths or with each other. This can be seen directly from the label sequences that were used for the definition of the paths in question: The paths constructed with the help of Table 3.1 are mutually independent because $K$ is an inner vertex of the path defined in line $i$ of this table if and only if one of the pairs listed in the third column of line $i$ is reversed in $K$. The paths from Case 3.1.1 are also independent of all previously constructed ones because the reversed pairs from the third column all have exactly one element in common with $J_1 \cup J_2$ and reversed pairs of this kind do not occur in the previous cases.

*Case* 3.1.2. There is no element of $P$ that strictly is between $p$ and $q$ in $L$. Hence $o'opqrr'$ is a segment of $L$. If $\{o', o\}$ (respectively, $\{r, r'\}$) is a jump, we define a corresponding sequence $s_{\{o',o\}}$ (respectively, $s_{\{r,r'\}}$), according to Table 3.1. We will later check how many of these sequences actually define $L$–$M$-paths, i.e., for how many of them the bold label is actually an incomparable pair. The independence of any of these paths from any other path defined so far is proved in the same way as in Case 3.1.1. If there is an edge $e$ incident to $L$ such that $\psi(e)$ equals $\{p, q\}$, we define $\{p, q\}$, $\{\mathbf{o}, \mathbf{q}\}$, $\{o, p\}$, $\{p, q\}$, $\{\mathbf{o}, \mathbf{r}\}$, $\{q, r\}$, $\{o, q\}$, $\{o, r\}$ as the corresponding sequence $s_{\{p,q\}}$ of edge labels. Again, we do not know whether the bold labels are incomparable pairs. But if we assume for the moment that this is the case, it is not difficult to check that the above sequence of labels defines an $L$–$M$-path. Note that the inner vertices of this path can be easily recognized because all pairs reversed in such a vertex are subsets of $J_1 \cup J_2$ and at least one of these pairs is different from $J_1$ and $J_2$. So this path has only the end vertices $L$ and $M$ in common with previously defined paths.

It remains to check that we have defined enough label sequences that actually correspond to paths in $G(P)$. In other words, we have to show that the following inequality holds:

$$\delta(G(P)) \leq \deg(L) - k, \tag{3.1}$$

where $k$ is the number of jumps $J$ in $o'opqrr'$ such that $s_J$ contains a bold label that is a comparable pair.

To turn the set of all neighboring pairs of $L$ into the corresponding set for $M$, we only have to replace three pairs by their counterparts from Table 3.2. From the condition $\deg(L) \leq \deg(M)$ and Proposition 2.1 it follows that there is at most one line in Table 3.2 such that the pair in the left column is an incomparable pair and its counterpart in the right column is not.

Table 3.2
*All neighboring pairs that are different in $L$ and $M$.*

| Neighboring pair of $L$ | Neighboring pair of $M$ |
|---|---|
| $\{o', o\}$ | $\{o', p\}$ |
| $\{r, r'\}$ | $\{q, r'\}$ |
| $\{p, q\}$ | $\{o, r\}$ |

We recall that a comparable pair $\{x, y\}$ of $P$ is called *bad* if $\{x, y\}$ is a bold label of a sequence $s_J$ such that $J$ is an incomparable pair. Additionally, we call $J$ the *owner* of the bad pair. Hence an owner of a bad pair is always an incomparable pair.

Note that the only pairs that can possibly be bad are $\{o', p\}$, $\{q, r'\}$, $\{o, r\}$, and $\{o, q\}$. According to what we have seen in the previous paragraph, only one of the three pairs in the right column of Table 3.2 can be bad, but not two or three at the same time. In Table 3.3 we list the arguments by which (3.1) holds for each possible combination of bad pairs; to see that the table is correct, it is important to note that the pairs $\{o, q\}$ and $\{o, r\}$ have the same owner. This means that if one of them is bad, the other one must either be bad or incomparable. And if they are both bad, the variable $k$ from (3.1) is one and not two because $k$ counts the number of sequences that contain a bad pair, not the bad pairs themselves.

TABLE 3.3
*This table is part of the proof of Theorem* 3.1. $i_0$ *is chosen such that* $\ell_{i_0} = o'$.

| Bad pairs | Owners | Reason why (3.1) holds |
|---|---|---|
| $\{o', p\}$ | $\{o', o\}$ | Lemma 3.6 with $I := \{i_0\}$ |
| $\{q, r'\}$ | $\{r, r'\}$ | Lemma 3.6 with $I := \{i_0 + 3\}$ |
| $\{o, q\}$ | $\{p, q\}$ | Lemma 3.6 with $I := \{i_0 + 1\}$ |
| $\{o, r\}$ | $\{p, q\}$ | shift down $p$ and afterwards $q$ in $L$, the resulting linear extension has lower degree by Proposition 3.5 |
| $\{o, q\}$ and $\{o', p\}$ | $\{p, q\}$ and $\{o', o\}$ | Lemma 3.6 with $I := \{i_0, i_0 + 1\}$ |
| $\{o, q\}$ and $\{q, r'\}$ | $\{p, q\}$ and $\{r, r'\}$ | shift down $p$ and afterwards $r$ in $L$, which by Proposition 3.5 lowers the degree by two |
| $\{o, q\}$ and $\{o, r\}$ | $\{p, q\}$ | Lemma 3.6 with $I := \{i_0 + 1\}$ |

*Case* 3.2. $J_1$ and $J_2$ have an element $p$ in common. In this case we can assume without loss of generality that there is a segment $p'pqrr'$ of $L$ such that $J_1 = \{p, q\}$ and $J_2 = \{p, r\}$. Note that for the independence of the paths to be defined in the following, we do not need to consider the paths from Case 3.1 because the assumptions of Case 3.1 and Case 3.2 concerning $J_1 \cap J_2$ mutually exclude each other. We have three possibilities for $\psi(e)$ in the given situation.

*Case* 3.2.1. $\psi(e) = \{p', p\}$. If $p' <_P q$, we need not construct a path because by Proposition 3.5 the linear extension generated from $L$ by shifting down $p$ has lower degree than $L$. If $p'$ and $q$ are incomparable, there is a path with edges labeled $\{p', p\}$, $\{p', q\}$, $\{p, q\}$, $\{p', p\}$, $\{p, r\}$, and $\{p', q\}$. The inner vertices of this path are easily recognized by the following property: Only members of

$$\big\{\{p', p\}, \{p', q\}, J_1, J_2\big\}$$

are reversed, and at least one of the reversed pairs is from the set $\big\{\{p', p\}, \{p', q\}\big\}$. This distinguishes the inner vertices of paths defined in the current case from those constructed in Cases 1 and 2.

*Case* 3.2.2. $\psi(e) = \{q, r\}$. In this case there is a path with edge labels $\{q, r\}$, $\{p, r\}$, $\{p, q\}$, $\{q, r\}$. The inner vertices of this path are easily recognized by the fact that the pair $\{q, r\}$ is reversed, which is never the case in any inner vertex of any path defined before.

*Case* 3.2.3. $\psi(e) = \{r, r'\}$. We have to consider two subsubcases here.

*Case* 3.2.3.1. $p <_P r'$. Then we have $p' \parallel q$ and $p' <_P p$ because otherwise $L$ would have greater degree than $M$ (cf. Proposition 2.1). We choose the path corresponding to the label sequence $\{r, r'\}$, $\{p, q\}$, $\{p', q\}$, $\{r, r'\}$, $\{p, r\}$, $\{p', q\}$. Because of $p' <_P p$, Case 3.2.3.1 is only relevant if $\{p', p\}$ is *not* a jump of $L$. So we do not require this path to be independent of the one defined in Case 3.2.1, where we assumed that $\{p', p\}$ is a jump of $L$. For the independence of the other previously constructed paths note that only pairs from

$$\{\{r, r'\}, \{p', q\}, J_1, J_2\}$$

are reversed, and there is always a reversed pair that is different from $J_1$ and $J_2$.

*Case* 3.2.3.2. $p$ and $r'$ are incomparable. In this case the labels $\{r, r'\}$, $\{p, q\}$, $\{p, r'\}$, $\{p, r\}$, $\{r, r'\}$, and $\{p, r'\}$ define our path. Of course we do not require this path to be independent of the one defined in Case 3.2.3.1, since that case is only relevant for $p <_P r'$. It remains to show the independence of the other previously defined paths, and the characterization for the inner vertices of our path is almost the same as in the previous case. Only pairs from

$$\{\{r, r'\}, \{p, r'\}, J_1, J_2\}$$

are being reversed, and there is always a reversed pair that is different from $J_1$ and $J_2$. □

**4. Isometric cycles and the cycle space.** In this chapter we investigate the cycle space (always over the two-element field $\mathbb{F}_2$) of graphs of linear extensions. For general information on the cycle space of a graph see, e.g., [3]. The basic question we want to consider comes up naturally when one looks at drawings of such graphs like the one in Figure 4.1: Is the cycle space of the graph of linear extensions generated by the 4- and 6-cycles? The answer is affirmative, and we don't even need additional assumptions, as the main theorem of this section shows.

THEOREM 4.1. *For every poset $P$, the cycle space of $G(P)$ is generated by the cycles of lengths* 4 *and* 6.

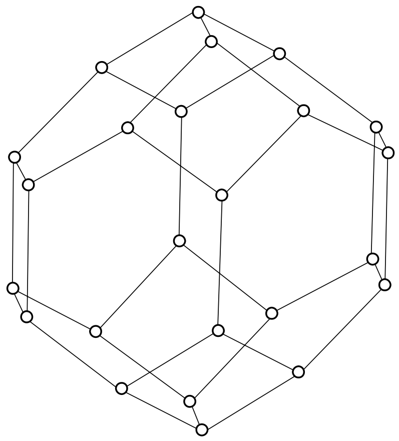


FIG. 4.1. *$G(A_4)$ for the* 4*-element antichain $A_4$.*

Our proof strategy will be to show first that the cycle space is generated by the isometric cycles of a graph (see the definition below; this result holds for arbitrary graphs), and then we will break down the isometric cycles into 4- and 6-cycles.

DEFINITION 4.2. *A subgraph $H$ of a graph $G$ is called an isometric subgraph of $G$ if $d_H(v, w) = d_G(v, w)$ holds for every two vertices $v$ and $w$ of $H$. An isometric subgraph that is a cycle will be called an isometric cycle.*

*Remark* 4.3. Note that we could equivalently demand that at least one of the shortest $v$–$w$-paths be a subgraph of $H$ for every two vertices $v$ and $w$ of $H$—where shortest paths are of course meant to be shortest with respect to $G$, not $H$. In particular, subgraphs that contain *every* shortest path between every two of their end vertices—so-called convex subgraphs—are also isometric subgraphs. Convex subgraphs play an important role in the investigation of the graph of linear extensions in [1] and [13].

The following theorem is a natural strengthening of the well-known fact that the induced cycles generate the cycle space, and the proof is based on the same idea. There doesn't seem to be a reference for this theorem in the literature, although it seems to be well known among the researchers in this area of graph theory, according to Victor Chepoi (personal communication). For the sake of completeness we include the proof. It is easy to see that every isometric subgraph is induced (but not vice versa), so this theorem is in fact stronger than the induced version.

THEOREM 4.4. *The cycle space of a graph $G$ is generated by the isometric cycles of $G$.*

*Proof.* Let $C$ be a cycle of $G$. We show by induction on the length of $C$ that $C$ is a sum of isometric cycles with respect to the vector addition in the cycle space. For a cycle of length 3 this is obvious because such a cycle is itself isometric. Now let $C$ be nonisometric. Then there is a path $P = v_0v_1 \dots v_k$ in $G$ such that $v_0$ and $v_k$ are vertices of $C$ and $k < d_C(v_0, v_k)$. Let $v_0 = v_{i_0}, v_{i_1}, \dots, v_{i_{m-1}}, v_{i_m} = v_k$ be the common vertices of $C$ and $P$ in their natural order along $P$ (which means $i_0 < i_1 < \cdots < i_m$). So for every pair $v_{i_j}, v_{i_{j+1}}$ there is a $v_{i_j}$–$v_{i_{j+1}}$-path in $P$. At least one of these paths must have a length smaller than the corresponding distance $d_C(v_{i_j}, v_{i_{j+1}})$ in $C$. Let us call this path $P'$, and let us call its end vertices $v'$ and $w'$. It is clear from the choice of the indices $i_1, \dots i_m$ that $v'$ and $w'$ are the only common vertices of $P'$ and $C$. In $C$ there are exactly two paths $P_1$ and $P_2$ with end vertices $v'$ and $w'$. $P_1$, $P_2$, and $P'$ are independent. See Figure 4.2 for an example of the situation. Now $C$ is the sum of two shorter cycles, namely, $P' \cup P_1$ and $P' \cup P_2$, and these are sums of isometric cycles by induction. ∎

*Remark* 4.5. It is easily seen from the proof that the isometric cycles share another interesting feature with the induced cycles: Every cycle that is not isometric (respectively, induced) is a sum of *shorter* isometric (respectively, induced) cycles. This comes in very handy when one uses induction as in the proof of Theorem 4.1 below.

All graphs of linear extensions are bipartite, as is easily seen by viewing the linear extensions as permutations and observing that an edge always joins an even to an odd permutation. So all cycles have even length, and in such cycles it is clear what is meant by opposite edges: $\{L, L'\}$ and $\{M, M'\}$ are opposite edges of a cycle $C$ when $d_C(L, M) = d_C(L', M')$.

THEOREM 4.6. *Let $P$ be a poset, and $e \neq f$ edges of an isometric cycle $C$ of $G(P)$. Then $e$ and $f$ are opposite if and only if $\psi(e) = \psi(f)$.*

*Proof.* "$\Rightarrow$": Let $e = \{L, L'\}$ and $f = \{M, M'\}$ be opposite edges of $C$ and, without loss of generality, $d(L, M') > d(L, M)$. There are two $L$–$M$-paths in $C$; let $W$ be the longer one. According to Propositions 2.2 and 2.3, $W$ has exactly one pair of edges with the same label. But when we delete the first or the last vertex of $W$, we get a shortest path from $L$ to $M'$ or from $L'$ to $M$, respectively. But on a shortest path no label occurs twice (Proposition 2.3), so $e$ and $f$ are the two edges with the same label.

"$\Leftarrow$": Now let $e$ and $f$ be *non*opposite edges of $C$. Choose $L$ and $M$ from the end vertices of $e$ and $f$ such that $d(L, M)$ is maximal. Then $e$ and $f$ are contained in

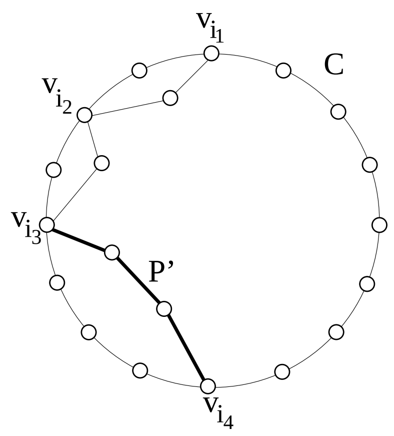


FIG. 4.2. *Example for the situation in the proof of Theorem* 4.4.

a shortest $L$–$M$-path, and by Proposition 2.3 they have different labels. □

COROLLARY 4.7. *Let $P$ be a poset. A cycle $C$ of $G(P)$ is isometric if and only if the following two statements are equivalent for every pair $e, f$ of distinct edges of $C$:*

(i) *$e$ and $f$ have the same label.*
(ii) *$e$ and $f$ are opposite.*

*Proof.* We have already shown one direction of the proof in the previous theorem. For the other direction let $C$ be a cycle of $G(P)$ for which statements (i) and (ii) are equivalent. For every two vertices $L$ and $M$ of such a cycle there is always an $L$–$M$-path such that no two edges of this path have the same label. By Proposition 2.3 this is a shortest path, and thus $d_C(L, M) = d_{G(P)}(L, M)$. □

LEMMA 4.8. *Let $P$ be a poset, and let $C$ be a cycle of $G(P)$ of length at least* 8. *Then at least one of the following statements is true.*

(i) *There are at least two consecutive edges of $C$ that are contained in a common* 4*-cycle.*

(ii) *There are at least three consecutive edges of $C$ that are contained in a common* 6*-cycle.*

*Proof.* Two incident edges which have disjoint edge labels are contained in a common 4-cycle. So we assume that $C$ is a cycle where incident edges never have disjoint edge labels.

It turns out that there is no element of $P$ that occurs in the label of every edge of $C$. Such an element would always be interchanged with its right neighbor when we move from vertex to vertex in the cycle $C$ in a fixed direction (and always with the left one when we go the other way round), but in this way we would never get back to where we started from and hence $C$ would not be a cycle.

Hence we know that there have to be three consecutive edges $e_1$, $e_2$, $e_3$ in $C$ with

$$\psi(e_1) \cap \psi(e_2) \neq \psi(e_2) \cap \psi(e_3).$$

Since the labels of two consecutive edges cannot have more than one element in common, the intersection of the three labels is empty:

$$\psi(e_1) \cap \psi(e_2) \cap \psi(e_3) = \emptyset.$$

In other words, there have to be four consecutive vertices $L_1, \dots, L_4$ of $C$ with the property that

$$\psi(\{L_1, L_2\}) \cap \psi(\{L_2, L_3\}) \cap \psi(\{L_3, L_4\}) = \emptyset. \tag{4.1}$$

Let $\{p, q\} := \psi(\{L_1, L_2\})$ and $\{p, r\} = \psi(\{L_2, L_3\})$. Because of our assumption that labels of two consecutive edges are never disjoint it follows from (4.1) that $r \in \psi(\{L_3, L_4\})$. Obviously $pqr$ (respectively, $rqp$) is a segment of $L_1$ and $qrp$ (respectively, $prq$) is a segment of $L_3$. Together with (4.1), this implies that $\psi(\{L_3, L_4\}) = \{q, r\}$. So we have a path $L_1 \dots L_4$ in $G(P)$ with edge labels of the form $\{p, q\}$, $\{p, r\}$, $\{q, r\}$, and we easily verify that such a path is always contained in a cycle of length 6. □

*Proof of Theorem* 4.1. Because of Theorem 4.4 it only remains to show that every isometric cycle of $G(P)$ is a sum of 4- and 6-cycles. This is trivial for 4- and 6-cycles, and to prove the theorem for an arbitrary isometric cycle we use induction on its length. Let $C$ be an isometric cycle of $G(P)$ with length greater than 6. According to Lemma 4.8 there is a cycle $C'$ of $G(P)$ of length 4 (respectively, 6) that has at least 2 (respectively, 3) edges in common with $C$. In both cases, $C + C'$ (with + meaning the addition defined in the cycle space of $G(P)$) is not longer than $C$. If $C + C'$ has strictly smaller length it is—by Theorem 4.4 and Remark 4.5—a sum of isometric cycles each of which is shorter than $C$, and we are done by induction.

If $C$ and $C + C'$ have the same length, adding $C'$ to $C$ just means replacing two consecutive edges by new ones—respectively, three edges if $C'$ is a 6-cycle. This replacement reverses the order of the affected edge labels, so there is always an edge $e$ that is replaced by an edge $f$ with a different label. The edge $e'$ that is opposite to $e$ in $C$ is not replaced because there are at most three consecutive edges being changed. Hence the opposite edges $f$ and $e'$ of $C + C'$ have different labels. This implies by Theorem 4.6 that $C + C'$ is not isometric. By Theorem 4.4 and Remark 4.5, $C + C'$ is a sum of shorter isometric cycles, and we are done by induction. □

## 5. Isometric embeddings.

**5. Isometric embeddings.** In this section we want to analyze the metric structure of graphs of linear extensions. The standard approach is to embed the graph in question as an isometric subgraph into a Cartesian product of graphs such that every factor is prime with respect to Cartesian products (for a detailed account of this method see [2, Part III]). However, in the context of order theory, embeddings into products with nonprime factors also deserve to be studied as long as they permit an order-theoretic interpretation. This is the case when the factors are themselves graphs of linear extensions as in the following theorem.

THEOREM 5.1. *Let $P$ be a poset with subposets $P_1, \dots, P_m$, such that every incomparable pair of $P$ is contained in exactly one $P_i$. Then $G(P)$ is isomorphic to an isometric subgraph $H$ of $\prod_{i=1}^m G(P_i)$.*

*Proof.* Let $f : V(G(P)) \to \prod_{i=1}^m V(G(P_i))$ be defined by $L \mapsto (L|_{P_1}, \dots, L|_{P_m})$. It follows directly from the hypothesis that $f$ is well defined and injective. Now let $L$

and $M$ be linear extensions of $P$. Then Proposition 2.2 yields

$$\begin{aligned}
d_{G(P)}(L,M) &= \frac{1}{2}|\mathcal{R}(L) \triangle \mathcal{R}(M)| \\
&= \frac{1}{2}|(\mathcal{R}(L) \cap \mathrm{Inc}(P)) \triangle (\mathcal{R}(M) \cap \mathrm{Inc}(P))| \\
&= \frac{1}{2}\Big|\biguplus_{i=1}^{m}\big[(\mathcal{R}(L) \cap \mathrm{Inc}(P_i)) \triangle (\mathcal{R}(M) \cap \mathrm{Inc}(P_i))\big]\Big| \\
&= \sum_{i=1}^{m} \frac{1}{2}|\mathcal{R}(L|_{P_i}) \triangle \mathcal{R}(M|_{P_i})| \\
&= \sum_{i=1}^{m} d_{G(P_i)}\big((f(L))_i, (f(M))_i\big) \\
&= d_{\prod_{i=1}^{m} G(P_i)}(f(L), f(M)),
\end{aligned}$$

where $(f(L))_i$ is the $i$th entry of the $m$-tuple $f(L)$ and "$\uplus$" denotes disjoint union. So the image of $G(P)$ under $f$ is an isometric subgraph of $\prod_{i=1}^{m} G(P_i)$. □

*Remark* 5.2. Note that we do not require the subposets $P_1, \ldots, P_m$ to be induced. If any of the subposets, say $P_{i_0}$, is induced, the above embedding has the additional property that none of the vertices in $G(P_{i_0})$ is superfluous: If we modify the product $\prod_{i=1}^{m} G(P_i)$ by deleting vertices from $G(P_{i_0})$, $H$ will not be a subgraph of the modified product. That is to say, for every vertex $v$ of $G(P_{i_0})$ there exists a vertex $(v_1, \ldots, v_m)$ of $H$ with $v_{i_0} = v$. This follows from the well-known fact (see, e.g., [18, Theorem 8.1]) that every linear extension of an induced subposet $P'$ of $P$ is of the form $L|_{P'}$, where $L$ is a linear extension of $P$.

When we choose the incomparable pairs of $P$ as subposets $P_i$ we can derive from Theorem 5.1 that $G(P)$ is an isometric subgraph of a hypercube. This was already shown by Reuter [12, p. 5], but the proof of Theorem 5.1 yields a very natural embedding: Choose some linear extension $L$ to be mapped to the vertex of the hypercube whose coordinates are all zero. Then the $i$th coordinate of any other linear extension $L'$ is 1 if and only if the $i$th incomparable pair of $P$ (according to some given numbering) is ordered differently in $L$ and $L'$.

**5.1. Application to the linear-extension-diameter.** From Theorem 5.1 we easily derive a new bound on the linear-extension-diameter $\mathrm{led}(P)$ of a poset $P$. This parameter was introduced by Felsner and Reuter in [4], and it is defined as the diameter of the corresponding graph of linear extensions:

$$\mathrm{led}(P) = \mathrm{diam}(G(P)) = \max_{L,M \in V(G(P))} d(L,M).$$

COROLLARY 5.3. *Let $P$ be a poset with subposets $P_1, \ldots, P_m$, such that every incomparable pair of $P$ is contained in exactly one $P_i$. Then the following inequality holds:*

$$\mathrm{led}(P) \leq \sum_{i=1}^{m} \mathrm{led}(P_i). \tag{5.1}$$

*Proof.* The following is an easy consequence of Theorem 5.1:

$$\mathrm{led}(P) = \mathrm{diam}(G(P)) \leq \mathrm{diam}\left(\prod_{i=1}^{m} G(P_i)\right) = \sum_{i=1}^{m} \mathrm{diam}(G(P_i)) = \sum_{i=1}^{m} \mathrm{led}(P_i). \qquad □$$

When we again choose the $P_i$ to be the incomparable pairs of $P$, this corollary yields the inequality

$$\text{led}(P) \leq \text{inc}(P), \tag{5.2}$$

which is already known from [4]. Felsner and Reuter also prove that equality holds in (5.2) if and only if $P$ is at most 2-dimensional. This means that if we choose at least one $P_i$ with higher dimension, the bound from Corollary 5.3 will be better than (5.2). In fact, the following example shows that a poset need not have dimension 1 or 2 for (5.1) to be sharp. The poset $Q$ depicted in Figure 5.1 has dimension 3 because deleting element 1 results in a well-known 3-dimensional poset $C$ called the chevron, and it is not hard to find a realizer of $Q$ with three elements.

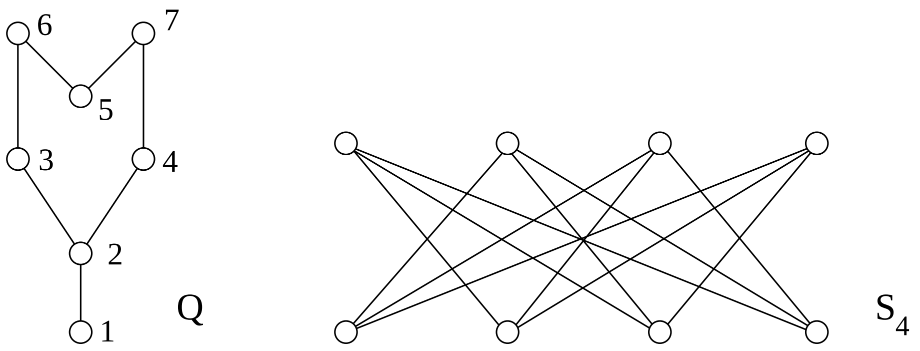


FIG. 5.1. *Two examples illustrating Corollary* 5.3.

From [4] we know that deleting an element $p$ of a poset $P$ makes the linear-extension-diameter drop by at most the number of elements that are incomparable to $p$. In our example $Q$, there is just one element incomparable to 1, hence $\text{led}(C) \geq \text{led}(Q)-1$. Also in [4], $\text{led}(C)$ is shown to be 6, hence $\text{led}(Q)$ can be at most 7. Indeed the linear extensions 1235647 and 5124736 have distance 7, so $\text{led}(Q) = 7$. If we define $Q_{15}$ to be the subposet of $Q$ induced be the elements 1 and 5, every incomparable pair of $P$ is contained either in $Q_{15}$ or in $C$. In this situation, Corollary 5.3 yields $\text{led}(Q) \leq \text{led}(C) + \text{led}(Q_{15}) = 6 + 1 = 7$, which is sharp. The same goes for the following inequality, which is deduced by Felsner and Reuter as an improved version of (5.2):

$$\text{led}(P) \leq \text{inc}(P) - (\dim(P) - 2). \tag{5.3}$$

In general, the inequalities (5.1) and (5.3) do not necessarily produce the same bound, and none of them is in general better than the other. While sometimes our new bound performs much worse than (5.3)—try it, e.g., on the standard example $S_n$ (see Figure 5.1 for $S_4$, and [18, p. 12] for the general definition)—there are also cases in which it is more precise. An easy example is the series-composition $S$ of 3-dimensional posets $P_1, \dots P_m$. In this situation, $\text{led}(S)$ is just the sum of the linear-extension-diameters of the individual $P_i$ (see [4]), hence inequality (5.1) is sharp. But it is easily verified that $\dim(S) = 3$, which means that (5.3) yields the following bound:

$$\text{inc}(S) - (\dim(S) - 2) = \sum_{i=1}^{m} \text{inc}(P_i) - 1 \geq \sum_{i=1}^{m} (\text{led}(P_i) + 1) - 1 = \text{led}(S) + m - 1.$$

Hence the gap between the two bounds can be arbitrarily large. It is not too difficult to construct other examples using the same basic idea.

**Acknowledgments.** The author thanks his supervisor Klaus Reuter for introducing him to the subject and providing the encouragement that made the diploma thesis [9] possible. The material in this article is taken from this thesis (with the exception of some improvements, mainly those in section 3). The author is grateful to Ekkehard Köhler whose suggestions greatly improved the presentation of the results.

## REFERENCES

[1] A. Björner and M. L. Wachs, *Permutation statistics and linear extensions of posets*, J. Combin. Theory Ser. A, 58 (1991), pp. 85–114.

[2] M. M. Deza and M. Laurent, *Geometry of Cuts and Metrics*, Springer-Verlag, Berlin, Heidelberg, New York, 1997.

[3] R. Diestel, *Graph Theory*, Springer-Verlag, Berlin, Heidelberg, New York, 1997.

[4] S. Felsner and K. Reuter, *The linear-extension-diameter of a poset*, SIAM J. Discrete Math., 12 (1999), pp. 360–373.

[5] H. Fleischner, *On spanning subgraphs of a connected bridgeless graph and their applications*, J. Combin. Theory Ser. B, 16 (1974), pp. 17–28.

[6] H. Fleischner, *The square of every two-connected graph is hamiltonian*, J. Combin. Theory Ser. B, 16 (1974), pp. 29–34.

[7] X. Li, *The connectivity of path graphs*, in Combinatorics, Graph Theory and Algorithms, Y. Alavi, D. R. Lick, and J. Liu, eds., World Scientific, River Edge, NJ, 1994, pp. 187–192.

[8] R. H. Möhring, *Computationally tractable classes of ordered sets*, in Algorithms and Order, I. Rival, ed., NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci. 255, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1989.

[9] M. Naatz, *Der Graph der linearen Erweiterungen einer geordneten Menge*, diploma thesis, Mathematisches Seminar der Universität Hamburg, Germany, 1998.

[10] G. Pruesse and F. Ruskey, *Generating the linear extensions of certain posets by transpositions*, SIAM J. Discrete Math., 4 (1991), pp. 413–422.

[11] G. Pruesse and F. Ruskey, *Generating linear extensions fast*, SIAM J. Comput., 23 (1994), pp. 373–386.

[12] K. Reuter, *The comparability graph and the graph of linear extensions of a poset*, Preprint Hamburger Beiträge zur Mathematik, Heft 57 (1996).

[13] K. Reuter, *Linear extensions of a poset as abstract convex sets*, Preprint Hamburger Beiträge zur Mathematik, Heft 56 (1996).

[14] S. Řiha, *A new proof of the theorem by Fleischner*, J. Combin. Theory Ser. B, 52 (1991), pp. 117–123.

[15] F. Ruskey and C. Savage, *Hamilton cycles that extend transposition matchings in Cayley graphs of $S_n$*, SIAM J. Discrete Math., 6 (1993), pp. 152–166.

[16] C. D. Savage and C.-Q. Zhang, *A note on the connectivity of acyclic orientation graphs*, Discrete Math., 184 (1998), pp. 281–287.

[17] G. Stachowiak, *Hamilton paths in graphs of linear extensions for unions of posets*, SIAM J. Discrete Math., 5 (1992), pp. 199–206.

[18] W. T. Trotter, *Combinatorics and Partially Ordered Sets: Dimension Theory*, Johns Hopkins Series in the Mathematical Sciences, The Johns Hopkins University Press, Baltimore, 1992.

[19] D. B. West, *Generating linear extensions by adjacent transpositions*, J. Combin. Theory Ser. B, 58 (1993), pp. 58–64.

SIAM J. Discrete Math. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 3, pp. 370–383

# THE ONLINE TRANSPORTATION PROBLEM*

BALA KALYANASUNDARAM† AND KIRK R. PRUHS†

**Abstract.** We study the online transportation problem under the assumption that the adversary has only half as many servers at each site as the online algorithm. We show that the Greedy algorithm is $\Theta(\min(m, \lg C))$-competitive under this assumption, where $m$ is the number of server sites and $C$ is the total number of servers. We then present an algorithm Balance, which is a simple modification of the Greedy algorithm, that is, $O(1)$-competitive under this assumption.

**Key words.** online algorithms, matching, competitive analysis

**AMS subject classifications.** 05C85, 05D15, 68R10

**PII.** S0895480198342310

## 1. Introduction.

We consider the natural online version of the well-known transportation problem [4, 6]. The initial setting consists of a collection $S = \{s_1, \dots, s_m\}$ of server sites in a metric space $\mathcal{M}$. Each server site $s_j$ has a positive integral capacity $c_j$. The online algorithm knows the servers sites and the capacities a priori. The online algorithm $\mathcal{A}$ sees over time a sequence $R = \{r_1, \dots r_n\}$ of requests for service, with each request being a point in $\mathcal{M}$. Note that we assume that $n$ is at most the aggregate server capacities. In response to the request $r_i$, $\mathcal{A}$ must select a site $s_{\sigma(i)}$ to service $r_i$. The cost for this assignment is the distance $d(s_{\sigma(i)}, r_i)$ in the metric space between $s_{\sigma(i)}$ and $r_i$. Each site $s_j$ can service at most $c_j$ requests. The dilemma faced by the online algorithm $\mathcal{A}$ is that, at the time of the request $r_i$, $\mathcal{A}$ is not aware of the location of the future requests. The objective function is to minimize $\frac{1}{n}\sum_{i=1}^{n} d(s_{\sigma(i)}, r_i)$, the *average cost* to service the requests. Note that this is equivalent to minimizing the *total cost* $\sum_{i=1}^{n} d(s_{\sigma(i)}, r_i)$.

For concreteness, consider the following two examples of online transportation problems. In the *fire station problem*, the site $s_j$ is a fire station that contains $c_j$ fire crews. Each request is a fire that must be handled by a fire crew. The problem is to assign the crews to the fire so as to minimize the average distance traveled to get to a fire. In the *school assignment problem*, the site $s_j$ is a school that has a capacity of $c_j$ students. Each request is a new student who moves into the school district. The problem is to assign the children to a school so as to minimize the average distance traveled by the children to reach their schools.

The standard measure of "goodness" of an online algorithm is the competitive ratio. For the online transportation problem, the competitive ratio for an online algorithm $\mathcal{A}$ is the supremum over all possible instances $I$, of $\mathcal{A}(I)/OPT(I)$, where $\mathcal{A}(I)$ is the total cost of the assignment made by $\mathcal{A}$ and $OPT(I)$ is the total cost of the minimum cost assignment for instance $I$. Note that as we are considering a finite problem, it does not make sense to allow an additive constant in the definition of competitive ratio that is used for infinite problems (see [1]). The standard way to

*Received by the editors July 24, 1998; accepted for publication (in revised form) February 28, 2000; published electronically May 26, 2000. A preliminary version of this paper appeared in the *Proceedings of the* 1995 *European Symposium on Algorithms.* The research of the first author was supported in part by NSF under grant CCR-9202158. The research of the second author was supported in part by NSF under grant CCR-9209283.

http://www.siam.org/journals/sidma/13-3/34231.html

†Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260 (kalyan@cs.pitt.edu, kirk@cs.pitt.edu).

interpret the competitive ratio is as a payoff of a game played by the online algorithm $\mathcal{A}$ against an all-powerful adversary that specifies the requests and services them in the optimal way. Note that the instance $I$ specifies the metric space as well as the values of each $s_j$, $c_j$, and $r_i$.

In [3, 5] the online assignment problem, a special case of online transportation in which each capacity $c_i = 1$, was studied. In [3], it was shown that the competitive ratio of the intuitively appealing greedy algorithm, which assigns the nearest available server site to the new request, has a competitive ratio of $2^m - 1$. In [3, 5], it was shown that the optimal deterministic competitive ratio is $2m-1$. The algorithm that achieves this competitive ratio requires a shortest augmenting path computation for each request. These results illustrate some shortfalls of using competitive analysis, namely:

- The achievable competitive ratios often grow quickly with input size, and thus do not reflect the performance on "normal" inputs.
- The algorithm that achieves the optimal competitive ratio is often unnecessarily complicated for "normal" inputs.
- The poor competitive ratio of an intuitive greedy algorithm may not reflect the fact that it may perform reasonably well on "normal" inputs.

In situations where competitive analysis suffers such shortcomings, it is important to find alternate ways to identify online algorithms that would work well in practice. In this paper, we adopt a modified version of competitive analysis called the *resource augmentation model*. Generally speaking, in this model the online algorithm is given slightly more resources than the adversary. For "normal" inputs, one might expect that the performance of an *offline* algorithm would not degrade significantly if its resources were slightly reduced. Hence, if we can prove that an online algorithm is competitive against an adversary with slightly less resources, then one might argue that the online algorithm will be competitive against an equivalently equipped offline algorithm on "normal" inputs. One can also view this augmentation as measuring the additional resources required by the online algorithm to offset the decrease in performance due to the online nature of the problem.

In the case of the transportation problem we compare the online algorithm with $c_i$ servers at $s_i$ to an offline line algorithm with $a_i = c_i/2$ servers at $s_i$ (we assume that $c_i$ is even). Given an instance $I$ of the online transportation problem with $n$ requests, $n \leq \sum_{i=1}^{m} a_i$, we let $I'$ be the same instance with each capacity $c_i$ replaced by $a_i$. We then say the *double-competitive ratio* of an online algorithm $\mathcal{A}$ is the supremum over all instances $I$, with $n \leq \sum_{i=1}^{m} a_i$ requests, of the ratio $\mathcal{A}(I)/OPT(I')$. We assume that the online algorithm has twice as many servers as the adversary because this is the least advantage that we can give to the online algorithm without annulling our analysis techniques.

In this paper we present the following results. In section 3, we show that the double-competitive ratio of the greedy algorithm is $\Theta(\min(m, \lg C))$, where $C = \sum_{i=1}^{m} c_i$ is the sum of the capacities. If the server capacity of each site is constant, then the double-competitive ratio is logarithmic in $m$, a significant improvement over the exponential bound on the traditional competitive ratio. In section 4, we describe the algorithm BALANCE, which is a simple modification of the greedy algorithm, and has a double-competitive ratio that is $O(1)$. Recall that the traditional competitive ratio of every deterministic online algorithm is $\Omega(m)$.

We now summarize related results. The resource augmentation model was introduced in [8] in the context of studying paging. This model has also been used to study variants of the $k$-server problem, a generalization of the paging problem (see, for example, [10]). References to other suggested variants of competitive analysis can be found in [1, 2]. Further, ancillary results on online assignment, which are not directly related to the results in this paper, can be found in [3]. In [9], the average competitive ratio for the greedy algorithm in the online assignment problem is studied under the assumption that the metric space is the Euclidean plane and the points are uniformly distributed in a unit square. The offline transportation problem can be solved in polynomial time [4, 6].

**2. Preliminaries.** In this section we introduce some definitions, facts, and concepts that are common to the remaining sections. We generally begin by assuming the simplifying condition that the online capacity $c_i$ of each server site is two. We will think of $s_i$ as containing two *online servers* $s_i^1$ and $s_i^2$ that move to service requests. We also think of $s_i$ as containing one *adversary server* $s_i^a$. We assume, without loss of generality, that the adversary services request $r_i$ with $s_i^a$. We use $s_{\sigma(i)}$ to denote the site that the online algorithm uses to service request $r_i$. We define a weighted bipartite graph $\mathcal{G} = (S \cup R, E)$, which we call the *response graph*, by including an *online edge* $(r_i, s_{\sigma(i)})$, and an *adversary edge* $(r_i, s_i)$ for each request $r_i$. The weight of each edge $(r_i, s_j)$ in $\mathcal{G}$ is the distance $d(r_i, s_j)$ between the $r_i$ and $s_j$ in the underlying metric space. In Figure 1, the online edges are the solid edges, the adversary edges are the dashed edges, the server sites are the filled circles, and the requests are the question marks. The notation shown in Figure 1 will be used throughout this paper.

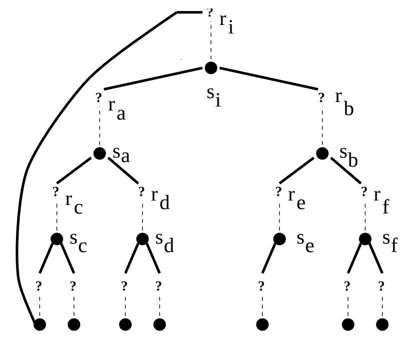


FIG. 1. *An example connected component of the response graph.*

LEMMA 1. *Assume $r_i$ is a request vertex that is in a cycle in $\mathcal{G}$. Let $\mathcal{T}$ be the connected component of $\mathcal{G} - (s_{\sigma(i)}, r_i)$ that contains $r_i$. Then $\mathcal{T}$ is a tree.*

*Proof.* Consider a breadth first search of $\mathcal{T}$ starting from $r_i$. Such a search divides $\mathcal{T}$ into levels $L_1, \ldots, L_b$, where the vertices in level $L_k$ are $k$ hops from $r_i$. The vertices in the odd levels are server vertices, and the vertices in even levels are request vertices. Edges that go from an even level to an odd level are adversary edges, and edges that go from an odd level to an even level are online edges. Assume in order to reach a contradiction that $C$ is a cycle in $\mathcal{T}$. Recall that we measure height from the root and the root is at level $L_1$. Let $y$ be the vertex in $C$ that is in the highest level $L_c$ (i.e., largest $c$), and let $x$ and $z$ be the two vertices adjacent to $y$ in $C$. Note that it may be the case that $x = z$. We now argue that $x$ and $z$ are in $L_{c-1}$. If $x$ (or $z$) is in $L_{c+1}$, we get a contradiction to the choice of $c$. On the other hand, if $x$ (or $z$) is in $L_{c-j}$ where $j > 1$, we get a contradiction to the level definition of $y$. Finally, if $x$ (or $z$) is in $L_c$, then we get a contradiction to the bipartiteness of $\mathcal{G}$. If $c$ is odd, then we get a contradiction to the fact that the adversary has only one server per site. If $c$ is even, we get a contradiction to the fact that the online algorithm uses only one server to service each request. $\square$

Let $\mathcal{T}$ be a tree as described in Lemma 1. If we root the tree $\mathcal{T}$ at $r_i$, then $\mathcal{T}$ has the following structure. For each request $r_j \in \mathcal{T}$, the one child of $r_j$ is $s_j$. If $r_j$ is not the root, then the server site $s_{\sigma(i)}$ is the parent of $r_j$ in $\mathcal{T}$. The leaves of $\mathcal{T}$ are server sites with no incident online edges. We denote the total cost of the adversary edges in a tree $\mathcal{T}$ by $OPT(\mathcal{T})$, that is, $OPT(\mathcal{T}) = \sum_{(r_i,s_i)\in\mathcal{T}} d(r_i, s_i)$. Analogously, we define $ON(\mathcal{T}) = \sum_{r_i\in\mathcal{T}} d(r_i, s_{\sigma(i)})$. Note that $ON(\mathcal{T})$ includes the cost of the online edge incident to the root of $\mathcal{T}$, even though this edge is not in $\mathcal{T}$. For a vertex $x \in \mathcal{T}$, we define the leaf distance $ld(x)$ to be the minimum over all leaves $s_j$ in $\mathcal{T}$ of the distance between $x$ and $s_j$. If a server at site $s_j$ servicing the root $r_i$ of $\mathcal{T}$ is not a node in $\mathcal{T}$, then we define $ld(s_j) = d(s_j, r_i) + ld(r_i)$; we make this definition so that such an $s_j$ need not be considered as a special case in our proofs. If $x$ is a node in $\mathcal{T}$, we define $\mathcal{T}(x)$ to be the subtree of $\mathcal{T}$ rooted at $x$. Intuitively, the leaf distance gives us an upper bound on the greedy cost to service a request since a server at the leaf is available to service the request.

In this paper, lg means the logarithm base 2.

## 3. Analysis of the Greedy algorithm.

We begin with the upper bound on the competitive ratio for the algorithm Greedy, which uses the nearest available server to service each request. We first assume that the online capacity of each server site is two and then show how to extend this to the general case of an arbitrary number of servers per site. This extension does not immediately follow by breaking a site into many sites, with two servers per site, since our bounds are in terms of the number of sites.

Theorem 2. *The double-competitive ratio of* Greedy *for online transportation, with $m \geq 2$ sites and two online servers per site, is at most $2\lg m$.*

In order to prove this theorem, we will divide the response graph $\mathcal{G}$ into edge disjoint rooted trees, $\mathcal{T}^1, \ldots, \mathcal{T}^l$. For each such tree, we will establish the competitive bound independently.

Our construction yields trees ($\mathcal{T}^j$'s) that satisfy the following *tree invariants*:

(1) Each nonleaf server site $s_i$ in $\mathcal{T}^j$ has two incident online edges in $\mathcal{T}^j$.

(2) Each leaf of $\mathcal{T}^j$ is a server site that had an unused server at the time of each request in $\mathcal{T}^j$.

We use the following iterative procedure to construct the trees.

**Tree construction procedure.** Assume that trees $\mathcal{T}^1, \ldots, \mathcal{T}^{j-1}$ have been constructed. We explain how to construct $\mathcal{T}^j$ in our next iteration. During this construction we will modify $\mathcal{G}$. The root of $\mathcal{T}^j$ is the latest request $r_{\alpha(j)}$ not included in a previous tree $\mathcal{T}^1, \ldots, \mathcal{T}^{j-1}$. The online edge incident to $r_{\alpha(j)}$ is removed from $\mathcal{G}$. Let $L_j$ be the collection of server vertices $s_i$ such that $s_i$ is reachable from $r_{\alpha(j)}$ and $s_i$ currently has at most one incident online edge in $\mathcal{G}$. Note that an $s_i \in L_j$ might have originally had two incident online edges if one or both of these edge lead to the root of one of the trees $\mathcal{T}^1, \ldots, \mathcal{T}^j$. Now let $\mathcal{T}^j$ be the tree induced by the edges on paths in $\mathcal{G}$ from $r_{\alpha(j)}$ to the server vertices in $L_j$. Note that by Lemma 1 there is a unique path from $r_{\alpha(j)}$ to each vertex in $L_j$. The edges and request vertices in $\mathcal{T}^j$ are then removed from $\mathcal{G}$, and we proceed to our next iteration to construct $\mathcal{T}^{j+1}$ if $\mathcal{G}$ still contains edges.

Lemma 3. *Each tree $\mathcal{T}^j$ satisfies the two tree invariants.*

*Proof.* Observe that each vertex in $L_j$ is a leaf server site that has at most one incident online edge in $\mathcal{T}^j$. Therefore, by the definition of $L_j$, all other server sites in $\mathcal{T}^j$ have two incident online edges each. Suppose that the second invariant is not satisfied for a server site $s_i \in L_j$. First observe that there are two online edges incident on $s_i$ in the original $\mathcal{G}$. Now observe that at least one of these two online edges must be incident on the root of one the trees $\mathcal{T}^1, \dots, \mathcal{T}^j$. Otherwise, either $s_i \notin L_j$ or $s_i$ would be part of one of the trees $\mathcal{T}^1, \dots, \mathcal{T}^{j-1}$. We now get a contradiction since the request nodes at the root of $\mathcal{T}^1, \dots, \mathcal{T}^j$ appear later than any other request nodes in $\mathcal{T}^j$. $\square$

**3.1. Analysis of a tree.** We first provide some intuition. We establish in Lemma 4 that the cost of a greedy edge is at most twice the cost of the adversaries' edges in the subtree below this greedy edge of minimum cost. Hence, intuitively the worst case is when these two subtrees are identical and the tree is then balanced. In this case we can conclude that the sum of the costs of the greedy edges in any one level of the tree is at most $OPT$. The logarithmic bound on the double-competitive ratio follows since a balanced tree has logarithmic depth. In the general case of an arbitrary tree, the first tree invariant allows us to inductively establish our desired bound.

We now fix a particular tree, say $\mathcal{T} = \mathcal{T}^j$, and for simplicity drop the $j$ superscript.

LEMMA 4. *For each request $r_i \in \mathcal{T}$,*

$$d(s_{\sigma(i)}, r_i) \le ld(r_i) \le OPT(\mathcal{T}(r_i))$$

*and*

$$ld(r_i) \le d(r_i, s_i) + 2 \cdot \min[OPT(\mathcal{T}(r_a)), OPT(\mathcal{T}(r_b))], \tag{1}$$

*where $r_a$ and $r_b$ are children, if both exist, of $s_i$ in $\mathcal{T}$.*

*Proof.* The proof is by induction on the number $k$ of request nodes in the induced tree $\mathcal{T}(r_i)$. If $k = 1$, then the child $s_i$ of $r_i$ is a leaf and $\mathcal{T}(r_i)$ consists of one adversary edge $(r_i, s_i)$ with $d(r_i, s_i) = ld(r_i) = OPT(\mathcal{T}(r_i))$. Applying the second tree invariant property and the definition of GREEDY, we have that $d(s_{\sigma(i)}, r_i) \le d(s_i, r_i) = OPT(\mathcal{T}(r_i))$.

Now suppose $k > 1$. Therefore, $s_i$ is not a leaf. Applying the first property of the tree invariant, $s_i$ has two children in $\mathcal{T}(r_i)$, say $r_a$ and $r_b$ (see Figure 1). By induction, it must be the case that $ld(r_a) \le OPT(\mathcal{T}(r_a))$, $d(s_i, r_a) \le OPT(\mathcal{T}(r_a))$, $ld(r_b) \le OPT(\mathcal{T}(r_b))$, and $d(s_i, r_b) \le OPT(\mathcal{T}(r_b))$. Therefore, by induction and the triangle inequality,

$$\begin{aligned}
ld(r_i) &\le d(r_i, s_i) + \min[d(s_i, r_a) + ld(r_a), d(s_i, r_b) + ld(r_b)] \\
&\le d(r_i, s_i) + \min[d(s_i, r_a) + OPT(\mathcal{T}(r_a)), d(s_i, r_b) + OPT(\mathcal{T}(r_b))] \\
&\le d(r_i, s_i) + \min[2 \cdot OPT(\mathcal{T}(r_a)), 2 \cdot OPT(\mathcal{T}(r_b))] \\
&\le d(r_i, s_i) + OPT(\mathcal{T}(r_a)) + OPT(\mathcal{T}(r_b)) \\
&= OPT(\mathcal{T}(r_i)).
\end{aligned}$$

Finally, observe that the algorithm GREEDY would only assign $s_{\sigma(i)}$ to $r_i$ if $d(s_{\sigma(i)}, r_i) \le ld(r_i)$. $\square$

LEMMA 5. *Let $r_i$ be a request in $\mathcal{T}$, and let $k$ be the number of request vertices in $\mathcal{T}(r_i)$. Then $ON(\mathcal{T}(r_i)) \le 2 \cdot \max(1, \lg k) \cdot OPT(\mathcal{T}(r_i))$.*

*Proof.* The proof is by induction on $k$. Observe that, according to our tree construction, each server node has either two children or none. Therefore, $k$ must be odd.

For $k = 1$, $ON(\mathcal{T}(r_i)) = d(s_{\sigma(i)}, r_i)$, which is less than $OPT(\mathcal{T}(r_i)) = d(r_i, s_i)$ by the definition of Greedy.

Now consider the case $k = 3$. Let the two children of $s_i$ in $\mathcal{T}(r_i)$ be $r_a$ and $r_b$. Note that

$$ON(\mathcal{T}(r_i)) = d(s_{\sigma(i)}, r_i) + d(s_i, r_a) + d(s_i, r_b)$$

and that

$$OPT(\mathcal{T}(r_i)) = d(r_i, s_i) + d(s_a, r_a) + d(s_b, r_b).$$

By the definition of Greedy, $d(s_i, r_a) \leq d(s_a, r_a)$ and $d(s_i, r_b) \leq d(s_b, r_b)$. By Lemma 4, $d(s_{\sigma(i)}, r_i) \leq OPT(\mathcal{T}(r_i))$. Hence, by substitution,

$$ON(\mathcal{T}(r_i)) \leq OPT(\mathcal{T}(r_i)) + d(s_a, r_a) + d(s_b, r_b) \leq 2 \cdot OPT(\mathcal{T}(r_i)).$$

Now consider the case $k > 3$. Once again let the two children of $s_i$ in $\mathcal{T}(r_i)$ be $r_a$ and $r_b$. Notice that $ON(\mathcal{T}(r_i)) = ON(\mathcal{T}(r_a)) + ON(\mathcal{T}(r_b)) + d(s_{\sigma(i)}, r_i)$. By (1),

$$d(s_{\sigma(i)}, r_i) \leq d(r_i, s_i) + 2 \cdot \min[OPT(\mathcal{T}(r_a)), OPT(\mathcal{T}(r_b))].$$

Hence,

$$\begin{aligned} ON(\mathcal{T}(r_i)) &\leq ON(\mathcal{T}(r_a)) + ON(\mathcal{T}(r_b)) \\ &\quad + d(r_i, s_i) + 2 \cdot \min[OPT(\mathcal{T}(r_a)), OPT(\mathcal{T}(r_b))]. \end{aligned} \tag{2}$$

Assume, without loss of generality, that $OPT(\mathcal{T}(r_a)) \leq OPT(\mathcal{T}(r_b))$. Let $w = OPT(\mathcal{T}(r_a)) + OPT(\mathcal{T}(r_b))$ and $x = OPT(\mathcal{T}(r_a))$. Hence, $x \leq w/2$. Also let $y$ be the number of request nodes in $\mathcal{T}(r_a)$. We now break the proof into cases.

In the first case, we assume that both $\mathcal{T}(r_a)$ and $\mathcal{T}(r_b)$ consist of more than one request vertex. By first applying the tree invariant property, we find that $s_a$ and $s_b$ cannot be leaf server sites of $\mathcal{T}$, since otherwise at least one of $\mathcal{T}(r_a)$ or $\mathcal{T}(r_b)$ will contain only one request vertex. Since $s_a$ and $s_b$ are not leaf server nodes, both must have two request children each, and thus the number of request vertices in each of $\mathcal{T}(r_a)$ and $\mathcal{T}(r_b)$ is at least 3. Hence, $3 \leq y \leq k-4$. By induction $ON(\mathcal{T}(r_a)) \leq 2x \lg y$ and $ON(\mathcal{T}(r_b)) \leq 2(w - x) \lg(k - 1 - y)$. Hence, by substituting into (2) we get

$$ON(\mathcal{T}(r_i)) \leq 2x \lg y + 2(w - x) \lg(k - 1 - y) + 2x + d(r_i, s_i).$$

Since $OPT(\mathcal{T}(r_i)) = w + d(r_i, s_i)$, in order to show $ON(\mathcal{T}(r_i)) \leq 2 \lg k \cdot OPT(\mathcal{T}(r_i))$, it is sufficient to show

$$2x \lg y + 2(w - x) \lg(k - 1 - y) + 2x + d(r_i, s_i) \leq 2(w + d(r_i, s_i)) \lg k.$$

In turn it is sufficient to show that

$$x \lg y + (w - x) \lg(k - 1 - y) + x \leq w \lg k. \tag{3}$$

Let $f(x, y) = x \lg y + (w - x) \lg(k - 1 - y) + x$ be the left-hand side of this inequality. Notice that $f(x, y)$ is linear in $x$. Hence, the maximum of $f(x, y)$ must occur at the boundary, that is, at the point $x = 0$ or the point $x = w/2$. Inequality (3) follows immediately for $x = 0$. For $x = w/2$ we have to find the value of $y$ that maximizes

$f(w/2, y) = (w/2)(\lg y + \lg(k-1-y) + 1)$. Applying the concavity of logarithm function, we know that $[\lg a + \lg b]/2 \leq \lg[(a+b)/2]$. We find that

$$\max_y f(w/2, y) \leq 2\frac{w}{2}\left(\lg\frac{k-1}{2}\right) + \frac{w}{2} \leq w\left(\lg\frac{k-1}{2} + 1\right) \leq w \lg k,$$

which by algebraic simplification is equivalent to $\lg(k-1) \leq \lg k$.

We now consider the case that $\mathcal{T}(r_a)$ contains only one request. So $y = 1$. By induction $ON(\mathcal{T}(r_a)) \leq 2x$ and $ON(\mathcal{T}(r_b)) \leq 2(w-x)\lg(k-2)$. Hence, by substituting into (2) we get

$$ON(\mathcal{T}(r_i)) \leq 2x + 2(w-x)\lg(k-2) + 2x + d(r_i, s_i).$$

In order to show that $ON(\mathcal{T}(r_i)) \leq 2\lg k \cdot OPT(\mathcal{T}(r_i))$ it is sufficient to show

$$2x + 2(w-x)\lg(k-2) + 2x \leq 2w \lg k.$$

Since the left-hand side is linear in $x$, we need consider only $x = 0$ and $x = w/2$. The case $x = 0$ follows immediately. If $x = w/2$ we are left with showing $(w/2)\lg(k-2) + w \leq w \lg k$. This is equivalent to $\lg(2\sqrt{k-2}) \leq \lg k$, or $(2\sqrt{k-2}) \leq k$, which one can verify holds for $k \geq 3$.

We now consider the case that $\mathcal{T}(r_b)$ contains only one request. So $y = k-2$. By induction $ON(\mathcal{T}(r_a)) \leq 2x\lg(k-2)$ and $ON(\mathcal{T}(r_b)) \leq 2(w-x)$. Hence, by substituting into (2) we get

$$ON(\mathcal{T}(r_i)) \leq 2x\lg(k-2) + 2(w-x) + 2x + d(r_i, s_i).$$

In order to show that $ON(\mathcal{T}(r_i)) \leq 2\lg k \cdot OPT(\mathcal{T}(r_i))$ it is sufficient to show that

$$x\lg(k-2) + (w-x) + x \leq w \lg k.$$

Since the left-hand side is linear in $x$, one need only verify that the inequality holds at the boundaries $x = 0$ and $x = w/2$. ∎

### 3.1.1. The general case.

*Proof of Theorem* 2. When we apply Lemma 5 to each tree $\mathcal{T}^i$, we get the desired result. ∎

We now extend the result to the case that the online capacities are larger than two. Recall that $C = \sum_{i=1}^{m} c_i$ is the total online capacity.

THEOREM 6. *The double-competitive ratio of* GREEDY *for online transportation is* $O(\min(m, \lg C))$.

*Proof.* The upper bound of $O(\lg C)$ is immediate by Theorem 2 if we conceptually split a server site with $c_i$ online servers into $c_i/2$ sites with 2 arbitrary online servers and 1 arbitrary adversary server.

To see the $O(m)$ bound we need to be more careful about how we distribute the server sites among various trees. Assume that the tree construction procedure just constructed a tree $\mathcal{T}^k$. We perform some pruning of $\mathcal{T}^k$, if necessary, before we proceed to construct $\mathcal{T}^{k+1}$. If no root-to-leaf path in $\mathcal{T}^k$ passes through two server sites that are at the same location, then the number of vertices in $\mathcal{T}^k$ is $O(2^m)$. Hence, the $O(m)$ bound follows from Lemma 5.

If $\mathcal{T}^k$ contains root-to-leaf paths that pass through two server sites that are at the same location, we show how to modify $\mathcal{T}^k$ to remove such paths. Assume that

$\mathcal{T}^k$ contains a root-to-leaf path that first passes through $s_i$ and then passes through $s_j$, where $s_i$ and $s_j$ are at the same location. We modify $\mathcal{T}^k$ by making server vertex $s_j$ the child of $r_i$ in $\mathcal{T}^k$. See Figure 2. Note that may remove edges and vertices originally below $s_i$ from $\mathcal{T}^k$. We repeat this process until $\mathcal{T}^k$ has no root-to-leaf path passing through two server sites at the same location. Notice that the resulting tree $\mathcal{T}^k$ still satisfies the tree invariants. Now we start the construction of $\mathcal{T}^{k+1}$. ☐

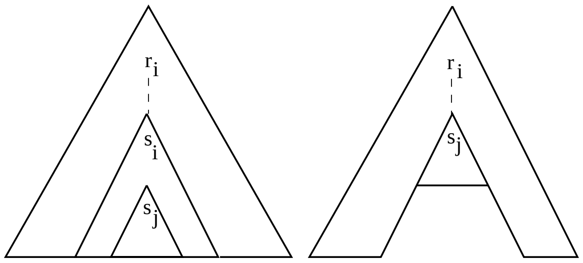


FIG. 2. *The original $\mathcal{T}^k$ on the left and the new $\mathcal{T}^k$ on the right.*

We now prove an asymptotically matching lower bound for the double-competitive ratio for GREEDY.

THEOREM 7. *The double-competitive ratio of* GREEDY *for the transportation problem is* $\Omega(\min(m, \lg C))$.

*Proof.* We embed $m$ server sites on the real line. The server site $s_1$ is located at the point $-1$. The server site $s_i$ $(2 \leq i \leq m)$ is located at $2^{i-1} - 1$. The online algorithm has $c_i = 2^{m-i+1}$ servers at $s_i$, while the adversary has $a_i = 2^{m-i}$ servers at $s_i$. Thus, the online algorithm has a total of $C = 2^{m+1} - 2$ servers, while the adversary has a total of $A = 2^m - 1$ servers. The requests occur in $m$ batches. The first batch consists of $2^{m-1}$ requests at the point 0. The $i$th batch $(2 \leq i \leq m)$ contains $2^{m-i}$ requests that occur at $2^{i-1} - 1$, the location of $s_i$. GREEDY responds to batch $i$ $(i < m)$ by answering each request in batch $i$ with a server at site $s_{i+1}$, thus depleting site $s_{i+1}$. GREEDY responds to the $m$th batch by moving one server from $s_1$. Thus, the total online cost is $m2^{m-1}$. By using the servers in $s_i$ to handle batch $i$ $(1 \leq i \leq m)$ it is possible to obtain a total cost of $2^{m-1}$. ☐

## 4. The algorithm BALANCE.

In this section we present an algorithm, BALANCE, with a double-competitive ratio of $O(1)$.

ALGORITHM BALANCE. At each site $s_h$ we classify half of the servers as *primary* and half of the servers as *secondary*. Let $c > 5 + 4\sqrt{2}$ be some constant. Define the *pseudodistance* from a request $r_i$ to a primary server at site $s_j$ to be $d(s_j, r_i)$ and the *pseudodistance* from $r_i$ to a secondary server at site $s_j$ to be $c \cdot d(s_j, r_i)$. BALANCE services each request $r_i$ with an arbitrary server with minimal pseudodistance from $r_i$.

Now our goal is to show that the double-competitive ratio of BALANCE for online transportation, with two online servers per site, is $O(1)$. First we break the response graph $\mathcal{G}$ into disjoint trees. Let $C_1, \ldots, C_l$ be the connected components of the response graph $\mathcal{G}$. By Lemma 1 each connected component of the response graph contains a unique cycle. Let $r_{\alpha(i)}$ be the latest request in the cycle in $C_i$. Let $\mathcal{T}^i$ be the tree that is $C_i$ minus the online edge incident to $r_{\alpha(i)}$. We set the root of $\mathcal{T}^i$ to be $r_{\alpha(i)}$. Each such tree $\mathcal{T}^i$ satisfies the following two tree invariants.

(1) Each nonleaf server site $s_j \in \mathcal{T}^i$, with one incident online edge in $\mathcal{T}^i$, had the secondary server available just before the time of each request $r_h \in \mathcal{T}^j$.

(2) Each leaf of $\mathcal{T}^i$ is a server site $s_j$ that had both of its servers available just before the time of each request in $\mathcal{T}^i$.

We now fix a particular tree, say $\mathcal{T} = \mathcal{T}^i$, and for simplicity drop the superscript $i$. In order to show that the double-competitive ratio of BALANCE is $O(1)$, it is sufficient to show that $ON(\mathcal{T}) = O(OPT(\mathcal{T}))$.

We first provide some intuition. The fact that nonleaf server sites in $\mathcal{T}$ may only have one incident online edge motivates us to partition the BALANCE edges into two groups, local edges and global edges. The cost of local edges are amortized against $OPT$ edges that are at most a constant number edges away in the tree (see Lemma 18 and Lemma 19). Thus any $OPT$ edge can be charged at most a constant number of times. As a consequence, the total cost of the local edges is $O(OPT)$. We then ignore the local edges and consider only global edges. We show that the cost of the global edges geometrically decreases level by level as we go up the tree (see Lemma 14 and Lemma 16 ). Hence, we conclude that the cost of the global edges is $O(OPT)$.

DEFINITION 8. *Let $s_i$ be a generic server site in $\mathcal{T}$. We say that the* primary server child *$s_a$ of $s_i$ is the server site that the adversary used to service the request serviced by $s_i^1$, and the* secondary server child *$s_b$ of $s_i$ is the server site that the adversary used to service the request serviced by $s_i^2$. We say that $s_a$ and $s_b$ are the* server children *of $s_i$. The* server parent *$s_{\sigma(i)}$ of $s_i$ is the server site used by* BALANCE *to service $r_i$. The site $s_i$ is a* double *if it has two server children, and otherwise $s_i$ is a* single.

LEMMA 9. *If* BALANCE *uses a server at site $s_{\sigma(i)}$ to handle a request $r_i \in \mathcal{T}$, then*

$$d(s_{\sigma(i)}, r_i) \le d(r_i, s_i) + ld(s_i)$$

*and*

$$ld(s_{\sigma(i)}) \le 2(d(r_i, s_i) + ld(s_i)). \tag{4}$$

*Proof.* Observe that there is a path in $\mathcal{T}$ from $r_i$ to some leaf $s_k$ with total length at most $d(r_i, s_i) + ld(s_i)$. If $s_{\sigma(i)} \neq s_k$, then $d(s_{\sigma(i)}, r_i) \le d(r_i, s_i) + ld(s_i)$ since BALANCE didn't use $s_k$ to service $r_i$. If $s_{\sigma(i)} = s_k$, then $d(s_{\sigma(i)}, r_i) \le d(r_i, s_i) + ld(s_i)$ by the triangle inequality. Now

$$ld(s_{\sigma(i)}) \le d(s_{\sigma(i)}, r_i) + d(s_i, r_i) + ld(s_i) \le 2(d(r_i, s_i) + ld(s_i)). \qquad \square$$

LEMMA 10. *Assume that* BALANCE *uses a secondary server $s^2_{\sigma(i)}$ to handle a request $r_i \in \mathcal{T}$ that is not the root of $\mathcal{T}$. Then*

$$d(s_{\sigma(i)}, r_i) \le (1/c)(d(r_i, s_i) + ld(s_i))$$

*and*

$$ld(s_{\sigma(i)}) \le (1 + 1/c)(d(r_i, s_i) + ld(s_i)).$$

*Proof.* Observe that there is a path in $\mathcal{T}$ from $r_i$ to some leaf $s_k$ with total length at most $d(r_i, s_i) + ld(s_i)$. Then $d(s_{\sigma(i)}, r_i) \le (1/c)(d(r_i, s_i) + ld(s_i))$ since BALANCE didn't use the primary server at $s_k$ to service $r_i$. Now

$$ld(s_{\sigma(i)}) \le d(s_{\sigma(i)}, r_i) + d(s_i, r_i) + ld(s_i) \le (1 + 1/c)(d(r_i, s_i) + ld(s_i)). \qquad \square$$

DEFINITION 11. *We let $\gamma = \frac{2c+2}{3c+1}$.*

OBSERVATION 12. *For our choice of $c > 5 + 4\sqrt{2}$, we have $\gamma \leq 2\gamma^2 < 1$.*

FACT 13. *For all nonnegative reals $x$ and $y$, and for all $c > 1$, $\min(2x, (1 + 1/c)y) \leq \gamma(x + y)$.*

*Proof.* Suppose $2x \geq (1+1/c)y$. It suffices to show that $\min(2x, (1+1/c)y) = (1+1/c)y \leq \gamma(\frac{1}{2}(1+1/c)y+y)$. Simple algebra shows that $(1+1/c)y = \gamma(\frac{1}{2}(1+1/c)y+y)$.

On the other hand, suppose $2x < (1+1/c)y$. It suffices to show that $\min(2x, (1+1/c)y) = 2x \leq \gamma(x + \frac{2c}{c+1}x)$. Simple algebra shows that $2x = \gamma(x + \frac{2c}{c+1}x)$. □

LEMMA 14. *Assume that $s_i$ is a double server site with server children $s_a$ and $s_b$. Then*

$$ld(s_i) \leq \gamma(d(r_a, s_a) + d(r_b, s_b) + ld(s_a) + ld(s_b)). \tag{5}$$

*Proof.* Note that $ld(s_i) \leq 2(d(r_a, s_a) + ld(s_a))$ by Lemma 9, and that $ld(s_i) \leq (1 + 1/c)(d(r_b, s_b) + ld(s_b))$ by Lemma 10. The lemma then follows by Fact 13. □

DEFINITION 15. *A server site $s_i$ is* local *if either $s_i$ is a single or one of $s_i$'s server children is a single. Otherwise, $s_i$ is* global. *For convenience, we call a request $r_i$* local *if $s_i$ is local. Otherwise $r_i$ is* global. *For a server site $s_i$ we use $z(s_{\sigma(i)})$ to denote $d(s_i, r_i) + d(s_a, r_a) + d(s_b, r_b)$.*

We now break the accounting of the online edges in $\mathcal{T}$ into cases. We first show that for every local server $s_i$, the cost for BALANCE to serve $r_i$ is $O(z(s_{\sigma(i)}))$.

LEMMA 16.

$$\sum_{\text{local } r_i} d(s_{\sigma(i)}, r_i) \leq (c^2 + 2c)OPT(\mathcal{T}).$$

*Further, for each parent server $s_{\sigma(i)}$ of a local server $s_i$,*

$$ld(s_{\sigma(i)}) \leq (c^2 + c)z(s_{\sigma(i)}) + ld(s_i). \tag{6}$$

*Proof.* We break the proof into two cases. In the first case assume that $s_i$ is a single. It must be the case that $d(s_{\sigma(i)}, r_i) \leq c \cdot d(r_i, s_i)$ since BALANCE didn't use a server at $s_i$ to handle $r_i$. Hence,

$$\sum_{\text{single local } s_i} d(s_{\sigma(i)}, r_i) \leq c \sum_{\text{single local } s_i} d(s_i, r_i) \leq c \cdot OPT(\mathcal{T}).$$

Further,

$$ld(s_{\sigma(i)}) \leq (c + 1)d(s_i, r_i) + ld(s_i).$$

In the second case assume that $s_i$ is double, and one of $s_i$'s server children is a single. For simplicity assume that $s_a$ is a single; one can verify that the following argument also holds if $s_b$ is a single. Since $s_a$ is a single we can apply the analysis from the previous case to get that $d(s_i, r_a) \leq c \cdot d(r_a, s_a)$. Since BALANCE serviced $r_i$ with a server at $s_{\sigma(i)}$ instead of the unused server at $s_a$ we get that

$$d(s_{\sigma(i)}, r_i) \leq c \cdot d(s_a, r_i) \leq c \left[d(s_i, r_i) + d(s_i, r_a) + d(s_a, r_a)\right].$$

Hence, by substitution,

$$d(s_{\sigma(i)}, r_i) \leq c \cdot d(r_i, s_i) + (c^2 + c)d(r_a, s_a). \tag{7}$$

Thus,

$$\begin{aligned}\sum_{\text{double local } s_i} d(s_{\sigma(i)}, r_i) &\leq c \sum_{\text{double local } s_i} d(r_i, s_i) \\ &\quad + (c^2 + c) \sum_{\text{double local } s_i} d(r_a, s_a) \\ &\leq (c^2 + 2c) OPT(\mathcal{T}).\end{aligned}$$

Furthermore,

$$\begin{aligned}ld(s_{\sigma(i)}) &\leq d(s_{\sigma(i)}, r_i) + d(r_i, s_i) + ld(s_i) \\ &\leq (c+1) d(r_i, s_i) + (c^2 + c) d(s_a, r_a) + ld(s_i).\end{aligned}$$

The result follows. $\square$

Now we consider the online cost of servicing the global $r_i$'s. Observe that by Lemma 9 it is the case that

$$\sum_{\text{global } r_i \in \mathcal{T}} d(s_{\sigma(i)}, r_i) \leq \sum_{\text{global } r_i \in \mathcal{T}} [d(s_i, r_i) + ld(s_i)] \leq OPT(\mathcal{T}) + \sum_{\text{global } s_i \in \mathcal{T}} ld(s_i).$$

We now will show that

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) = O(OPT(\mathcal{T})).$$

DEFINITION 17. *Let $s_i$ be a global server site in $\mathcal{T}$. We define $S(s_i)$ to be the set of global server sites in the subtree of $\mathcal{T}$ rooted at $s_i$ with the property that, for any server site $s_j \in S(s_i)$, there are no global server sites in the unique path from $s_i$ to $s_j$ in $\mathcal{T}$. We define $LC(s_i)$ to be the sum of the cost of the offline edges $(s_j, r_j)$ in the subtree rooted at $s_i$ with the property that the unique path from $s_i$ to $r_j$ in $\mathcal{T}$ does not pass through a global server site.*

To give an alternative explanation of $LC(s_i)$, consider pruning $\mathcal{T}$ at the global server vertices, which results in a collection of trees rooted at global server vertices. Then $LC(s_i)$ is the total cost of the offline edges in the tree rooted at $s_i$.

LEMMA 18. *For any global server site $s_i$,*

$$ld(s_i) \leq 2(c^2 + c) LC(s_i) + 2\gamma^2 \sum_{s_j \in S(s_i)} ld(s_j).$$

*Proof.* If the server children $s_a$ and $s_b$ of $s_i$ are both global, then

$$ld(s_i) \leq \gamma(d(r_a, s_a) + d(r_b, s_b)) + \gamma(ld(s_a) + ld(s_b)).$$

Hence, the result follows in this case since $\gamma \leq 2\gamma^2 < 1$.

Now assume that at least one server child of $s_i$, say, $s_a$, is local. Note that $s_a$ and $s_b$ must be a double, since $s_i$ is global. Further, assume for the moment that $s_b$ of $s_i$ is global. Let $s_c$ and $s_d$ be the two server children of $s_a$. Then using (5) we get that

$$\begin{aligned}ld(s_i) &\leq \gamma(d(r_a, s_a) + d(r_b, s_b)) + \gamma ld(s_b) \\ &\quad + \gamma^2(d(r_c, s_c) + d(r_d, s_d)) + \gamma^2(ld(s_c) + ld(s_d)).\end{aligned}$$

In this case we continue to expand $ld(s_c)$ and $ld(s_d)$ using the general expansion method described below. Now consider the case that both $s_a$ and $s_b$ are local. Let $s_e$ and $s_f$ be the server children of $s_b$. Then using (5) we get that

$$\begin{aligned} ld(s_i) &\le \gamma(d(r_a, s_a) + d(r_b, s_b)) \\ &\quad + \gamma^2(d(r_c, s_c) + d(r_d, s_d) + d(r_e, s_e) + d(r_f, s_f)) \\ &\quad + \gamma^2(ld(s_c) + ld(s_d) + ld(s_e) + ld(s_f). \end{aligned}$$

In this case we continue to expand $ld(s_c)$, $ld(s_d)$, $ld(s_e)$, and $ld(s_f)$ using the general expansion method described below.

We now describe the general expansion method. We expand each $ld(s_j)$ using (4) and (6). A term of the form $ld(s_j)$ in the right-hand side of a recurrence relation is replaced according to the following rules.

(1) If $s_j$ is a leaf in $\mathcal{T}$, then $ld(s_j)$ is replaced by 0.

(2) If $s_j$ has a local server child $s_k$, then $ld(s_j)$ is replaced by $(c^2+c)z(s_j)+ld(s_k)$, which is valid by (6). Observe that since $s_k$ is local it will be subsequently expanded.

(3) If none of $s_j$'s server children are local, then $ld(s_j)$ is replaced by $2(d(r_k, s_k)+ ld(s_k))$, where $s_k$ is an arbitrary server child of $s_j$. This is valid by (4). Note that in this case the term $ld(s_k)$ is not replaced again since $s_k \in S(s_i)$.

The $d(s_x, r_x)$ terms that appear in this general replacement process are all included in $LC(s_i)$. Hence, we get

$$ld(s_i) \le \alpha LC(s_i) + \beta \sum_{s_j \in S(s_i)} ld(s_j)$$

for some $\alpha$ and $\beta$. Since each offline edge appears in this general replacement process at most twice, and in each case the coefficient is at most $(c^2 + c)$, we can conclude that $\alpha \le 2(c^2 + c)$. Note that in rule (2), the coefficient in front of the $ld(s_j)$ term before the replacement process is the same as the coefficient in front of the $ld(s_k)$ term after the replacement process. Since the coefficient in front of each $ld$ term is $\gamma^2$ before the application of any of the general replacement rules, and the only way that it can change is by application of rule (3), which is a terminal replacement rule, each coefficient on an $ld(s_j)$ term when the general replacement process terminates is at most $2\gamma^2$. The result follows. ∎

LEMMA 19. *For the choice of $c > 5 + 4\sqrt{2}$ and consequently $2\gamma^2 < 1$,*

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) \le \frac{2(c^2 + c)}{1 - 2\gamma^2} OPT(\mathcal{T}).$$

*Proof.* First observe that if $s_i$ and $s_j$ are two different global server sites, then there is no common offline cost in the sums $LC(s_i)$ and $LC(s_j)$. As a consequence we get

$$\sum_{\text{global } s_i \in \mathcal{T}} LC(s_i) \le OPT(\mathcal{T}).$$

Therefore, it suffices to show that

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) \le \frac{2(c^2 + c)}{1 - 2\gamma^2} \sum_{\text{global } s_i \in \mathcal{T}} LC(s_i).$$

Applying Lemma 18 we have that

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) \leq 2(c^2+c) \sum_{\text{global } s_i \in \mathcal{T}} LC(s_i) + \sum_{\text{global } s_i \in \mathcal{T}} 2\gamma^2 \sum_{s_j \in S(s_i)} ld(s_j).$$

Since $S(s_i) \cap S(s_j) = \emptyset$ for $i \neq j$, we get that

$$\sum_{\text{global } s_i \in \mathcal{T}} 2\gamma^2 \sum_{s_j \in S(s_i)} ld(s_j) \leq 2\gamma^2 \sum_{\text{global } s_i \in \mathcal{T}} ld(s_i).$$

Now substituting back, we get

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) \leq \frac{2(c^2+c)}{1-2\gamma^2} \sum_{\text{global } s_i \in \mathcal{T}} LC(s_i).$$

The result follows. ◻

THEOREM 20. *The double-competitive ratio of* BALANCE*, with two online servers per site, in the online transportation problem is*

$$(c+1)^2 + \frac{2c(c+1)}{1-2\gamma^2}.$$

*Proof.* The total online cost is $(c^2+2c+1)OPT(\mathcal{T})$ from Lemma 16, plus $OPT(\mathcal{T})$, plus $\frac{2(c^2+c)}{1-2\gamma^2}OPT(\mathcal{T})$ from Lemma 19. ◻

We now assume an arbitrary number of online servers per site.

THEOREM 21. *The double-competitive ratio of* BALANCE *for the online transportation problem is*

$$(c+1)^2 + \frac{2c(c+1)}{1-2\gamma^2}.$$

*Proof.* This follows immediately by conceptually splitting each server site $s_i$ into $c_i/2$ sites. ◻

**5. Conclusion.** We believe that there are two main contributions of this paper. The first is that we give some evidence that greedy-like algorithms for the online transportation problem may perform reasonably well in many circumstances. The second is that these results support the hypothesis that resource augmentation analysis is a useful algorithmic analysis technique. A survey of the many applications of resource augmentation subsequent to this research can be found in [7].

The most obvious avenue for further investigation is to determine the competitive ratio in the resource augmentation model when the adversary's capacity is more than half of the online capacity. It seems that some new techniques will be needed in this case since the response graph no longer has the tree-like property from Lemma 1 that was so critical in our proofs.

**Acknowledgments.** We thank the anonymous referees for many helpful comments.

## REFERENCES

[1] A. BORODIN AND R. EL-YANIV, *Online Computation and Competitive Analysis*, Cambridge University Press, Cambridge, UK, 1998.

[2] A. Fiat and G. Woeginger, *Competitive odds and ends*, in On-Line Algorithms: The State of the Art, A. Fiat and G. Woeginger, eds., Lecture Notes in Comput. Sci. 1442, Springer-Verlag, New York, 1998, pp. 385–394.

[3] B. Kalyanasundaram and K. Pruhs, *Online weighted matching*, J. Algorithms, 14 (1993), pp. 478–488.

[4] J. Kennington and R. Helgason, *Algorithms for Network Programming*, John Wiley, New York, 1980.

[5] S. Khuller, S. Mitchell, and V. Vazirani, *On-line algorithms for weighted matchings and stable marriages*, Theoret. Comput. Sci., 127 (1994), pp. 255–267.

[6] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, New York, Montreal, London, 1976.

[7] K. Pruhs, *A Survey of Resource Augmentation Analysis*, in preparation.

[8] D. Sleator and R. Tarjan, *Amortized efficiency of list update and paging rules*, Comm. ACM, 28 (1985), pp. 202–208.

[9] Y. T. Tsai, C. Y. Tang, and Y. Y. Chen, *Average performance of a greedy algorithm for the on-line minimum matching problem on Euclidean space*, Inform. Process. Lett., 51 (1994), pp. 275–282.

[10] N. Young, *The k-server dual and loose competitiveness for paging*, Algorithmica, 11 (1994), pp. 525–541.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 3, pp. 384–402

# BIN PACKING WITH DISCRETE ITEM SIZES, PART I: PERFECT PACKING THEOREMS AND THE AVERAGE CASE BEHAVIOR OF OPTIMAL PACKINGS[*]

E. G. COFFMAN, JR.[†], C. COURCOUBETIS[‡], M. R. GAREY[§], D. S. JOHNSON[¶], P. W. SHOR[¶], R. R. WEBER[‖], AND M. YANNAKAKIS[§]

**Abstract.** We consider the one-dimensional bin packing problem with unit-capacity bins and item sizes chosen according to the discrete uniform distribution $U\{j,k\}$, $1 < j \le k$, where each item size in $\{1/k, 2/k, \dots, j/k\}$ has probability $1/j$ of being chosen. Note that for fixed $j,k$ as $m \to \infty$ the discrete distributions $U\{mj, mk\}$ approach the continuous distribution $U(0, j/k]$, where the item sizes are chosen uniformly from the interval $(0, j/k]$. We show that average-case behavior can differ substantially between the two types of distributions. In particular, for all $j,k$ with $j < k-1$, there exist on-line algorithms that have constant expected wasted space under $U\{j,k\}$, whereas no on-line algorithm has even $o(n^{1/2})$ expected waste under $U(0,u]$ for any $0 < u \le 1$. Our $U\{j,k\}$ result is an application of a general theorem of Courcoubetis and Weber [C. Courcoubetis and R.R. Weber, *Probab. Engrg. Inform. Sci.*, 4 (1990), pp. 447–460] that covers all discrete distributions. Under each such distribution, the optimal expected waste for a random list of $n$ items must be either $\Theta(n)$, $\Theta(n^{1/2})$, or $O(1)$, depending on whether certain "perfect" packings exist. The perfect packing theorem needed for the $U\{j,k\}$ distributions is an intriguing result of independent combinatorial interest, and its proof is a cornerstone of the paper.

**Key words.** bin packing, on-line, average-case analysis, approximation algorithms

**AMS subject classifications.** 68W25, 68W40

**PII.** S0895480197325936

## 1. Introduction.

Suppose one is given items of sizes $1, 2, 3, \dots, j$, one of each size, and is asked to pack them into bins of capacity $k$ with as little wasted space as possible, i.e., one is asked to find a least cardinality partition (packing) of the set of items such that the sizes of the items in each block (bin) sum to at most $k$. For what values of $j$ and $k$ can the set be packed perfectly (i.e., so that the sizes of the items in each block sum to exactly $k$)? Clearly the sum of the item sizes must be divisible by $k$, but what other conditions must be satisfied? Surprisingly, the divisibility constraint is not only necessary but sufficient. Readers might want to try their hand at proving this. Relatively short proofs exist, as illustrated in the next section, but a certain ingenuity is required to find one. The exercise serves as a warm-up for the following more general and more difficult theorem, also proved in the next section, in which there are $r$ copies of each size, for some $r \ge 1$.

THEOREM 1 (perfect packing theorem). *For positive integers $k$, $j$, and $r$, with $k \ge j$, one can perfectly pack a list $L$ consisting of $rj$ items, $r$ each of sizes $1$ through $j$, into bins of size $k$ if and only if the sum of the $rj$ item sizes is a multiple of $k$.*

[*]Received by the editors August 11, 1997; accepted for publication (in revised form) October 14, 1999; published electronically May 26, 2000.
http://www.siam.org/journals/sidma/13-3/32593.html

[†]New Jersey Institute of Technology, Newark, NJ 07102 (coffman@homer.njit.edu). The research of this author was completed while at Bell Labs.

[‡]Department of Computer Science, Athens University of Economics and Business, Athens, Greece (courcou@csi.forth.gr).

[§]Bell Labs, Murray Hill, NJ 07974 (mihalis@research.bell-labs.com). M.R. Garey is now retired.

[¶]AT&T Labs–Research, Florham Park, NJ 07932 (dsj@research.att.com, shor@research.att.com).

[‖]Statistical Laboratory, University of Cambridge, Cambridge CB2 1SB, UK (R.R.Weber@statslab.cam.ac.uk).

In set-theoretic terms, the question answered by Theorem 1 is an intriguing puzzle in pure combinatorics. But our motivation to work on it came from its relevance to certain fundamental questions about the average-case analysis of algorithms. In particular, consider the standard bin packing problem, in which one is given a list of items $L = (a_1, a_2, \ldots, a_n)$, where each $a_i$ has a positive size $s_i \leq 1$, and is asked to find a packing of these items into a minimum number of unit-capacity bins. Clearly, Theorem 1 can be cast in these terms by a simple rescaling.

In most real-world applications of bin packing, as in Theorem 1, the item sizes are drawn from some finite set. However, the usual average-case analysis of bin packing heuristics has assumed that item sizes are chosen according to continuous probability distributions, which by their nature allow an uncountable number of possible item sizes (see [1, 8], for example). The assumption of a continuous distribution has the advantage of sometimes simplifying the analysis, and has been justified on the grounds that continuous distributions should serve as reasonable approximations for discrete ones. But there are reasons to ask whether this is actually true. For example, let $U(0,u]$ denote the continuous uniform distribution over the interval $(0,u]$ and let $U\{j,k\}$ denote the discrete uniform distribution on the set $\{1/k, \ldots, j/k\}$. Then the limit of the distributions $U\{mj, mk\}$, as $m \to \infty$ is $U(0, j/k]$, but in the limit combinatorial questions such as those addressed by Theorem 1 evaporate. This suggests that something important (and interesting) may in fact be lost in the transition from discrete to continuous models. The results in this paper illustrate that, indeed, fundamental aspects of the average-case behavior of classical bin packing algorithms are obscured by the continuous models.

To describe these results, we need the following notation. If $A$ is an algorithm and $L$ is a list of items, then $A(L)$ is the number of bins used when $A$ is applied to $L$, and $s(L)$ is the sum of the item sizes in $L$. The waste in the packing of $L$ by $A$ is denoted by $W^A(L) = A(L) - s(L)$. Note that this equals the sum of the gaps in the bins used by $A$'s packing of $L$, where the gap in a bin containing items of total size $\sigma$ is $1 - \sigma$. Let OPT$(L)$ denote the length of an optimal packing; OPT also denotes a corresponding optimal off-line packing algorithm. In what follows, $L_n$ is a list of $n$ items whose sizes are independent samples from a given distribution. To avoid trivialities, we disallow the item size $k$, and hence ignore the distribution $U\{k,k\}$. Further, we will ignore the (deterministic) distribution $U\{1,k\}$. Thus, only the distributions $U\{j,k\},\ 1 < j < k$, will be of interest.

This paper is the first in a series of articles covering the results announced in [2, 6], all dealing with classical bin packing under discrete distributions and showing qualitative differences between algorithmic behavior under such discrete distributions and their continuous analogues. The specific problems addressed by the series are summarized in Table 1, and are shown along with the continuous analogues for contrast. The present paper, Part I of the series, proves the results marked by a bullet (•) in the table. In this table $\Omega(f(n))$ has the Knuthian sense of "greater than $cf(n)$ for some $c > 0$ and all sufficiently large $n$" and $\underline{\Omega}(f(n))$ has the weaker sense of Hardy and Littlewood's "not $o(f(n))$," i.e. "greater than $cf(n)$ for some $c > 0$ and infinitely many $n$." We say $g(n)$ is $\Theta(f(n))$ (respectively, $\underline{\Theta}(f(n))$) when $g(n)$ is both $O(f(n))$ and $\Omega(f(n))$ (respectively, $\underline{\Omega}(f(n))$). A subscript of $u$ or $k$ indicates that the hidden multiplicative constants depend on $u$ or $k$ and so we have asymptotics in $n$ that hold for fixed $u$ or $k$. Where more than one growth rate is possible, we list all that have been observed for distributions of the stated type, using "..." to indicate when other possibilities have not been ruled out. In the case of general discrete distributions it is NP-hard to determine which possibilities apply under first fit decreasing (FFD),

TABLE 1
*Results for expected waste.*

| | $U(0,1]$ | Ref | $U\{j,k\}$, $j=k-1,k$ | Ref |
|---|---|---|---|---|
| OPT | $\Theta(n^{1/2})$ | [17, 16] | $\Theta(n^{1/2})$ | [8] |
| FFD, BFD | $\Theta(n^{1/2})$ | [17, 16] | $\Theta(n^{1/2})$ | [8] |
| FF | $\Theta(n^{2/3})$ | [7] | $\Theta(n^{1/2}k^{1/2})$, $k=O(n^{1/3})$ | [7] |
| | | | $\Theta(n^{2/3})$, $k=\Omega(n^{1/3})$ | [7] |
| BF | $\Theta(n^{1/2}\log^{3/4} n)$ | [19] | $\Theta(n^{1/2}\log^{3/4} k)$, $k=O(n)$ | [5] |
| | | | $\Theta(n^{1/2}\log^{3/4} n)$, $k=\Omega(n)$ | [5] |
| Best on-line | $\underline{\Theta}(n^{1/2}\log^{1/2} n)$ | [19, 20] | $\underline{\Omega}_k(n^{1/2})^\dagger$ | [–] |

| | $U(0,u]$, $u<1$ | Ref | $U\{j,k\}$, $1\le j\le k-2$ | Ref |
|---|---|---|---|---|
| OPT | $\Theta_u(1)$ | [1] | $\Theta_k(1)$ | [•] |
| FFD | $\Theta_u(1)$, $u\le 1/2$ | [1, 13] | $\Theta_k(1)$, $\Theta_k(n^{1/2})^\ddagger$, $\Theta_k(n)^\S$ | [3] |
| | $\Theta_u(n^{1/3})$, $u>1/2$ | [1] | | |
| FF | $\Theta_u(n)^*$ | [2] | $\Theta_k(1)$, $\Theta_k(n^{1/2})^\ddagger$, $\Theta_k(n)^*$ | [6, 4] |
| BF | $\Theta_u(n)^*$ | [2] | $\Theta_k(1)$, $\Theta_k(n^{1/2})^\ddagger$, $\Theta_k(n)$ | [6, 4, 15] |
| Best on-line | $\underline{\Omega}_u(n^{1/2})$ | [•] | $\Theta_k(1)$ | [•] |

| | General continuous | Ref | General discrete | Ref |
|---|---|---|---|---|
| OPT | $\Theta_u(1)$, $\Theta_u(n^{1/2})$, $\Theta_u(n),\ldots$ | [1, 17, 16] | $O_k(1)$, $\Theta_k(n^{1/2})$, $\Theta_k(n)$ | [9] |
| FFD | $\Theta_u(1)$, $\Theta_u(n^{1/3})$, $\Theta_u(n^{1/2})$, $\Theta_u(n),\ldots$ | [1, 17, 16] | $O_k(1)$, $\Theta_k(n^{1/2})$, $\Theta_k(n)^\S$ | [3] |
| Best on-line | $\underline{\Theta}(n^{1/2}\log^{1/2} n)$, $\Theta_u(n),\ldots$ | [19, 20] | $O_k(1)$, $\Theta_k(n^{1/2})$, $\Theta_k(n)$ | [9] |

§ These results also hold for BFD.

† Easily shown to be $\underline{\Omega}(n^{1/2})$; $O(n^{1/2}\log^{3/4} k)$ by the results for Best Fit; conjectured to be $\underline{\Theta}(n^{1/2}\log^{1/2} k)$, $k=O(n)$ and $\underline{\Theta}(n^{1/2}\log^{1/2} n)$, $k=\Omega(n)$.

‡ Not ruled out by theorems, but no occurrences known either. For FFD and $j\le k-2$, $\Theta_k(n^{1/2})$ is known not to occur for any $k\le 10{,}000$ [3].

∗ Conjecture supported by simulation studies.

best fit decreasing (BFD), and the best on-line algorithm, although exponential-time decision algorithms exist [9, 4].

Our primary focus in this paper is on the distributions $U\{j,k\}$ with $1<j\le k-2$. The marked results for these distributions are both contained in the following theorem, which is proved in section 3.

THEOREM 2. *For any $j,k$, $1<j\le k-2$, there exists an on-line algorithm $A$ with running time bounded by a polynomial in $n$ and $k$ such that if $L_n$ has item sizes generated according to $U\{j,k\}$ then $E[W^A(L_n)]=\Theta(1)$.*

The corresponding result in the continuous model, which is proved in section 5, shows substantially worse behavior.

THEOREM 3. *If $L_n$ has item sizes generated according to $U(0,u]$ for $0<u<1$, and $A$ is any on-line algorithm, then there exists a constant $c>0$ such that $E[W^A(L_n)]>cn^{1/2}$ for infinitely many $n$.*

We should note that the comparison between discrete and continuous cases is much closer in the case of $U\{k-1,k\}$, which corresponds to $U(0,1]$. In fact, for unconstrained (off-line) optimal packings, one obtains the same $\Theta(n^{1/2})$ expected waste results in both cases. The proof of this fact in the discrete case can be assembled from standard techniques in the continuous theory, for example by observing that the algorithm MATCH and its analysis in [8, p. 100] carry over to the discrete case. Note that this bound is independent of $k$. There does appear to be a dependence on $k$

in the on-line case, but one can obtain $O_k(n^{1/2})$ behavior in a quite straightforward manner. Simply keep a pool of $\lceil k/2 \rceil$ bin types, one each for each matching pair of item sizes $(i, k-i)$, $1 \le i \le k/2$ and one for item size $k/2$ if $k$ is even. When an item of size $i$ arrives, we look to see if there is a partially full bin of its type containing an item of size $k-i$ and if so, add the new item to the bin, filling it. Otherwise a new bin of the given type is started. It is easy to show that this procedure has $O_k(n^{1/2})$ expected waste.

The plan for the remaining papers in this series is as follows. Part II of the series [7] proves the first fit (FF) results under $U\{k-1, k\}$, and in so doing, the $O(n^{2/3})$ upper bound for $U(0,1]$. Part III [3] will cover the results on FFD and BFD under discrete distributions (both uniform and general). Parts IV [4] and V [5] will present the results for best fit (BF) under $U\{j,k\}$, $1 < j \le k-2$ and $U\{k-1,k\}$, respectively (see also [15] which generalizes the results of [6, 4] for the case $U\{k-2,k\}$).

The current paper is organized as follows. The proof of the perfect packing theorem (Theorem 1) appears in section 2. In addition to being of independent combinatorial interest, this result contributes to the proof of Theorem 2, which we present in sections 3 and 4. In section 3, we describe a general result of Courcoubetis and Weber [9] which implies that for any discrete item-size distribution, $E[W^{\mathrm{OPT}}(L_n)]$ must be one of $\Theta(n)$, $\Theta(n^{1/2})$, or $O(1)$, and no matter which case applies, there is an on-line algorithm $A$ whose expected waste obeys the same bounds. Which case applies depends on the existence of certain perfect packings, and here is where Theorem 1 comes in. The algorithms provided by [9] for $U\{j,k\}$, $1 < j \le k-2$, are randomized algorithms, and in section 4 we present corresponding deterministic algorithms and give the somewhat more difficult proof that they also have $O(1)$ expected waste. Theorem 3, this paper's contribution to the theory of continuous distributions is proved in section 5.

We conclude in section 6 with a brief discussion of the significance of Theorem 2 and possible extensions to it. As stated, the theorem provides algorithms that are tailored to the particular distribution $U\{j,k\}$ in question. We can, however, provide a single algorithm $A$ with running time bounded by a polynomial in $n$ and $k$ that yields $E[W^A(L_n)] = \Theta(1)$ for any such distribution, even if $j$ and $k$ are not known in advance and have to be discovered on-line. We also discuss the possibility of extending Theorem 2 to more general classes of discrete distributions and mention some more recent developments. In particular, in [10, 11] a simple new deterministic on-line algorithm is introduced that supersedes the algorithms presented in this paper, in that it has $O(1)$ expected waste for *every* discrete distribution that has $E[W^{\mathrm{OPT}}(L_n)] = O(1)$, and has a running time $O(nk)$ for all $U\{j,k\}$ distributions.

**2. The perfect packing theorem.** We begin our proof of Theorem 1 with three lemmas that list a number of special instances that lead to perfect packing. The first lemma takes care of the special case, $r = 1$.

LEMMA 4. *Suppose $m$, $j$, and $k$ are positive integers such that $j \le k$ and $mk = j(j+1)/2$. Then the set of $j$ items, one each of sizes* $1, \dots, j$, *perfectly packs into $m$ bins of size $k$.*

*Proof.* The proof is by induction. Pick $j$ and $k$ and assume the theorem is true for all pairs that are smaller in lexicographic order than $(k, j)$. The theorem is clearly true for $k \le 2$ or $j \le 2$, so assume $k, j > 2$.

If $j > k/2$ then we can start by perfectly packing bins with pairs of items $(j - i, k - j + i)$, $0 \le i < j - k/2$, after which the remaining items are those of sizes $1, \dots, k-j-1$, plus the item of size $k/2$ if $k$ is even. Since the sum of the sizes of the items that have been packed at this point is a multiple of $k$, the sum of the sizes of remaining items is also a multiple of $k$. If $k$ is odd, the unpacked items are an instance of $(k, k-j-1)$, with $k-j-1 < j$ and the induction hypothesis applies. If $k$ is even then $k/2$ divides $j(j+1)/2$ and all remaining items are no larger than $k/2$. Thus the items $1, \dots, k-j-1$ form an instance of $(k/2, k-j-1)$ and by the induction hypothesis can be perfectly packed into bins of size $k/2$. These half-bins and the item of size $k/2$ can then be combined into bins of size $k$.

Now suppose $j \leq k/2$. If $k$ is even then we have an instance of $(k/2, j)$ and the induction hypothesis applies. If $k$ is odd, first note that $k/2 \geq j$ and $j > 2$ implies $k > j+1$, which together with $mk = j(j+1)/2$ implies $j > 2m$. Thus we can construct $m$ pairs of items each of total size $k' = 2j - 2m + 1$ by combining $j - i$ with $k' - j + i$, $0 \leq i \leq m-1$. If we place one pair in each of our $m$ bins, we now have $m$ bins with gaps of size $k - k' = k - 2j + 2m - 1$ and items of sizes $1, \dots, j - 2m$. Because $mk = j(j+1)/2$, the sum of these item sizes must be $m(k - k')$, and so an application of the induction hypothesis to the instance $(k - k', j - 2m)$ completes the proof. ∎

LEMMA 5. *Consider $r > 1$ sets, the $i$th of which consists of $j$ items of consecutive sizes, $\ell_i + 1, \dots, \ell_i + j$, for some $\ell_i \geq 0$. Suppose either* (a) *$r$ is even or* (b) *$j$ is odd. Then these $rj$ items perfectly pack into $j$ bins of size equal to the sum of the average item sizes in the $r$ groups, i.e., $r(j+1)/2 + \sum_{i=1}^{r} \ell_i$.*

*Proof.* The lemma will follow if we can show that for $\ell_i = 0$, $1 \leq i \leq r$, and bins of size $r(j+1)/2$ it is possible to pack perfectly the items into the $j$ bins in such a manner that each bin contains exactly one item from each of the $r$ sets.

If $r$ is even then we simply take two of the sets and pack the $i$th largest item in one set with the $i$th smallest item in the other set, i.e., as the pair $(i, j-i+1)$, $i = 1, \dots, j$. This fills $j$ bins to level $j+1$. By repeating this $r/2$ times we fill $j$ bins of size $r(j+1)/2$.

If $r$ and $j$ are both odd then an extra step is required. The idea is first to pack items in triples, one item from each of three sets, such that the sum of each triple is the same. It is easiest to appreciate the construction by considering an example, say $j = 9$. The triples, which each sum to 15, are given in the columns below.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 8 | 6 | 4 | 2 | 9 | 7 | 5 | 3 | 1 |

In general, the triples are $(i, i + (j+1)/2, j + 1 - 2i)$, $i = 1, \dots, (j-1)/2$, and $(i, i - (j-1)/2, 2j - 2i + 1)$, $i = (j+1)/2, \dots, j$. The result of packing these one per bin is to fill all $j$ bins to level $3(j+1)/2$. The number of remaining sets is even and the remaining spaces in the $j$ bins are equal. Thus, the procedure for case (a) can be applied to complete the packing in each bin. ∎

The following lemma provides part of the induction step used in the proof of Theorem 1.

LEMMA 6. *Consider a quadruple $(k, j, r, m)$ of positive integers such that $k \geq j$ and $mk = rj(j+1)/2$. Then there exists a perfect packing of $r$ copies of $1, \dots, j$ into $m$ bins of size $k$ if there exists a perfect packing for each lexicographically smaller quadruple of this form, and if any one of the following holds:*

(a) *$j \geq k/2$.*

(b) *$r$ does not divide $k$.*

(c) *$k$ or $r$ is even.*

(d) $j \leq (r-1)k/2r$.

*Proof.* First, using the arguments of Lemma 4, we demonstrate how to reduce the problem to a smaller instance if (a) holds. If $j \geq k/2$ and $k$ is odd, then we can pack bins with pairs $(j-i, k-j+i)$, $i = 0, \ldots, j-(k+1)/2$. The remaining items, which are of sizes $1, \ldots, k-j-1$, define the smaller instance $(k, k-j-1, r, m')$, where $m' = m - r(2j+1-k)/2$. If $j > k/2$ and $k$ is even, then we can pack bins in the same way, $i = 0, \ldots, j-1-k/2$. The remaining items, which are of sizes $1, \ldots, k-j-1$ and $k/2$, can be packed into bins of size $k/2$, by the induction hypothesis that there exists a perfect packing for $(k/2, k-j-1, r, m')$, where $m' = 2m - r(2j-k)$.

If (b) holds then $r$ and $m$ must have a common factor $p > 1$ and the problem reduces to the instance $(k, j, r/p, m/p)$.

Now suppose neither (a) nor (b) holds but (c) does. If $k$ is even then $k/2$ divides $rj(j+1)/2$. Since (a) doesn't hold, $j < k/2$. Thus the problem reduces to a smaller instance in which the bin size is $k/2$. If $r$ is even, then since (b) does not hold, $k$ is divisible by $r$ and so must be even too. Thus the same argument applies.

Finally, for case (d), assume that (a), (b), and (c) do not hold, i.e., $j < k/2$, $r$ divides $k$ and $k$ and $r$ are both odd. Let $r_1 = (r+1)/2$, $k_1 = kr_1/r$ and $r_2 = (r-1)/2$, $k_2 = kr_2/r$. Note that $r_1 + r_2 = r$ and $k_1 + k_2 = k$. The fact that $r$ is odd implies that $r_1$ and $r_2$ are integers. The fact that $r$ divides $k$ implies that $k_1$ and $k_2$ are integers, with $mk_1 = r_1 j(j+1)/2$ and $mk_2 = r_2 j(j+1)/2$. Since by assumption $j < k/2$, we have $k_1 \geq j$, and hence by hypothesis for the instance $(k_1, j, r_1, m)$, we can pack $r_1$ copies of $1, \ldots, j$ into $m$ bins of size $k_1$. Similarly, if also $j \leq k_2$ then we can pack $r_2$ copies of $1, \ldots, j$ into $m$ bins of size $k_2$. Since $k_1 + k_2 = k$ we can combine pairs of bins of sizes $k_1$ and $k_2$ into bins of size $k$. Thus there is a reduction to smaller instances if $j \leq k_2 = (r-1)k/2r$, i.e., if (d) holds. $\square$

*Proof of the perfect packing theorem.* Instances for which the theorem is to be proved are described by the quadruples of Lemma 6. Notice that it would be enough to specify the triple $(k, j, r)$; however, it is helpful to mention $m$ explicitly. The proof of the theorem is by induction on $(k, j, r)$ under lexicographical ordering. By Lemma 4 it is true for $r = 1$. Assume all quadruples that are smaller than $(k, j, r, m)$ can be perfectly packed and $r > 1$. We show there exists a perfect packing of $r$ copies of $1, \ldots, j$ into $m$ bins of size $k$. By Lemma 6, we need only consider the case when $k$ and $r$ are odd, $r$ divides $k$ and $(r-1)k/2r < j < k/2$. Note that in this case $(r-1)k/2r$ is an integer and $k/2r$ is 0.5 more than an integer. We show below that we can perfectly pack all the items of sizes from $j + 1 - (r-1)k/2r$ through $j$ into bins of size $k$. (Note that the lower bound on this range is greater than 1 because of the above lower bound on $j$.) The theorem then follows because the remaining items form a smaller quadruple, so by the induction hypothesis they can be perfectly packed into bins of size $k$.

To follow the construction below, the reader may find it helpful to consider a specific example. Consider the quadruple $(k, j, r, m) = (165, 77, 5, 91)$. Note that $k$ and $r$ are odd, $r$ divides $k$, and $j$ lies between $(r-1)k/2r = 66$ and $k/2 = 82.5$. We show below how to perfectly pack all items of sizes $12, \ldots, 77$. The remaining items form the smaller quadruple $(165, 11, 5, 2)$.

To pack all items of sizes from $j+1-(r-1)k/2r$ through $j$, we divide the range of item sizes into intervals, i.e., sets of consecutive integers. Each interval is symmetric about a multiple of $k/2r$ and has one of two lengths depending on whether the interval is symmetric about an odd or even multiple of $k/2r$. To form the intervals, we first take the largest interval that is symmetric about $(r-1)k/2r$; this is the interval

$[(r-1)k/r-j, j]$. Note that this interval does not include $(r-2)k/2r$ since $j < k/2 = rk/2r$. Next we take the largest interval that can be formed from the remaining items that is symmetric about $(r-2)k/2r$, obtaining the interval $[j-k/r+1, (r-1)k/r-j-1]$. Continuing in this fashion and taking intervals symmetric about further multiples of $k/2r$, we end up with intervals of two kinds. First, there are $(r-1)/2$ intervals centered on even multiples of $k/2r$, with the interval centered on $(r-1-2i)k/2r$ being $[(r-1-i)k/r-j, j-ik/r]$, where $i$ ranges from 0 to $(r-3)/2$. Second, there are an equal number of interval centered on odd multiples of $k/2r$, with the interval centered on $(r-2i)k/2r$ being $[j-ik/r+1, (r-i)k/r-j-1]$, where $i$ ranges from 1 to $(r-1)/2$. Note that the smallest endpoint is $j-ik/r+1$ for $i=(r-1)/2$, which equals $j+1-(r-1)k/2r$ as claimed above.

For the numerical example above, there are two intervals of each type. Intervals of the first type are $[22, 44]$ and $[55, 77]$; they are of length 23 and symmetric about 33 and 66. Intervals of the second type are $[12, 21]$ and $[45, 54]$; they are of length 10 and symmetric about 16.5 and 49.5. In general, intervals of the first type have odd length $2j-(r-1)k/r+1$ and are symmetric about an even multiple of $k/2r$. Intervals of the second type have even length $k-2j-1$ and are symmetric about an odd multiple of $k/2r$. Our plan is to use Lemma 5 to perfectly pack into bins of size $k$ those items whose sizes lie in intervals of the same type.

We begin by considering all those intervals of the first type. These have odd lengths and they are symmetric about points $ik/r$, $i = 1, \ldots, (r-1)/2$. There are $r$ items of each size in each of these intervals. Our strategy is to partition these intervals into groups that satisfy the hypotheses of Lemma 5 (b). That is, we arrange for the midpoints of the intervals within each group to sum to $k$. Since the midpoints correspond to the average item sizes for the corresponding intervals, and the number of items in the intervals is odd, Lemma 5 (b) implies that we can perfectly pack the items in the intervals of each group. Constructing these groups is a bin packing problem in which the midpoints of the intervals take on the role of item sizes. In what follows we write 'items' in single quotes when speaking of the midpoints of intervals, possibly normalized, and viewing them as items to be perfectly packed in bins of some required size. In considering intervals of the first type, it is as though we had $r$ 'items' of each of the sizes $ik/r$, $i = 1, \ldots, (r-1)/2$, and wished to pack them into bins of size $k$. After a normalization that multiplies each item size by $r/k$, this is equivalent to the problem of packing $r$ 'items' of each of the sizes $1, \ldots, (r-1)/2$ into bins of size $r$. That is, we have a smaller version $(k', j', r', m')$ of our packing problem, with $j' = (r-1)/2$, $k' = r' = r$ and $m' = r'j'(j'+1)/2k'$. But by the induction hypothesis this means that the desired packing can be achieved. In the example, it is as though we had 5 'items' of sizes 33 and 66 that are to be packed in bins of size 165. Normalizing by a factor of 1/33, this is equivalent to the problem instance $(5, 2, 5, 3)$.

We must now pack items whose sizes lie in intervals of the second type. These intervals are of even length, symmetric about the points $ik/2r$, for $i$ odd and $i = 1, \ldots, r-2$. Again, there are $r$ items of each size in these intervals. As above, we exhibit a reduction to a smaller perfect packing problem. After multiplying item sizes by $2r/k$ the problem is equivalent to perfectly packing $r$ copies of 'items' of sizes $1, 3, 5, \ldots, r-2$ into bins of size $2r$. For the example, this is 5 copies of 'items' of sizes 1 and 3, to be perfectly packed into 2 bins of size 10. Unfortunately, if the sum of the 'item' sizes is an odd multiple of $r$ the 'items' cannot be perfectly packed into bins of size $2r$. For this reason, and also because it is convenient to do so even when the sum of the 'item' sizes is a multiple of $2r$, we consider perfect packings into bins of sizes $r$ and $2r$. Assume for the moment that $r$ copies of 'items' of sizes $1, 3, 5, \ldots, r-2$

can be perfectly packed into bins of sizes $r$ and $2r$. If they are packed entirely into bins of size $2r$, then the number of 'items' in each bin must be even (as all 'item' sizes are odd), and so Lemma 5 applies and implies that the original items can be perfectly packed into bins of size $k$. On the other hand, suppose a bin of size $r$ is required. The set of 'items' that are packed into a bin of size $r$ corresponds to a set of intervals whose midpoints sum to $k/2$. Recall that the intervals are of even length. We divide each such interval into its first half and its second half, obtaining twice as many intervals, whose midpoints now sum to $k$. Now we can again use Lemma 5 to construct the perfect packing.

The final step in the proof is to show that we can indeed perfectly pack $r$ copies of each of the item sizes $1, 3, 5, \dots, r-2$ into bins of sizes $r$ and $2r$. We shall use a different packing depending upon whether $r = 4\alpha + 1$ or $r = 4\alpha + 3$.

For the case $r = 4\alpha + 1$, the 'items' are perfectly packed by the following simple procedure. We begin by packing one bin with $(1, 1, r-2)$, one bin with $(2i+1, 2i+1, r-2i-2, r-2i)$, for each $i = 1, \dots, \alpha-1$, and one bin with $(2i-1, 2i+1, r-2i, r-2i)$, for each $i = 1, \dots, \alpha$, (noting that in the final case, when $i = \alpha$, we get three 'items' of size $2i+1 = r-2i$). This packs four 'items' of each size larger than 1, and three 'items' of size 1. We can apply this packing $\alpha$ times, leaving us with one 'item' of each size larger that 1 and $\alpha+1$ 'items' of size 1. Then we pack one bin with $(1, 2i-1, r-2i)$ for each $i = 1, \dots, \alpha$. This uses up all the remaining 'items' (where $\alpha+1$ items of size 1 are used because there are two 'items' of size 1 when $i = 1$). For the numerical example, in which $\alpha = 1$, this construction says that we should pack 5 copies of 1 and 3 into bins of size 5 and 10 by first packing one bin with $(1, 1, 3)$ and then one bin with $(1, 3, 3, 3)$. This leaves one 'item' of size 3 and two of size 1. These perfectly pack into a bin of size 5.

When $r = 4\alpha + 3$ the procedure is very similar to that above. We begin by packing one bin with $(1, 1, r-2)$, one bin with $(2i+1, 2i+1, r-2i-2, r-2i)$, for each $i = 1, \dots, \alpha$, and one bin with $(2i-1, 2i+1, r-2i, r-2i)$, for each $i = 1, \dots, \alpha$. As before, this packs four 'items' of each size larger than 1, and three 'items' of size 1. We apply this packing $\alpha$ times, leaving us with three 'items' of each size larger than 1, and $\alpha+3$ 'items' of size 1. Then we pack one bin with $(2i-1, 2i+1, r-2i, r-2i)$, for each $i = 1, \dots, \alpha$. This leaves us with $\alpha+2$ 'items' of size 1, two 'items' of size $2\alpha+1$, and one 'item' of each other size. Finally, as before, we pack one bin with $(1, 2i-1, r-2i)$, for each $i = 1, \dots, \alpha+1$, which uses up all remaining items. ∎

## 3. Proof of Theorem 2 using randomized algorithms.

Recall the theorem statement: For any distribution $U\{j, k\}$, with $1 < j \le k-2$, there is an on-line algorithm A with running time bounded by a polynomial in $n$ and $k$, and with $E[W^A(L_n)] = \Theta(1)$. In this section we show that this is true if we are willing to consider randomized algorithms.

We rely on the following general result of Courcoubetis and Weber [9]. Let $\vec{p} = (p_1, \dots, p_d)$ be a discrete distribution on the set $S = \{s_1, \dots, s_d\}$; $p_i$ is the probability of item size $s_i$. Any packing of items with sizes from $S$ into a bin of size $k$ can be viewed as a nonnegative integer vector $\vec{c} = (c_1, \dots, c_d)$, where $\sum_{i=1}^{d} c_i s_i \le k$. Of particular interest are those vectors that give rise to a sum of exactly $k$, which we shall call *perfect packing configurations*. For instance, if $S = \{1, 2, 3\}$ and $k = 7$, one such configuration would be $(1, 0, 2)$. Let $\mathbb{P}_{S,k}$ denote the set of all perfect packing configurations for a given $S$ and $k$. Let $\Lambda_{S,k}$ be the convex cone in $\mathbb{R}^d$ spanned by all nonnegative linear combinations of configurations in $\mathbb{P}_{S,k}$.

Theorem (Courcoubetis and Weber [9]).

(a) *If $\vec{p}$ lies in the interior of $\Lambda_{S,k}$, then there is a polynomial-time randomized on-line bin packing algorithm $A$ with $E[W^A(L_n)] = O(1)$.*

(b) *If $\vec{p}$ lies on the boundary of $\Lambda_{S,k}$, then $E[W^{OPT}(L_n)] = \Theta(n^{1/2})$ and there is a polynomial-time randomized on-line bin packing algorithm $A$ with $E[W^A(L_n)] = \Theta(n^{1/2})$.*

(c) *If $\vec{p}$ lies outside of $\Lambda_{S,k}$, then $E[W^{OPT}(L_n)] = \Theta(n)$.*

Case (a) is the relevant one here. Case (b) holds for the distributions $U\{k-1,k\}$, but we already have shown in the introduction how the conclusion of (b) can be proved by more direct means for those distributions. We explain below how the algorithm implicit in case (a) works, but first we use that case to prove the version of Theorem 2 involving randomized algorithms. In general it is NP-hard to determine which of the three cases applies to a given distribution (as can be proved by a straightforward transformation from the PARTITION problem [14]). However, for the distributions $U\{j,k\}$ we can use the following lemma, which we shall prove using the perfect packing theorem.

Lemma 7. *For each $i,j,k$ with $i \le j < k-1$, there exist positive integers $r_i, s_i, m_i < 2k^2$ such that the set of $r_i j + s_i$ items consisting of $r_i + s_i$ items of size $i$ together with $r_i$ items of each of the other $j-1$ sizes can be packed perfectly into $m_i$ bins of size $k$.*

Note that this lemma implies that the $j$-dimensional vector $\bar{e} = (1,1,\ldots,1)$ is strictly inside the appropriate cone when $S = \{1,2,\ldots,j\}$, $j < k-1$. This is because $\bar{e}$ is in the interior of the cone spanned by vectors of the form $(r_i,\ldots,r_i,r_i+s_i,r_i,\ldots,r_i)$, $i = 1,\ldots,j$, and those vectors are sums of perfect packing configurations by Lemma 7. The proof of Theorem 2 thus follows from case (a) of the Courcoubetis and Weber theorem.

*Proof of Lemma* 7. We make use of the perfect packing theorem. There are two cases. If $k \ge i+j$, we simply set $r_i = k-i$ and $s_i = m_i = j(j+1)/2$. Note that the total size of $r_i$ items each of the sizes $1,\ldots,j$ equals $r_i j(j+1)/2$, so by the perfect packing theorem, we can perfectly pack them into $j(j+1)/2 = m_i$ bins of size $r_i = k-i$. The remaining $s_i = m_i$ items of size $i$ can then go one per bin to fill these bins up to size precisely $k$.

On the other hand, suppose $k < i+j$. Now things are a bit more complicated. We have $r_i = 2(k-i)$, $s_i = (k-j)(k-j-1)$, and $m_i = s_i + r_i(2j-k+1)/2$. By the perfect packing theorem $r_i$ items each of the sizes $1,\ldots,k-j-1$ perfectly pack in $s_i$ bins of size $k-i$. (Such items exist because by assumption $j < k-1$.) We then add the additional $s_i$ items of size $i$ to these bins, one per bin, to bring each bin up to size $k$. There remain $r_i$ items each of sizes $k-j$ through $j$, for a total of $r_i(j-(k-j-1)) = 2(m_i-s_i)$ items. These can be used to completely fill the remaining $m_i - s_i$ bins with pairs of items of sizes $(j,k-j),(j-1,k-j+1),\ldots,(\lceil k/2\rceil,\lfloor k/2\rfloor)$. Note that if $k$ is even, the last bin type contains two items of size $k/2$, but we have an even number of such items by our choice of $r_i = 2(k-i)$, so this presents no difficulty.

It is easy to verify that in both cases the values of $r_i, s_i, m_i$ are all less than $2k^2$. ∎

We now describe how the algorithms implicit in the Courcoubetis and Weber theorem [9] work for our distributions. Each bin can be labeled with a "type" that specifies the configuration of items it contains. Given a distribution $U\{j,k\}$, we choose a set of perfect packing configurations $\{\vec{c}_q : 1 \le q \le Q\}$ that spans a cone having $(1,1,\ldots,1)$ in its interior, say those configurations generated in the proof of Lemma 7.

Let $c_{q,i}$ denote the number of items of size $i$ used in configuration $\vec{c}_q$. (Note that by the lemma, we may assume that $Q \leq \sum_{i=1}^{j} m_i < 2jk^2 = O(k^3)$.) We imagine that there is a separate packing facility for each configuration $\vec{c}_q$, and that each arriving item is routed to one such packing facility, where it finds space in a partially full bin of that type or starts a new bin of that type. Let $e_{q,i}$ denote the number of different item sizes amongst those used in $\vec{c}_q$ for which there are presently fewer spaces at facility $q$ than there are for item size $i$. Note that $e_{q,i} = 0$ for $i$ such that $c_{q,i} = 0$. The routing probabilities are determined by finding an $\epsilon > 0$ such that the following linear program has a feasible solution:

$$\sum_{q=1}^{Q} \alpha_q (c_{q,i} + \epsilon e_{q,i}) = 1/j, \quad i = 1, \ldots, j.$$

$$\alpha_q > 0, \quad 1 \leq q \leq Q.$$

Such an $\epsilon$ exists because $(1, 1, \ldots, 1)$ is in the interior of the cone spanned by the $c_q$'s; a precise value for $\epsilon$ can be computed directly from the distance of $(1, 1, \ldots, 1)$ from the boundary of the cone, which itself can be derived from the proof of Lemma 7. Moreover, it is not difficult to see that this distance is such that the number of bits of precision needed to describe $\epsilon$ is $O(\log k)$. Thus the linear program can be constructed and solved in time polynomial in $k$, or in constant time for fixed $k$.

When an item of size $i$ is to be packed, it is randomly routed for packing at facility $q$ with probability

$$\alpha_q (c_{q,i} + \epsilon e_{q,i})/(1/j).$$

The impact of this construction is to ensure that facility $q$ receives items of size $i$ at a faster rate than items of size $i'$ whenever there are more spaces for items of size $i$ than $i'$ in partially full bins of type $q$. For the details of why this suffices, see [9].

**4. A deterministic algorithm.** The algorithm described at the end of the previous section is randomized but relatively straightforward. Our deterministic algorithm is a bit more complicated and works on slightly different principles. (Designing a new algorithm from scratch turns out to be easier than attempting to derandomize the above algorithm directly.) We shall use the following consequence of Lemma 7.

LEMMA 8. *For each $i, j, k$ with $i \leq j < k - 1$, there exist positive integers $r'_i, s'_i, m'_i < 4k^5$, with $s'_i \leq r'_i$, such that the set of $r'_i j - s'_i$ items consisting of $r'_i - s'_i$ items of size $i$ together with $r'_i$ items of each of the $j - 1$ other sizes can be packed perfectly into $m'_i$ bins of size $k$.*

*Proof.* Let $r_i, s_i, m_i > 0$ be as given in the conclusion of Lemma 7. This means that for $1 \leq i \leq j$, the set $R_i$ consisting of $r_i$ items each of sizes $1, 2, \ldots, j$ together with an additional $s_i$ items of size $i$ can be packed perfectly into $m_i$ bins. Note that by the proof of Lemma 7, we may assume that $s_i = (k-j)(k-j-1)$ if $i > k - j$ and otherwise $s_i = j(j+1)/2$. Denote the first quantity by $\overline{s}$ and the second by $\underline{s}$.

First let us suppose $j \leq k/2$, in which case there is no $i$ for which $i > k - j$. We can thus construct the set desired for $i, j, k$ by merging together one copy each of the sets $R_h$ for all $h$, $1 \leq h \leq j$ except $i$. This set has the parameters $r'_i = \sum_{q \neq i} r_h + \underline{s}$ and $s'_i = \underline{s}$, and packs perfectly into $m'_i = \sum_{q \neq i} m_h$ bins by Lemma 7.

The situation is a bit more complicated if $j > k/2$ and $s_i$ can take on both values $\underline{s}$ and $\overline{s}$. Now we construct our desired set by merging together $\overline{s}$ copies of each $R_h$,

$1 \le h \le k-j$, with $\underline{s}$ copies of each $R_h$, $k-j < h \le j$, and then deleting one copy of $R_i$. This set has the parameters $r'_i = \sum_{h=1}^{k-j} r_h\overline{s} + \sum_{h=k-j+1}^{j} r_h\underline{s} + \underline{s}\overline{s} - r_i$ and $s'_i = \underline{s}$ or $\overline{s}$ depending on whether or not $i \le k-j$. It packs perfectly into $m'_i = \sum_{h=1}^{k-j} m_h\overline{s} + \sum_{h=k-j+1}^{j} m_h\underline{s} - m_i$ bins by Lemma 7.

The fact that in both cases $r'_i, s'_i, m'_i < 4k^5$ can be derived from the above arguments and the proof of Lemma 7. $\square$

We now begin the description of our deterministic algorithm for $U\{j,k\}$. As in the randomized algorithm of the previous section, we shall build our packing based on the fixed set of $Q < 2k^3$ perfect packing configurations $\{\vec{c}_q : q = 1,\dots,Q\}$ derived from Lemma 7 for $j$ and $k$. (Note that these are the only bin configurations needed in the proof of Lemma 8, since that proof constructs its packings based on the packings of Lemma 7.) Every time we start a new bin in our packing, we permanently assign one of these configurations to it, as in the previous algorithm. The bin remains open until it is completely packed in the manner required by its assigned configuration. During certain phases of the algorithm we may also open some empty bins and assign them configurations, with the bins losing those assignments (and open status) if they remain empty at the end of the phase.

If $\vec{c} = (c_1, c_2, \dots, c_j)$ is one of our perfect packing configurations, let $(c_1 : a_1,\ c_2 : a_2,\ \dots\ , c_j : a_j)$ be a *labeled* configuration denoting a bin with configuration $\vec{c}$ that contains $a_i \le c_i$ items of size $i$, $1 \le i \le j$. As far as the packing process is concerned, the "state" of the system can at any time be described by giving the current number of open bins of each possible labeled configuration.

Let $I_n$ be the state after the $n$th item has been packed; $I_0$ denotes the empty state in which there are no partially packed bins. We define two functions of $I_n$ that are of use in the description and subsequent analysis of our algorithm. Suppose there are $x_i$ as-yet-unfilled spaces for items of size $i$ in the open bins. Let $X(I_n) = \sum_{i=1}^{j} x_i$ denote the total number of items that are required to complete the packing of the open bins of state $I_n$. Let $T(I_n) = j\sum_{i=1}^{j} r'_i\lfloor x_i/s'_i\rfloor$, where $r'_i$ and $s'_i$ are the numbers given in Lemma 8. Note that by Lemma 8, we can assign configurations to additional (empty) bins in such a manner that if there were exactly $T(I_n)/j$ items of each of the sizes $1,\dots,j$ to be packed then these could be used to perfectly pack these newly configured empty bins, with $s'_i\lfloor x_i/s'_i\rfloor$ items of size $i$ left over, $1 \le i \le j$: For each $i$ we would use $(r'_i - s'_i)\lfloor x_i/s'_i\rfloor$ items of size $i$ and $r'_i\lfloor x_i/s'_i\rfloor$ items of each of the other sizes to perfectly pack $m'_i\lfloor x_i/s'_i\rfloor$ empty bins. The $s'_i\lfloor x_i/s'_i\rfloor$ surplus items of size $i$ could be used to fill all but $u_i = x_i - s'_i\lfloor x_i/s'_i\rfloor$ of the spaces for items of size $i$ that already existed in our packing of the first $n$ items. Note that $0 \le u_i < s'_i$. Let $R_{min} = \min\{r'_i/s'_i : 1 \le i \le j\}$, and note that $R_{min} \ge 1$ since by Lemma 8, $s'_i \le r'_i$ for all $i$. It is easy to see that by definition of $T(I_n)$,

$$jR_{\min}X(I_n) - j\sum_{i=1}^{j} r'_i \le T(I_n) < 4k^6X(I_n). \tag{4.1}$$

The algorithm proceeds in phases, depending on the value of $X(I_n)$ at the end of the previous phase. The final state of one phase will also be viewed as the initial state of the following phase. We denote the indices of the final states of the phases by $n_h$, $h \ge 0$, with $n_0 = 0$. The sequence of $n_h$'s is defined inductively as follows. Suppose there are more than $n_h$ items so the current phase is not the last.

If $X(I_{n_h}) < L$, where $L$ is specified below, we take $n_{h+1} = n_h + 1$, that is, the next phase involves only the packing of the next item. This item is placed in the first bin in the list having a space for it or, if necessary, a new bin. For the purpose of defining "first," we assume the open bins are ordered lexicographically according to their labeled configuration, with ties broken according to the order in which the bins were opened. (Any definition of "lexicographic" will do, although presumably we would want all the empty labeled configurations to come after all the partially full ones.) When no open bin has space for the item, the new bin where it is placed is assigned the lexicographically first configuration that includes an item of the given size.

If, on the other hand, $X(I_{n_h}) \geq L$ then $n_{h+1} = n_h + T(I_{n_h})$. The phase proceeds as follows: Initially, a set of $m \equiv \sum_i m'_i \lfloor x_i/s'_i \rfloor < 2k^5 X(I_{n_h})$ empty bins are opened with configurations assigned in the manner described above. Thereafter, as each item arrives during the phase, it is packed in the first bin that has a space for an item of that size, with "first" defined as above. If during the phase we encounter more than $u_i + T(I_{n_h})/j$ items of some size $i$ then it is necessary to open additional bins for items of size $i$. Each time such an item arrives, the new bin is assigned the lexicographically first configuration that includes items of size $i$. The phase terminates after $T(I_{n_h})$ items have been packed or after the last item has been packed, whichever comes first. At the end of the phase, any still-empty bin that has a configuration assigned to it is stripped of that assignment and returned to the pool of unopened empty bins.

This completes our description of the algorithm (except for specifying the constant $L$, which we shall do below). We now begin our analysis. Our first goal is to show that $X$ and $X^2$ are well behaved. Lemma 9 bounds the expected change in these functions during a phase.

LEMMA 9. *For all $h \geq 0$,*

$$X(I_{n_{h+1}}) \leq X(I_{n_h}) + k - 1 \quad \text{if } X(I_{n_h}) < L \tag{4.2}$$

$$E\left[X(I_{n_{h+1}})^2 | X(I_{n_h})\right] \leq 100k^{18} X(I_{n_h}) \quad \text{if } X(I_{n_h}) \geq L. \tag{4.3}$$

*Proof.* When $X(I_{n_h}) < L$ the phase lasts for the packing of a single item. In the worst case, this item starts a new bin. Since no configuration can contain more than $k$ items, this means that $X(I_{n_{h+1}})$ has at most $k-1$ more items, so (4.2) holds.

Now assume $X(I_{n_h}) \geq L$, in which case the phase lasts for $T \equiv T(I_{n_h}) \leq 4k^6 X(I_{n_h})$ items by (4.1). The number $N_i$ of items of size $i$ amongst the $T$ items that are packed during this interval has a binomial distribution with mean $T/j$ and variance $(T/j)(1-1/j)$. If these $T$ items contained exactly $T/j$ items of each size then, using the scheme described above, we would have $X(I_{n_{h+1}}) = \sum_i u_i$. Any shortfall or surplus of $N_i$ over $T/j$ can produce at most $|N_i - T/j|$ extra partially full bins. Since by Lemma 8, $u_i < s'_i < 4k^5$, $1 \leq i \leq j$,

$$X(I_{n_{h+1}}) \leq \sum_{i=1}^{j} (u_i + (k-1)|N_i - T/j|) < 4jk^5 + k\sum_i |N_i - T/j| \,.$$

Squaring the above, taking expected values, and using (4.1), knowledge of the mean and variance of $N_j$, and Cauchy–Schwarz inequalities, we derive (4.3) from

$$\begin{aligned}
E[X(I_{n_{h+1}})^2] &< 16k^{12} + 8k^7 E\left[\sum_i |N_i - T/j|\right] + k^2 E\left[\left(\sum_i |N_i - T/j|\right)^2\right] \\
&\le 16k^{12} + 8k^7 \sum_i \sqrt{E(N_i - T/j)^2} + k^2 E\left[j \sum_i |N_i - T/j|^2\right] \\
&\le 16k^{12} + 8k^7 j\sqrt{(T/j)(1-1/j)} + k^2 j^2 (T/j)(1-1/j) \\
&\le 16k^{12} + 8k^{7.5}\sqrt{T} + k^3 T \\
&< 25k^{12} T \\
&\le 100k^{18} X(I_{n_h}). \qquad \square
\end{aligned}$$

The analysis which follows begins by considering the Markov chain $\{I_{n_h}\}$, restricted to the set of states $Z$ reachable from the empty state $z = \phi$ by a sequence of phases of the algorithm — note that in the chain $\{I_{n_h}\}$ a *step* corresponds to an entire phase of the algorithm. It is easily verified that $\{I_{n_h}\}$ restricted to $Z$ is irreducible, since any state can eventually be converted to $\phi$ by an arrival sequence having positive probability. We may also assume $\{I_{n_h}\}$ is aperiodic. To ensure this, we need only to augment the set $\{\vec{c}_q : q = 1, \ldots, Q\}$ of perfect packing configurations used by our algorithm to include the following two configurations, assuming they are not already present: (a) the configuration consisting of $k$ items of size 1 and (b) the configuration consisting of one size-2 item and $k-2$ items of size 1. (The latter configuration is possible since we are already assuming that $j > 1$.) Now whenever we are in empty state $\phi$ there will be ways of returning to it in either $k$ or $k-1$ steps, depending on the next items to arrive. This will prevent $\phi$ and every other state from being periodic.

Let $P_y^{(t)}(z)$ be the $t$-step transition probability from state $y$ to state $z$. We have the following convergence result.

LEMMA 10. *For any $j, k$, and $L \ge 200k^{18}$, $P_\phi(z)$ converges to a stationary distribution $\{\pi(z)\}$ geometrically quickly in the $X^2$ norm, i.e., there exists constants $K > 0$ and $\rho > 1$ such that for all $x \in Z$,*

$$\left|\sum_{z\in Z} X(z)^2 P_x^{(h)}(z) - \sum_{z\in Z} X(z)^2 \pi(z)\right| \le K\rho^{-h}, \quad h \ge 1. \tag{4.4}$$

*Proof.* We use the following result on geometric ergodicity, a specialization of Theorem 15.0.1 of [18] to countable chains, which uses the function $\mathbf{1}_C(x) = 1$ if $x \in C$ and 0 otherwise.

THEOREM (Meyn and Tweedie [18]). *Suppose $\{\Phi_k\}$ is an irreducible, aperiodic Markov chain on a countable state space $Z$, and let $P_y^{(t)}(z)$ be the $t$-step transition probability from state $y$ to state $z$ for $\{\Phi_k\}$. If there is a function $V : Z \to [1, \infty)$, constants $b < \infty$, $\beta > 0$, and a finite subset $C \subset Z$ such that for all $x \in Z$,*

$$\sum_{z\in Z} P_x(z) V(z) \le (1-\beta)V(x) + b\mathbf{1}_C(x),$$

*then $P_\phi^{(h)}(z)$ converges to a stationary distribution $\pi(z)$ geometrically quickly in the $V$ norm in the sense that there exist constants $K > 0$, $\rho > 1$ such that for all $x \in Z$,*

$$\sup_{|f|\le V} \left|\sum_{z\in Z} f(z) P_x^{(h)}(z) - \sum_{z\in Z} f(z)\pi(z)\right| \le K\rho^{-h}.$$

To apply the theorem, suppose $L \geq 200k^{18}$. Then by (4.2) and (4.3)

(4.5)
$$X(I_{n_{h+1}})^2 \leq (1/2)X(I_{n_h})^2 + (1/2)L^2 + 2(k-1)L + (k-1)^2 \quad \text{if } X(I_{n_h}) < L,$$

(4.6)
$$E[X(I_{n_{h+1}})^2 | X(I_{n_h})] \leq 100k^{18}(X(I_{n_h})/L)X(I_{n_h}) \leq (1/2)X(I_{n_h})^2 \quad \text{if } X(I_{n_h}) \geq L.$$

Let $V(z) = X(z)^2$ for all $z \in Z$ except $\phi$, for which we take $V(\phi) = 1$, and let $C = \{z : X(z) < L\}$. By (4.5)–(4.6) we can then take $\beta = 1/2$ and $b = L^2/2 + 2(k-1)L + (k-1)^2$. □

Now note that if we needed only to prove that the expected waste of our algorithm was bounded at the times $n_h$ at which phases end, we would be done. This is because the waste in any state $z$ is bounded by $X(z)$ (and so by $X(z)^2$) and Lemma 10 tells us that the Markov chain that considers only the states at the ends of phases has an equilibrium distribution and that the expected value of $X(z)^2$ under that equilibrium distribution (i.e., $\sum_{z \in Z} X(z)^2 \pi(z)$) is a finite constant; moreover, $E[X(I_{n_h})^2]$ converges geometrically to this constant. Unfortunately, we need to prove that $E[X(I_n)]$ is bounded by a fixed constant for *all* sufficiently large $n$, not just those $n$ at which phases end, and this latter result is not implied by Lemma 10. So further argument is needed.

Let us now construct a Markov chain that changes state after each item is packed, not after each phase. Note that $\{I_n\}$ is not Markov since we need to retain more information in a state than simply the current packing state, given that the number of steps to the end of a phase depends not only on the current packing state but also on the packing state that existed at the beginning of the phase and the number of steps taken since then. For a given beginning-of-phase state $z \in Z$, let $t(z)$ denote the length of the phase beginning at $z$. That is, $t(z) = 1$ if $X(z) < L$ and $t(z) = T(z)$ otherwise. The set of states of our Markov chain, which we denote by $\{I'_n\}$, is $Z \cup W$, where

$$W = \left\{ w(z, \xi, i) : z \in Z, \xi \in \{1, 2, \ldots, j\}^{t(z)}, 1 \leq i < t(z) \right\}.$$

The state $w(z, \xi, i)$ corresponds to the packing state (the ordered set of partially empty and empty bins to which configurations have been assigned) that would be obtained from $z$ after the beginning-of-phase assignment of configurations to empty bins has been made and after items of sizes $\xi_1, \xi_2, \ldots, \xi_i$ have been packed (in that order), where $\xi = (\xi_1, \xi_2, \ldots, \xi_{t(z)})$. The norm $X(w(z, \xi, i))$ is simply the number of empty slots in this derived set of bins.

The transition function $p$ for $\{I'_n\}$ is defined as follows: For states $z \in Z$, $p_z(w(z', \xi, i))$ is nonzero only if $z' = z$ and $i = 1$, with all $j^{t(z)}$ such states being equally likely. We have $p_{w(z,\xi,i)}(w(z, \xi, i+1)) = 1$ for states $w(z, \xi, i) \in W$ with $i < t(z) - 1$, and 0 for all other states, i.e., the transition is deterministic. Finally, the transition from $w(z, \xi, t(z) - 1)$ is also deterministic, with the state to which the transition is made being the state $z' \in Z$ that would result from the phase starting at $z$ if the $t(z)$ items to arrive had the sizes and order specified by $\xi$. Note that $\{I'_n\}$ inherits from $\{I_{n_h}\}$ the properties of being irreducible and aperiodic.

LEMMA 11. *Let $\pi$ be the equilibrium distribution for $\{I_{n_h}\}$, and let $E_\pi[t]$ denote the expected value of $t(z)$ under $\pi$, i.e., $\sum_{z \in Z} \pi(z)t(z)$. Then $\{I'_n\}$ has an equilibrium* *distribution $\pi'$ where*

$$\pi'(z) = \frac{\pi(z)}{E_\pi[t]}, \quad z \in Z,$$
$$\pi'(w(z,\xi,i)) = \frac{\pi(z)}{j^{t(z)}E_\pi[t]}, z \in Z,\ \xi \in \{1,2,\dots,j\}^{t(z)},\ 1 \le i < t(z).$$

*Proof.* Note that $E_\pi[t]$ is a well-defined constant, since $E_\pi[X] \equiv \sum_{z\in Z}\pi(z)X(z)$ is bounded by Lemma 10, and $t(z) \le 4k^6X(z)$ by (4.1). Thus $\pi'$ is well defined. To prove Lemma 11 we use a standard result from [12, p. 393].

THEOREM (Feller [12]). *Suppose we are given an aperiodic, irreducible Markov chain on a countable state space $S$ with transition function $p$, and there exists a function $\mu : S \to [0,1]$ such that* (a) $\sum_{s\in S}\mu(s) = 1$ *and* (b) *for all states $s' \in S$, $\mu(s') = \sum_{s\in S}\mu(s)p_s(s')$. If we let $s_n$ denote the state after $n$ steps, then we have for all states $s, s' \in S$,*

$$\lim_{n\to\infty} P[s_n = s'|s_0 = s] = \mu(s').$$

Given this theorem, we prove Lemma 11 by showing that (a) and (b) hold for $\pi'$. For (a), we have

$$\begin{aligned}\sum_{z\in Z}\pi'(z) + \sum_{w\in W}\pi'(w) &= \frac{\sum_{z\in Z}\pi(z)}{E_\pi[t]} + \sum_{z\in Z}\sum_{i=1}^{t(z)-1}\sum_{\xi\in\{1,2,\dots,j\}^{t(z)}}\frac{\pi(z)}{j^{t(z)}E_\pi[t]}\\ &= \sum_{z\in Z}\frac{\pi(z)}{E_\pi[t]} + \sum_{z\in Z}\frac{\pi(z)(t(z)-1)}{E_\pi[t]}\\ &= \sum_{z\in Z}\frac{\pi(z)t(z)}{E_\pi[t]} = 1.\end{aligned}$$

Property (b) holds for all states $w(z,\xi,i) \in W$ by design, and for states $z \in Z$ by the fact that $\pi$ is the equilibrium distribution for $\{I_{n_h}\}$. Thus the above theorem of Feller applies and the proof of Lemma 11 is complete. ∎

There are now only two more steps to completing a proof that our deterministic algorithm has bounded expected waste. We must simply verify the following two claims:

$$E_{\pi'}[X] \equiv \sum_{x\in Z\cup W}\pi'(x)X(x) < \infty, \tag{4.7}$$

$$\lim_{n\to\infty}\sum_{x\in Z\cup W}P[I'_n = x]\cdot X(x) = E_{\pi'}[X]. \tag{4.8}$$

Claim (4.8) follows from (4.7) and Theorem 14.0.1 of [18], which when specialized to countable chains goes as follows.

THEOREM (Meyn and Tweedie [18]). *Suppose $\{\Phi_k\}$ is an irreducible, aperiodic Markov chain on a countable state space $Z$ having equilibrium distribution $\pi$. Let $\pi_n$ be the distribution of states at step $n$ for some fixed initial configuration, and suppose $f : Z \to [0,\infty]$. Then if $E_\pi[f(z)] < \infty$, we have*

$$\lim_{n\to\infty} E_{\pi_n}[f(z)] = E_\pi[f(z)].$$

For (4.7) we first need to bound $X(x)$ for $x \in W$. Since $X(x)$ can increase by no more than $k-1$ each time an item is packed, we conclude by (4.1) that

$$X(w(z,\xi,i)) \le X(z) + (k-1)t(z) < 4k^7 X(z). \tag{4.9}$$

Thus

$$\begin{aligned}\sum_{x\in Z\cup W} \pi'(x)X(x) &\le \sum_{z\in Z} \frac{\pi(z)X(z)}{E_\pi[t]} + \sum_{z\in Z}\sum_{i=1}^{t(z)-1}\sum_{\xi\in\{1,2,\dots,j\}^{t(z)}} \frac{\pi(z)X(w(z,\xi,i))}{j^{t(z)}E_\pi[t]}\\ &\le \frac{1}{E_\pi[t]}\sum_{z\in Z}\pi(z)t(z)4k^7X(z)\\ &\le \frac{1}{E_\pi[t]}\sum_{z\in Z}\pi(z)16k^{13}X(z)^2 = \frac{16k^{13}}{E_\pi[t]}\sum_{z\in Z}\pi(z)X(z)^2.\end{aligned}$$

Lemma 10 says that $\sum_{z\in Z}\pi(z)X(z)^2$ is bounded. Hence (4.7) holds.

## 5. Proof of Theorem 3.

**5. Proof of Theorem 3.** Recall the theorem statement: If $L_n$ has item sizes generated according to $U(0,u]$ for $0<u<1$, and $A$ is any on-line algorithm, then there exists a constant $c>0$ such that $E[W^A(L_n)] > cn^{1/2}$ for infinitely many $n$.

*Proof.* Let $w(t)$ denote the amount of empty space in partially filled bins after $t$ items have been packed. We show that for any $n>0$ the expected value of the average of $w(1),\dots,w(n)$ is $\Omega(n^{1/2}u^3)$. This implies that $E[w(n)]$ must be $\underline{\Omega}(n^{1/2}u^3)$, i.e., not $o(n^{1/2})$.

Consider packing item $a_{t+1}$. Let $v(t)$ denote the number of nonempty bins that have a gap of at least $u^2/8$ after the first $t$ items have been packed. There are at most $v(t)$ bins into which one can put an item larger than $u^2/8$. Therefore, if $a_{t+1}$ is to leave a gap of less than $\delta$ in its bin, either it must have size less than $u^2/8$ or its size must be within $\delta$ of the empty space in one of these $v(t)$ bins with gaps larger than $u^2/8$. The probability of this is at most $[u^2/8+\delta v(t)]/u$. By choosing $\delta = u^2n^{-1/2}/8$, conditioning on whether $v(t)$ is greater or less than $n^{1/2}$ and noting that the size of $a_{t+1}$ is distributed as $U(0,u]$ independent of $v(t)$, we have

$$P(a_{t+1} \text{ leaves gap } <\delta) \le P(v(t)\ge n^{1/2}) + u/4.$$

Now

$$\begin{aligned} E[w(t)] &\ge \delta\textstyle\sum_{s=0}^{t-1} P(a_{s+1} \text{ is last in a bin and leaves gap } \ge\delta)\\ &= \delta\textstyle\sum_{s=0}^{t-1}[P(a_{s+1} \text{ is last in a bin})\\ &\qquad\qquad - P(a_{s+1} \text{ is last in a bin and leaves gap } <\delta)]\\ &\ge \delta\textstyle\sum_{s=0}^{t-1}[P(a_{s+1} \text{ is last in a bin}) - P(a_{s+1} \text{ leaves gap } <\delta)]\\ &\ge \delta\textstyle\sum_{s=0}^{t-1}\left[P(a_{s+1} \text{ is last in a bin}) - \sum_{s=0}^{t-1}P(v(s)\ge n^{1/2}) - \sum_{s=0}^{t-1}u/4\right].\end{aligned}$$

Let $S_t$ be the sum of the first $t$ item sizes and note that $S_t$ is a lower bound on the number of bins and hence on the number of items that are the last item in a bin. We thus have

$$E\left[\sum_{s=0}^{t-1}P(a_{s+1} \text{ is last in a bin})\right] \ge E[S_t] = tu/2.$$

Using the fact that $\delta = u^2 n^{-1/2}/8$, we then have

$$E[w(t)] \geq (u^2 n^{-1/2}/8)\left[tu/4 - \sum_{s=0}^{t-1} P(v(s) \geq n^{1/2})\right].$$

If $\sum_{s=0}^{n-1} P(v(s) \geq n^{1/2}) \leq nu/24$, we have for all $t \geq n/2$,

$$E[w(t]) \geq (u^2 n^{-1/2}/8)[nu/8 - nu/24] = u^3 n^{1/2}/96.$$

This implies

$$E\left[\frac{1}{n}\sum_{t=1}^{n} w(t)\right] \geq u^3 n^{1/2}/192.$$

On the other hand, if $\sum_{s=0}^{n-1} P(v(s) \geq n^{1/2}) \geq nu/24$ then

$$\begin{aligned} E\left[\frac{1}{n}\sum_{t=1}^{n} w(t)\right] &\geq \frac{1}{n}\sum_{t=1}^{n} P(v(t) \geq n^{1/2}) n^{1/2}(u^2/8) \\ &\geq n^{1/2}(u^2/8)(u/24) = u^3 n^{1/2}/192. \end{aligned}$$

These imply that $E[w(n)]$ is $\underline{\Omega}(n^{1/2})$. $\square$

It should be noted that the above proof relies heavily on the fact that the distribution is continuous, since this is the reason why the union of $n^{1/2}$ intervals of size $\delta$ cannot cover the full probability space. Our discrete distributions $U\{j,k\}$ do not have this failing and for this reason we can obtain significantly better average-case behavior for them.

**6. Concluding remarks.** In this paper we have proved a combinatorial theorem about perfect packings and used it to derive results about the expected behavior of online bin packing algorithms under the discrete uniform distributions $U\{j,k\}$, $1 < j \leq k-2$, in which item sizes are restricted to the finite set $\{i/k : \ 1 \leq i \leq j\}$, each item size being equally likely. These results, which imply that for any such distribution there exist on-line algorithms with bounded expected waste, contrast distinctly with the result proved in section 5 about the corresponding continuous uniform distributions $U[0, j/k]$, for which expected waste must grow as $n^{1/2}$. We presented both randomized algorithms and deterministic ones, with the price for the deterministic ones being an apparent increase in both running time and expected waste. It should be noted, however, that in both cases the upper bounds we have proved on expected waste are quite large, and even though more careful arguments might enable us to lower the estimated bounds substantially, it is not clear that the "constant" waste provided by our algorithms would in practice be better than the slowly growing waste that other heuristics might provide, especially for large values of $k$.

One additional weakness of our results as proved in sections 3 and 4 is that they require a distinct algorithm for each $j, k$, rather than yielding a single algorithm that works and provides bounded expected waste for all $U\{j,k\}$, $1 < j \leq k-2$. It is not difficult to see, however, that our underlying lemmas and their proofs can be used to provide general on-line algorithms (both randomized and deterministic) with running times of the form $O(np(k))$, where $p(k)$ is a polynomial in $k$ that bounds the time to pack an individual item. The parameters $k$ and $j$ can either be given as part of the input or, better yet, derived empirically by observing the first-arriving items.

Given $j$ and $k$, the general algorithms determine the required set of perfect packing configurations $\{\vec{c}_h : 1 \le h \le H\}$ by implementing the proof of Lemma 7. The required values of $r_i, s_i, m_i$ and (in the deterministic case) $r'_i, s'_i, m'_i$ can be determined using the arguments in the proofs of that lemma and Lemma 8. It is then a simple exercise in data structures (which we leave to the reader) to verify that the packing operations for both the randomized and deterministic algorithms can be performed in worst-case polynomial time (in $k$), independent of $n$.

Note, however, that these "generalized" algorithms are still quite limited, in that they only know how to pack items generated according to the set of probability distributions $U\{j, k\}$. They may not even construct legal packings for other discrete distributions, even though many such distributions also have on-line algorithms that yield constant expected waste. This follows from the theorem of Courcoubetis and Weber [9] cited in section 3, which relates the existence of such algorithms for a distribution $\vec{p}$ to the existence of appropriate perfect packing configurations. Although as we have remarked it is NP-complete to tell for an arbitrary distribution which case of the Courcoubetis–Weber theorem applies, it is easy to construct individual distributions other than the $U\{j, k\}$ that satisfy the hypotheses of this theorem, and it is likely that there are other general classes of interesting distributions for which the needed perfect packing configurations can be proved to exist by appropriate analogues of our perfect packing theorem. We leave the investigation of such questions to future research.

One useful tool may be the recent result of [10, 11] that for discrete distributions with integer item sizes one can determine which case of the Courcoubetis–Weber theorem applies in pseudopolynomial time (time bounded by a polynomial in the bin capacity $B$, rather than polynomial in $\log B$ as is required by the definition of polynomial time). For the integer item size case, these references also show that complicated algorithms such as those presented here will not be necessary when dealing with distributions for which $E[W^{\mathrm{OPT}}(L_n)]$ is sublinear. In [10] it is shown that a simple deterministic $O(nB)$-time on-line variant on the *Sum-of-Squares* algorithm of [11] has $O(1)$ expected waste for all distributions with $E[W^{\mathrm{OPT}}(L_n)] = O(1)$ and $\Theta(\sqrt{n})$ expected waste for all distributions with $E[W^{\mathrm{OPT}}(L_n)] = \Theta(\sqrt{n})$.

**Acknowledgment.** We are grateful to a referee for many helpful comments, including a major simplification to the proof of (4.7).

## REFERENCES

[1] J. L. Bentley, D. S. Johnson, F. T. Leighton, C. C. McGeoch, and L. A. McGeoch, *Some unexpected expected behavior results for bin packing*, in Proceedings of the 16th Annual ACM Symposium on Theory of Computing, ACM, New York, 1984, pp. 279–288.

[2] E. G. Coffman, Jr., C. Courcoubetis, M. R. Garey, D. S. Johnson, L. A. McGeoch, P. W. Shor, R. R. Weber, and M. Yannakakis, *Fundamental discrepancies between average-case analyses under discrete and continuous distributions*, in Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, ACM, New York, 1991, pp. 230–240.

[3] E. G. Coffman, Jr., D. S. Johnson, L. A. McGeoch, P. W. Shor, and R. R. Weber, *Bin Packing with Discrete Item Sizes, Part* III*: Average Case Behavior of FFD and BFD*, in preparation.

[4] E. G. Coffman, Jr., D. S. Johnson, P. W. Shor, and R. R. Weber, *Bin Packing with Discrete Item Sizes, Part* IV*: Markov Chains, Computer Proofs, and Average-Case Analysis of Best Fit Bin Packing*, in preparation.

[5] E. G. Coffman, Jr., D. S. Johnson, P. W. Shor, and R. R. Weber, *Bin Packing with Discrete Item Sizes, Part* V*: Tight Bounds on Best Fit*, in preparation.

[6] E. G. Coffman, Jr., D. S. Johnson, P. W. Shor, and R. R. Weber, *Markov chains, computer proofs, and average-case analysis of best fit bin packing*, in Proceedings of the 25th Annual ACM Symposium on Theory of Computing, ACM, New York, 1993, pp. 412–421.

[7] E. G. Coffman, Jr., D. S. Johnson, P. W. Shor, and R. R. Weber, *Bin packing with discrete item sizes, part* II*: Tight bounds on first fit*, Random Structures Algorithms, 10 (1997), pp. 69–101.

[8] E. G. Coffman, Jr., and G. S. Lueker, *An Introduction to the Probabilistic Analysis of Packing and Partitioning Algorithms*, Wiley, New York, 1991.

[9] C. Courcoubetis and R. R. Weber, *Stability of on-line bin packing with random arrivals and long-run average constraints*, Probab. Engrg. Inform. Sci., 4 (1990), pp. 447–460.

[10] J. Csirik, D. S. Johnson, C. Kenyon, J. B. Orlin, P. W. Shor, and R. R. Weber, *On the sum-of-squares algorithm for bin packing*, in Proceedings of the 32nd ACM Symposium on Theory of Computing, ACM, New York, 2000, to appear.

[11] J. Csirik, D. S. Johnson, C. Kenyon, P. W. Shor, and R. R. Weber, *A self organizing bin packing heuristic*, in Algorithm Engineering and Experimentation, M. Goodrich and C. C. McGeoch, eds., Lecture Notes in Comput. Sci. 1619, Springer-Verlag, Berlin, 1999, pp. 246–265.

[12] W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. I, 3rd ed., Wiley, New York, 1968.

[13] S. Floyd and R. M. Karp, *FFD bin packing for item sizes with distributions on* $[0, 1/2]$, Algorithmica, 6 (1991), pp. 222–240.

[14] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman, New York, 1979.

[15] C. Kenyon, Y. Rabani, and A. Sinclair, *Biased random walks, Lyapunov functions, and stochastic analysis of best fit bin packing*, in Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, Atlanta, GA, 1996, pp. 351–358.

[16] W. Knödel, *A bin packing algorithm with complexity* $o(nlogn)$ *in the stochastic limit*, in Proceedings of the 10th Symposium on Mathematical Foundations of Computer Science, J. Gruska and M. Chytil, eds., Lecture Notes in Comput. Sci. 118, Springer-Verlag, Berlin, 1981, pp. 369–378.

[17] G. S. Lueker, *An Average-Case Analysis of Bin Packing with Uniformly Distributed Item Sizes*, Tech. Report 181, Dept. of Information and Computer Science, University of California, Irvine, CA, 1982.

[18] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*, Springer-Verlag, Berlin, 1993.

[19] P. W. Shor, *The average case analysis of some on-line algorithms for bin packing*, Combinatorica, 6 (1986), pp. 179–200.

[20] P. W. Shor, *How to pack better than Best Fit: Tight bounds for average-case on-line bin packing*, in Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1991, pp. 752–759.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 3, pp. 403–418

# THE CAPACITATED $K$-CENTER PROBLEM*

SAMIR KHULLER† AND YORAM J. SUSSMANN‡

**Abstract.** The capacitated $K$-center problem is a basic facility location problem, where we are asked to locate $K$ facilities in a graph and to assign vertices to facilities, so as to minimize the maximum distance from a vertex to the facility to which it is assigned. Moreover, each facility may be assigned at most $L$ vertices. This problem is known to be NP-hard. We give polynomial time approximation algorithms for two different versions of this problem that achieve approximation factors of 5 and 6. We also study some generalizations of this problem.

**Key words.** approximation algorithms, facility location, $K$-center

**AMS subject classifications.** 05C85, 68Q20, 68R10, 90C27, 90C35

**PII.** S0895480197329776

**1. Introduction.** The $K$-center problem is a basic facility location problem [18] and is defined as follows: given an edge-weighted graph $G = (V, E)$, find a subset $S \subseteq V$ of size at most $K$ such that each vertex in $V$ is "close" to some vertex in $S$. More formally, the objective function is defined as follows:

$$\min_{S\subseteq V} \max_{u\in V} \min_{v\in S} d(u, v),$$

where $d$ is the distance function. For example, one may wish to install $K$ fire stations and minimize the maximum distance (response time) from a location to its closest fire station. The problem is known to be NP-hard [8].

An approximation algorithm with a factor of $\rho$, for a minimization problem, is a polynomial time algorithm that guarantees a solution with cost at most $\rho$ times the cost of an optimal solution. Approximation algorithms for the basic $K$-center problem have been very well studied and are known to be optimal [7, 9, 10, 11]. These schemes present natural methods for obtaining an approximation factor of 2. Several approximation algorithms are known for interesting generalizations of the basic $K$-center problem as well [3, 10, 17]. The generalizations include cases when each node has an associated "cost" for placing a center on it, and rather than limiting the number of centers, we have a limited budget [10, 17]. Other generalizations include cases where the vertices have weights and we consider the weighted distance from a node to its closest center [3, 17].

Recently, a very interesting generalization that we call the capacitated $K$-center problem was studied by Bar-Ilan, Kortsarz, and Peleg [1]. The input specifies an upper bound on the number of centers $K$, as well as a maximum load $L$. We have to output a set of at most $K$ centers, as well as an assignment of vertices to centers. No

*Received by the editors November 10, 1997; accepted for publication (in revised form) March 16, 2000; published electronically June 20, 2000. A preliminary version of this paper appeared in the Proceedings of the Fourth Annual European Symposium on Algorithms, Barcelona, Spain, 1996.
http://www.siam.org/journals/sidma/13-3/32977.html

†Department of Computer Science and UMIACS, University of Maryland, College Park, MD 20742 (samir@cs.umd.edu). The research of this author was supported by NSF research initiation award CCR-9307462 and NSF CAREER award CCR-9501355.

‡Department of Computer Science, University of Maryland, College Park, MD 20742. Current address: Fundtech Corporation, 157 Technology Parkway, Suite 100, Norcross, GA 30092 (yorams@fundtech.com). The research of this author was supported by NSF CAREER award CCR-9501355 and was performed while the author a Ph.D. student at the University of Maryland.

more than $L$ vertices may be assigned to a single center. Under these constraints, we wish to minimize the maximum distance between a vertex $u$ and its assigned center $\phi(u)$. Formally, we have

$$\min_{S \subseteq V} \max_{u \in V} d(u, \phi(u))$$

such that

$$|\{u \mid \phi(u) = v\}| \leq L \quad \forall v \in S,$$

where

$$\phi : V \to S.$$

Bar-Ilan, Kortsarz, and Peleg [1] gave the first polynomial time approximation algorithm for this problem with an approximation factor of 10. Various applications for capacitated centers were first mentioned in [14, 15]. A slightly different problem, where the radius is fixed and where one has to minimize the number of centers, shows up in the Sloan digital sky survey project [13].

**1.1. Our results.** In section 2 we discuss a simplification of the problem where a node may appear multiple times in $S$ (i.e., more than one center can be put at a node). We will refer to this problem as the capacitated multi-$K$-center problem. By introducing some new ideas and using the basic approach proposed in [1], we are able to give a polynomial time algorithm that achieves an approximation factor of 5. In section 3 we show how to solve the problem when we are allowed only one center at a vertex. The high-level structure of the algorithm is the same, but the assignment of centers to vertices has to be done extremely carefully. This problem will be referred to as the capacitated $K$-center problem. For this version of the problem we obtain an approximation factor of 6. It is worth noting that, in fact, we prove that our solution is at most 6 times an optimal solution that is allowed to put multiple centers at a single vertex.

The algorithm can be easily extended to the more general case when each vertex has a demand $d_i$, and multiple centers may be used to satisfy its demand. The total demand assigned to any center should not exceed $L$. Using the method in [1], we can obtain an approximation factor of 13 for the version with costs. (Each vertex has a cost for placing a center on it, and we are working with a fixed budget.) In section 4 we study some other variants of this problem. Let a $(c_1K, c_2L, c_3R)$ solution denote a solution using at most $c_1K$ centers, each with a load of at most $c_2L$, which assigns every node to a center at distance at most $c_3R$, where $R$ is the radius of the optimal solution. For any $x \geq 1$, let $c = \frac{x+1}{x}$. We obtain a $\left(\frac{2}{c}K, cL, 2R\right)$ solution for the capacitated multi-$K$-center problem, and a $\left(\frac{2}{c}K, cL, 4R\right)$ solution for the capacitated $K$-center problem.

## 2. Algorithm for capacitated multi-$K$-centers.

We first give a high-level description of the algorithm. We may assume for simplicity that $G$ is a complete graph, where the edge weights satisfy the triangle inequality. (We can always replace any edge by the shortest path between the corresponding pair of vertices.)

**High-level description.** The algorithm uses the threshold method used for the $K$-center problem [4, 9, 10]. Sort all edge weights in nondecreasing order. Let the (sorted) list of edges be $e_1, e_2, \ldots, e_m$. For each $i$, let the threshold graph $G_i$

be the unweighted subgraph obtained from $G$ by including edges of weight at most $w(e_i)$. Run the algorithm below for each $i$ from 1 to $m$, until a solution is obtained. (Hochbaum and Shmoys [10] suggest using binary search to speed up the computation. If running time is not a factor, however, it does appear that to get the best solution (in practice) we should run the algorithm for all $i$ and take the best solution.) In each iteration, we work with the unweighted subgraph $G_i$. Since $G_i$ is an unweighted graph, when we refer to the distance between two nodes, we refer to the number of edges on a shortest path between them. In iteration $i$, we find a solution using some number of centers. If the number of centers exceeds $K$, we prove that there is no solution with cost at most $w(e_i)$. If the number of centers is at most $K$, we show that the maximum distance in $G_i$ between a vertex and its assigned center is at most 5. This gives an approximation factor of 5.

Observe that if we select as centers a set of nodes that is sufficiently well separated in $G_i$, then no two centers that we select can share the same center in the optimal solution, if the optimal solution has radius $w(e_i)$. This suggests the following initial approach. First find a maximal independent set $I$ in $G_i^2$. ($G_i^2$ is the graph obtained by adding edges to $G_i$ between nodes that have a common neighbor.) This technique was introduced by Hochbaum and Shmoys [9, 10] and has been used extensively to solve $K$-center problems. We refer to a node in the maximal independent set as a *monarch*. The algorithm also constructs a "tree" of monarchs which will be used to assign vertices to centers. There are two key differences between our algorithm and the one presented in [1].

1. We use a specific procedure to find a maximal independent set, as opposed to selecting an arbitrary maximal independent set.
2. We deal with all monarchs uniformly rather than dealing with the light and heavy[1] monarchs separately as in [1].

Each monarch has an empire that consists of a subset of vertices within the immediate neighborhood of the monarch in $G_i^2$. When a monarch is added to the maximal independent set, all such vertices that do not currently belong to an empire are added to this monarch's empire. The algorithm also constructs a tree of monarchs as the monarchs are selected. This tree has the property that an edge in the tree corresponds to a pair of monarchs whose distance in $G_i$ is *exactly* three. Each monarch then tries to collect a domain of size $L$—a subset of vertices that are close to the monarch and assigned to it. In doing so, a monarch may grab vertices from other empires if none are available in its own empire. After this process is complete there may still be unassigned vertices. We then use the tree of monarchs to assign new centers to handle the unassigned vertices. Nodes that are left unassigned in a particular empire may be assigned to the parent monarch. Eventually, we will put additional centers at the monarch vertices (recall that more than one center may be located at a single vertex).

```
CAPACITATED-CENTERS ($G = (V, E), K, L$).
1. Sort all edges in nondecreasing weight order ($e_1, \ldots, e_m$).
2. for $i = 1$ to $m$ do
3.     Let $G_i = (V, E_i)$ where $E_i = \{e_1, \ldots, e_i\}$.
4.     Unmark all vertices.
5.     if ASSIGNCENTERS($G_i$) then exit.
6. end-proc
```

[1]These terms will be explained shortly.

```
AssignCenters($G_i$).
1.  successful = true
2.  Let $n_i^c$ be the number of vertices in connected component $G_i^c$.
3.  if $(\sum_c \lceil \frac{n_i^c}{L} \rceil) > K$ then successful = false
    else
4.    for each connected component $G_i^c$ of $G_i$ do
5.        AssignMonarchs($G_i^c$).
6.        AssignDomains($G_i^c$).
7.        ReAssign($G_i^c$).
8.    if total centers used $> K$ then successful = false
9.  return (successful)
10. end-proc
```

AssignCenters($G_i$) tries to assign centers within each connected component. Each component is dealt with separately. (This can be done because if there is an optimal solution with maximum radius $w(e_i)$, then no center in the optimal solution will be assigned vertices that belong to different connected components of $G_i$.) A lower bound on the number of centers in each component is $\lceil \frac{n_c}{L} \rceil$, where $n_c$ is the number of vertices in $G_i^c$. If the number of allowed centers is smaller than $\sum_c \lceil \frac{n_c}{L} \rceil$, then there is no solution.

AssignMonarchs($G_i^c$) (described in Appendix A) assigns monarchs (nodes in the independent set) in a breadth first search (BFS) manner. After putting a vertex in the independent set, mark all unmarked nodes within distance 2 in $G_i$. To pick a new vertex to add to the independent set, pick an unmarked vertex that is adjacent to a marked vertex. Rather than describing the algorithm in terms of $(G_i^c)^2$, we work with $G_i^c$ to separate the nodes in a monarch's empire into level-1 and level-2 nodes. The level-1 nodes are adjacent to the monarch, and the level-2 nodes are at distance 2 from the monarch. Define $E_1(v)$ and $E_2(v)$ to be the level-1 and level-2 nodes, respectively, in $v$'s empire. Thus the empire of $v$ is $E_1(v) \cup E_2(v)$. The algorithm maintains a queue $Q$ which is initialized by adding one vertex. Neighbors of level-2 nodes, which are unmarked, are candidates to be chosen in the independent set and are added to $Q$.

Before discussing the pertinent properties of AssignMonarchs, we show its execution on the simple example given in Figure 1. The algorithm starts from vertex 1, which is made a monarch. Vertex 2 is added to $E_1(1)$ (level 1 in its empire), and vertices 3 and 4 are added to $E_2(1)$. $Q$ currently contains vertices 5, 8, and 10. Vertex 5 is then chosen as a monarch, and vertices 6 and 7 are added to $E_1(5)$. Vertices 8 and 9 are both added to $E_2(5)$. $Q$ now contains vertices 8, 10, 14, and 16. The next vertex chosen from $Q$ is 10 since 8 is marked. Vertices 11 and 12 are added to $E_1(10)$, and vertex 13 is added to $E_2(10)$. $Q$ now contains vertices 8, 14, and 16. Vertex 14 is now chosen from $Q$, and vertices 15 and 16 are added to $E_1(14)$ and $E_2(14)$, respectively. The algorithm stops since there are no more unmarked vertices in $Q$.

There are a few important properties of the monarchs produced by algorithm AssignMonarchs.

*Important properties.*

1. The distance between any two monarchs is at least 3.
2. The distance between a monarch $m$ (except for the root) and its "parent monarch" $p(m)$ in the tree is exactly 3.
3. The distance between a monarch and any vertex in its empire is at most 2.
4. Each monarch (except for the root) has at least one edge to a level-2 vertex

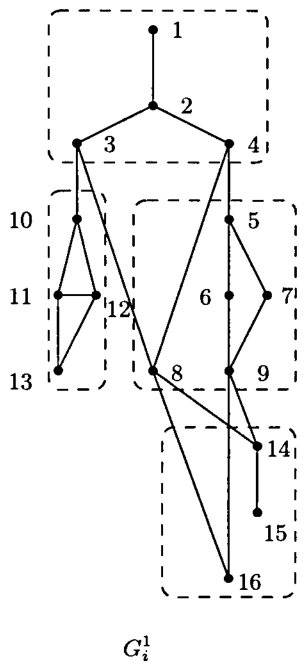


FIG. 1. *Example to show execution of* ASSIGNMONARCHS.

in its parent monarch's empire. Moreover, each such level-2 vertex has only *one* such neighbor that is a monarch. More generally, *any* vertex can have at most *one* neighbor that is a monarch (corollary of property 1).

Procedure ASSIGNDOMAINS (Appendix B) tries to assign a domain of size at most $L$ to each monarch. The objective is to assign as many vertices as possible to some domain, subject to the following constraints.

1. A vertex may be assigned to a monarch's domain only if it is at distance at most 2 from the monarch.
2. A monarch may include in its domain a vertex from another empire only if each vertex in its empire belongs to some domain.

This assignment can be viewed as a "b-matching" problem [16]. One way to implement this procedure is by finding a min-cost maximum flow in an appropriate graph $G'$ defined as follows.

Let $G' = (M \cup V \cup \{s,t\}, E')$, where $M$ is the set of monarchs in $G_i^c$ and $E' = \{(m,v) \mid m \in M,\, v \in V$, distance from $m$ to $v$ is at most $2\}$. Add edges $\{(s,m) \mid m \in M\}$ and $\{(v,t) \mid v \in V\}$. For $m \in M$, $v \in V$, set capacities $u(s,m) = L$, $u(m,v) = 1$, and $u(v,t) = 1$. Cost of edge $c(m,v) = 1$ if $v$ is not in $m$'s empire. Cost of all other edges is 0. Compute a min-cost maximum integral flow in $G'$ from $s$ to $t$. For each monarch $m$, set $domain(m) = \{v \mid v$ receives one unit of flow from $m$ in $G'\}$. For each $v$, if $v \in domain(m)$, then define $\phi(v) = m$.

DEFINITION 2.1. *A* light *monarch is one that has a domain of size* $< L$. *A* heavy *monarch is one that has unassigned vertices in its empire. A* full *monarch is one that is neither heavy nor light.*

LEMMA 2.2. *If $m$ is a heavy monarch, then each vertex in $m$'s domain belongs to $m$'s empire.*

*Proof.* Assume that there is a vertex $u$ in $m$'s domain that is not in $m$'s empire. Let $x$ be a vertex in $m$'s empire that is not assigned to any domain. We can change the flow function in $G'$ and send one unit of flow from $m$ to $x$ instead of $m$ to $u$. This produces a max flow in $G'$ of lower cost, which is a contradiction. ∎

If there is no heavy monarch, all vertices belong to a domain and the algorithm halts successfully.

Let $K_L$ be the number of light monarchs. Let $n_L$ be the number of vertices belonging to the domains of light monarchs, and let $n$ be the total number of vertices.

THEOREM 2.3. *The number of centers required is at least* $K_L + \lceil (n - n_L)/L \rceil$.

The proof is simpler than the proof given in [1]. The following lemmas were established in [1]. We repeat them for completeness.

Let $\mathcal{E}$ be the set of monarchs as defined in [1]. We repeat the definition here.

Let $\mathcal{E}_0$ be the set of light monarchs. Iteratively, add to $\mathcal{E}_0$ any monarch that contains a vertex in its domain that could have been assigned to a monarch in $\mathcal{E}_0$.

$$\mathcal{E}_j = \mathcal{E}_{j-1} \cup \{m \in M | \exists v \in V, \exists m' \in \mathcal{E}_{j-1},\ \phi(v) = m,\ \text{and } d(v, m') \leq 2 \text{ in } G_i\}.$$

Let $\mathcal{E}$ be the largest set $\mathcal{E}_j$ obtained in this process. Let $\mathcal{F}$ be the set of remaining monarchs.

LEMMA 2.4. *The set $\mathcal{E}$ does not contain any heavy monarchs.*

*Proof.* Suppose heavy monarch $\theta$ was added at iteration $j$. We can transfer a node $v$ to a center $\theta'$ in $\mathcal{E}_{j-1}$. By a sequence of such transfers, we eventually reach a center in $\mathcal{E}_0$ which has at most $L - 1$ nodes in its domain and can absorb the extra node. This corresponds to a higher flow, since the heavy monarch can absorb an unassigned node, which is a contradiction. ∎

LEMMA 2.5. *Consider a center in an optimal solution that covers a monarch in $\mathcal{E}$. This center cannot be assigned any nodes that are not in the domains of monarchs in $\mathcal{E}$.*

*Proof.* Assume for contradiction that $\theta$ is a center in the optimal solution that covers both $e \in \mathcal{E}$ and $u$. If $u$ does not belong to any domain, then, since the distance from $u$ to $e$ in $G_i$ is at most 2, we can perform a sequence of transfers, eventually reaching a center in $\mathcal{E}_0$ which has at most $L - 1$ nodes in its domain, and can absorb the extra node, resulting in a higher flow, which is a contradiction. If $u$ belongs to the domain of monarch $f$, then, since the distance from $e$ to $u$ in $G_i$ is at most 2, and the distance from $u$ to $f$ in $G_i$ is at most 2, it must be the case that $f \in \mathcal{E}$ as required by the lemma. ∎

*Proof of Theorem* 2.3. Each monarch in $\mathcal{E}$ is covered by a distinct center in the optimal solution, and these centers of the optimal solution cannot cover any other nodes in $\mathcal{F}$. Let $n_{\mathcal{E}}$ be the number of vertices in the domains of monarchs in $\mathcal{E}$. Then we need at least $|\mathcal{E}| + \lceil (\frac{n - n_{\mathcal{E}}}{L}) \rceil$ centers. This is the same as $K_L + \lceil (\frac{n + (|\mathcal{E}| - K_L) \cdot L - n_{\mathcal{E}}}{L}) \rceil$. Since $n_{\mathcal{E}} = (|\mathcal{E}| - K_L) \cdot L + n_L$, we get $K_L + \lceil (\frac{n - n_L}{L}) \rceil$. ∎

We will prove that this is also an upper bound on the number of centers we use. We now describe procedure REASSIGN.

Procedure REASSIGN (Appendix C) allocates new centers to cover the nodes left unassigned after the execution of ASSIGNDOMAINS. It works with one empire at a time, repeatedly selecting a leaf node from the tree of monarchs and then deleting that monarch from the tree. Each monarch will cover as many of the unassigned nodes in its empire as it can, with each new center covering $L$ unassigned nodes. This will leave $\ell < L$ nodes still unassigned. To ensure that nodes can only be passed up the tree once, we assign the $\ell$ remaining nodes to $m$, freeing up to $\ell$ nodes previously assigned to $m$. We then pass these nodes to the parent monarch of $m$.

In practice, one would pass to the parent monarch $p(m)$ those nodes which are closest to $p(m)$.

THEOREM 2.6. *Each vertex is assigned to a center whose distance in $G_i$ is at most* 5. *Moreover, we use at most $K_L + \lceil (n - n_L)/L \rceil$ centers.*

*Proof.* All centers except possibly light monarchs cover $L$ nodes by construction. The size of the domain of a light monarch does not decrease. Therefore the total number of centers used is at most $K_L + \lceil (n - n_L)/L \rceil$.

A node is either covered by the node from which it receives flow, or by the parent of its original monarch. In the former case, it is at distance at most 2 from the center that covers it. In the latter case the passed nodes are always covered by their monarch's parent, i.e., they are only passed once. Thus the distance from a node to the center that covers it is at most 5 (at most 2 to its original monarch, and 3 more to the parent monarch). ∎

## 3. Algorithm for $K$-centers.

We now consider the version where we are required to pick $K$ distinct vertices as centers. We use the same high-level approach as in the previous case but need to pick the centers carefully. We are able to show that the algorithm obtains an approximation factor of 6.

The main difficulty lies in allocating centers to cover the vertices left unassigned by ASSIGNDOMAINS. We first introduce some new notation.

Nodes in a monarch's empire are called its subjects. In ASSIGNMONARCHS, each level-2 subject $w$ of a monarch is brought in by a node $u$ at distance 1 from the monarch. We define $link(w) = u$. Each monarch $m$ (except the root) was placed into $Q$ by a unique level-2 subject of its parent monarch. This node is called the *spouse* $s(m)$ of monarch $m$ (Figure 2). Note that each monarch has a unique spouse, and a node can be the spouse of at most one monarch (by property 4).

Observe that obtaining a factor of 7 using the previous approach is easy: simply run the above algorithm and then "shift" all but one of the centers allocated at monarch $m$ to nodes in its empire (which are at distance at most 2 from $m$). Since each child monarch of $m$ is adjacent to a distinct node in $m$'s empire (its spouse), it is easy to see that there will be sufficiently many nodes in $m$'s empire to perform the shift. Obtaining the bound of 6 requires a more complex algorithm which is described next.

We need to be careful when allocating new centers to cover unassigned nodes. We require that: (1) a node can only be allocated as a center once; (2) monarchs have sufficient available nodes to allocate centers for the nodes passed to them. To ensure this we enforce the following rule. A monarch may allocate centers of the following types only:

1. nodes in its empire, or
2. nodes at distance 1 from itself (which may not be in its empire), as long as a monarch does not allocate its spouse as a center.

We define a *tree* $T(m)$ of height 2 corresponding to each monarch $m$. The root of $T(m)$ is monarch $m$. The leaves of this tree are all the level-2 subjects of $m$ that are the spouse of some other monarch. For any leaf $w$, we make $link(w)$ the parent of the leaf. These nodes are the children of $m$ in the tree $T(m)$.

In Figure 3 we show a monarch $m$ together with all the level-2 subjects of $m$ that are the spouse of some other monarch (for example, $m'$). For each leaf $w$, we also show $link(w)$. Notice that $link(w)$ may not be in monarch $m$'s empire.

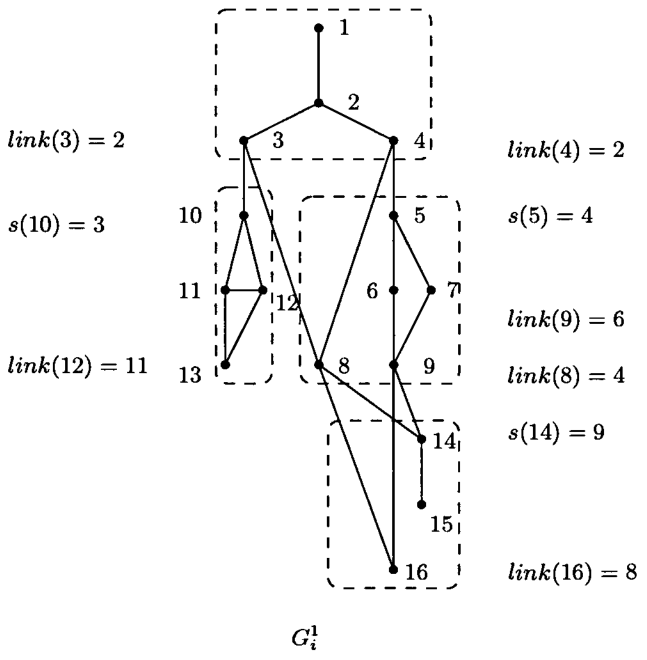


FIG. 2. *Example to illustrate links and spouses.*

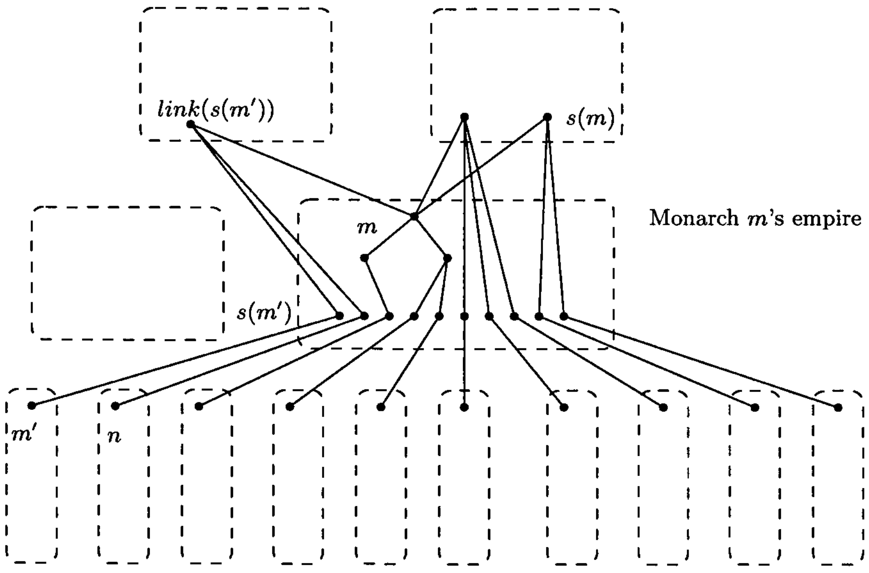


FIG. 3. *Example to illustrate tree $T(m)$ of a monarch $m$.*

Observe that nodes that are the spouse of some monarch may belong to two trees. We therefore specify that a monarch may assign a center to any vertex in its tree $T(m)$ other than its spouse. This ensures that no vertex is assigned as two centers by two different monarchs.

Tree $T(m)$ will be used in assigning vertices that are passed to monarch $m$. Nodes passed from monarch $m'$ to monarch $m$ are covered by one of five nodes: the spouse of monarch $m'$, i.e., $s(m')$, the spouse's link $link(s(m'))$, the spouse of a sibling monarch $s(n)$ (where $p(n) = p(m') = m$ and $link(s(n)) = link(s(m'))$), the link of a sibling monarch's spouse $link(s(n))$ (where $p(n) = p(m') = m$), or monarch $m$.

A monarch does not allocate centers at nodes that are passed to it. Because of this, we may have to allocate centers at vertices that are already assigned to a center. We therefore specify that in this case the new center does not cover itself but covers $L$ other vertices. A vertex allocated as a center which is not assigned to a center covers itself as well as $L - 1$ other vertices.

The algorithm given in this section differs from that in section 2 in the selection of new centers to cover the vertices left unassigned by AssignDomains. We give a high-level description of the new selection scheme below; pseudocode for the new ReAssign procedure that implements the scheme is given in Appendix D.

**High-level description.** We repeatedly select a leaf monarch in the tree of monarchs and allocate centers to cover nodes in its empire as well as nodes passed to it from its children monarchs. Let $m$ be the monarch currently under consideration. If $m'$ is a child monarch of $m$, we will assume that $m'$ passes the excess nodes in its empire to $m$. Each leaf node $s(m')$ in $T(m)$ is labeled with the number of excess nodes that monarch $m'$ is passing to $m$.

Nodes passed to $m$ are assigned to centers placed on nodes in $T(m)$. There are a few important things to note here. (1) When we begin to process monarch $m$, no centers are currently placed at any nodes in $T(m)$ (except for $m$ itself). (2) Monarch $m$ is responsible for allocating centers for all the nodes that are passed to it from its children monarchs. (3) Monarch $m$ is responsible for the free nodes in its empire. However, some of the free nodes at distance 2 from $m$ may belong to trees of other monarchs and may have centers already placed on them, in which case we will assume they are assigned to their own centers. If a vertex at distance 2 is in monarch $m$'s domain and a center is placed on it by the tree it belongs to, then it remains in $m$'s domain and does not change its assignment.

Monarch $m$ first assigns the nodes that are passed to it from children monarchs, using $T(m)$ to place full centers. Any nodes that are left over (at most $L - 1$) that were not assigned are assigned to monarch $m$, displacing vertices that were in $m$'s domain, which become unassigned. At this stage there may be many free nodes in $m$'s empire—nodes that were never part of a domain, as well as the nodes that were recently displaced from $m$'s domain. Note that the nodes that were never part of a domain do not have a center on them, while the ones that were displaced from $m$'s domain could have centers placed on them (since they may belong to other trees and may have been chosen as centers). However, there are at most $L - 1$ of these, so any which do not get assigned within $m$'s empire can be passed to $m$'s parent monarch. We now choose centers from the set of nodes that never belonged to any domain. In doing so, we may assign some of the displaced nodes as well. The remaining unassigned vertices, including the displaced nodes, are passed to $m$'s parent monarch.

We now describe in detail how the passed vertices are handled. Group the leaves of $T(m)$, placing leaves $u$ and $v$ in the same group iff $link(u) = link(v)$. We process the groups in turn, processing the group whose common link is $s(m)$ last, if such a group exists.

We assign passed nodes to centers by processing the groups in order. We process a leaf node in a group as follows: We start by adding the vertices passed to the leaf node to a list called *pending*. Whenever *pending* has at least $L$ vertices, we create a center and assign vertices to it. There are two things we have to be careful about: If we create a center at a vertex that is free, we have to assign the vertex to its own center. If a center is going to be assigned vertices from different groups, we move the center up one level, from a leaf node to the link node. Centers chosen in the last group are not assigned any vertices from other groups, and so we never assign a center at $s(m)$ but only at leaf nodes in this group.

To ensure that nodes in $T(m)$ do not have centers placed on them, we process the monarchs in $T$ in the reverse of the order in which they were placed in $T$. Note that if a node $v$ in $T(m)$ belongs to the empire of another monarch $m'$, then $m'$ must have been placed in $T$ before $m$, otherwise $m$ would have placed $v$ in its own empire. We thus process $m$ before $m'$. If a center is placed at $v$ by $m$, then $v$ is assigned to itself in case it was free. When we eventually process $m'$, we are guaranteed that if $v$ is free, it does not have a center placed on it.

For the last group, we proceed as above, except that any nodes carried over from the last group are picked up by monarch $m$, possibly replacing some nodes already assigned to $m$. These replaced nodes are either passed or allocated a center in $E_1(m) \cup E_2(m)$ by monarch $m$. (Note that if monarch $m$ is light, then the nodes are passed; if not, then it does not grab nodes from other empires, so it is safe to allocate centers at/for them.)

*Example.* We now discuss the example given in Figure 4 in detail. We process the leaves from left to right. Each leaf is labeled with the number of vertices that are passed to it from the corresponding child monarch. Assume that $L$ is 10. After we process $u_1$, *pending*$(m)$ has size 4, and no centers are allocated. When we process $u_2$, *pending*$(m)$ has size 12. Since we can allocate a full center at $u_2$, we do so. Since $u_2$ is free, we assign $u_2$ to itself and assign 9 $(= L - 1)$ vertices from *pending*$(m)$ to $u_2$. The size of *pending*$(m)$ is now 3. In processing $u_3$, we add two more vertices to *pending*$(m)$. Before we process the leaves in $v_2$'s group, we set $X = v_1$. Observe that vertices passed to nodes in $v_1$'s group are going to share a center with vertices passed to nodes in $v_2$'s group; hence we "promote" the center one level up. When we process $u_4$ and $u_5$, we add 3 more vertices to *pending*$(m)$, which now has size 8. We then process $u_6$, adding 8 vertices to *pending*$(m)$. Since we can allocate a full center, we allocate a center at $v_1$ (current value of $X$). Since $v_1$ is currently unassigned, we assign $v_1$ to itself and assign 9 $(= L - 1)$ vertices from *pending*$(m)$ to it. The size of *pending*$(m)$ is now 7. When we process $u_7$, we add 5 vertices to pending. Since we can allocate a full center, we create a center at $u_7$. Since $u_7$ is assigned, we assign 10 vertices from *pending*$(m)$ to it. This leaves 2 vertices in *pending*$(m)$ that are assigned to monarch $m$, possibly displacing other assigned vertices.

LEMMA 3.1. *Each node is allocated as a center at most once.*

*Proof.* A node $v$ may be allocated as a center either by its monarch, or by the monarch $m$ whose tree $T(m)$ it belongs to. If $v$'s monarch allocates $v$ as a center, then it must be the case that $v$ is unassigned, which means $v$ cannot be currently allocated as a center. If $m$ allocates $v$ as a center, then since $m$ was processed before $v$'s monarch, $v$ could not have already been allocated as a center.

If $v$ belongs to trees $T(m)$ and $T(m')$ for two different monarchs, then it must be the case that for one of the monarchs, say $m'$, $v = s(m')$. Then $m'$ will not allocate $v$ as a center. □

LEMMA 3.2. *Each monarch $m$ has sufficient available nodes in its tree $T(m)$ to allocate centers for nodes passed to it.*

*Proof.* Each monarch passes at most $L - 1$ nodes to its parent. If monarch $m$ has

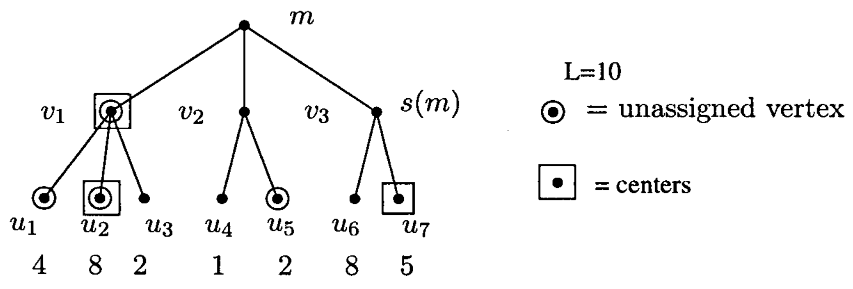


FIG. 4. *Example to illustrate assignment of centers in tree $T(m)$.*

$N$ children monarchs, then it must be the case that each child monarch has a unique spouse in $T(m)$. In addition, all these spouses are level-2 nodes in $T(m)$, so they are all available to allocate as centers. ☐

LEMMA 3.3. *Every node is assigned to a center.*

*Proof.* All nodes passed to a monarch $m$ are assigned centers from $T(m) \cup \{m\}$. Unassigned nodes in $m$'s empire are either assigned a center or passed. There are at most $L-1$ nodes displaced from $domain(m)$, hence they are either allocated a center from $m$'s empire or passed to $m$'s parent. ☐

THEOREM 3.4. *Each vertex is assigned to a center whose distance in $G_i$ is at most* 6. *Moreover, we use at most $K_L + \lceil (n - n_L)/L \rceil$ centers.*

*Proof.* All centers except possibly light monarchs cover $L$ nodes by construction. The size of the domain of a light monarch does not decrease. Therefore the total number of centers used is at most $K_L + \lceil (n - n_L)/L \rceil$.

A node which is not passed is covered either by the monarch from which it receives flow or by a node in its monarch's empire. In the former case, it is at distance at most 2 from the center that covers it, and in the latter case it is at distance at most 4 from the center that covers it. A node which is passed from monarch $m'$ to monarch $m$ is covered by one of the following: (1) $s(m')$, (2) $link(s(m'))$, (3) $s(n)$, where $p(n) = p(m') = m$ and $link(s(n)) = link(s(m'))$, (4) $link(s(n))$, where $p(n) = p(m') = m$, or (5) monarch $m$. The distance bounds are as follows.

| Case | Distance |
|---|---|
| Case (1) | $\leq 3$ |
| Case (2) | $\leq 4$ |
| Case (3) | $\leq 5$ |
| Case (4) | $\leq 6$ |
| Case (5) | $\leq 5$ |

☐

In Figure 5 we give an example showing that the factor of 6 is tight for our algorithm. All edges in the example are edges in $G_i$. In the example, monarchs $m_2$ and $m_3$ pass 2 and 3 nodes, respectively, to their parent $m_1$. Monarch $m_2$ passes itself and $v_4$, and monarch $m_3$ passes itself, $v_5$, and $v_6$. The algorithm assigns all 5 passed nodes to a center which it places at $v_1$, leaving $v_6$ at a distance of 6 from its center. Vertex $v_3$ absorbs the remaining nodes in $m_1$'s empire. It is clear, however, that OPT covers all nodes within distance 1.

**Running time.** The bottleneck in the running time of this algorithm is the flow computation. If we use binary search in CAPACITATED-CENTERS, the algorithm computes $O(\log n)$ maximum flows.

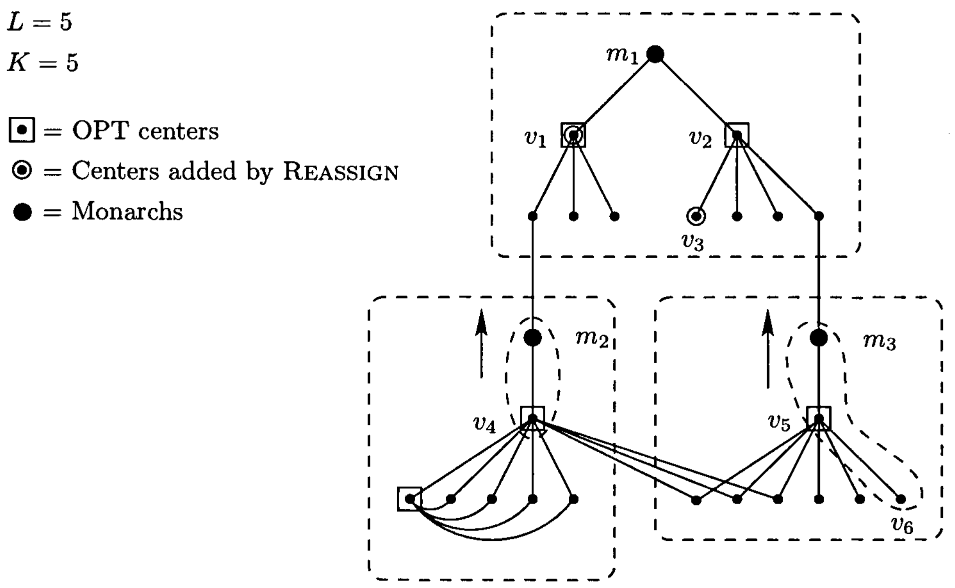


FIG. 5. *Example to show that the factor of* 6 *is tight.*

**3.1. Capacitated centers with costs.** The capacitated $K$-center problem with costs is a generalization of the capacitated $K$-center problem where a cost function is defined on the vertices and the objective is to pick a set of centers whose total cost is at most $K$, such that the radius is minimized. (Note that this is equivalent to the weighted capacitated $K$-center problem in [1]. We use *cost* here to distinguish from weights as defined in, for example, [3, 17].) More formally, we are given a cost function $c : V \to \mathcal{R}^+$, and we add the constraint $\sum_{v \in S} c(v) \leq K$ to the statement of the capacitated $K$-center problem.

Bar-Ilan, Kortsarz, and Peleg gave the first polynomial time approximation algorithm for this problem with an approximation factor of 21. Their technique, which involves finding a minimum-cost perfect matching in a bipartite graph, generalizes to finding a $2\rho + 1$ solution given a $\rho$-approximation algorithm for the capacitated $K$-centers problem. It therefore yields an approximation algorithm with an approximation factor of 13 when combined with our algorithm for capacitated $K$-centers.

**4. Remarks.** It is possible to improve the quality of the approximation if one is willing to allow some slack on the number of centers used and the maximum load. Let a $(c_1K, c_2L, c_3R)$ solution denote a solution using at most $c_1K$ centers, each with a load of at most $c_2L$, which assigns every node to a center at distance at most $c_3R$, where $R$ is the radius of the optimal solution. Thus the algorithms given above obtain a $(K, L, 5R)$ solution to the capacitated multi-$K$-center problem and a $(K, L, 6R)$ solution to the capacitated $K$-center problem.

For the capacitated multi-$K$-center problem, we can obtain for any $x \geq 1$ a $\left(\frac{2}{c}K, cL, 2R\right)$ solution, where $c = \frac{x+1}{x}$. For example, when $x = 1$, this gives a $(K, 2L, 2R)$ solution. We can also achieve a $(2K, L, 2R)$ solution to this problem by always allocating $\left\lceil \frac{unassigned(m)}{L} \right\rceil$ extra centers in each empire.

The algorithm works as follows. We modify REASSIGN to overload centers in some empires and allocate extra centers in others. Specifically, let $unassigned(m) = k'L + \ell$.

If $\ell \leq \frac{L}{x}$, then allocate $k'$ additional centers at monarch $m$ and use the centers at $m$ to cover all nodes in *unassigned*$(m)$. Clearly, no center at monarch $m$ has to cover more than $\frac{x+1}{x}L$ nodes in this case. If $\ell > \frac{L}{x}$, then allocate $k'+1$ additional centers at monarch $m$ and use them to cover all nodes in *unassigned*$(m)$. This may cause us to use more than $K$ centers, since we allocate new lightly loaded centers at the end. We show below that we use at most $\frac{2x}{x+1}K$ centers.

THEOREM 4.1. *For any $x \geq 1$, the algorithm gives a $\left(\frac{2}{c}K, cL, 2R\right)$ solution, where $c = \frac{x+1}{x}$.*

Let the sets $\mathcal{E}$ and $\mathcal{F}$ be defined as before. Let $\mathcal{X}$ be the set of monarchs at which an extra center was allocated to cover $\ell > \frac{L}{x}$ nodes. Let $C_1$ and $C_2$ be the number of centers used in the optimal solution to cover nodes in empires in $\mathcal{E}$ and $\mathcal{F}$, respectively. Clearly, any extra centers we use must be allocated at monarchs in $\mathcal{F}$. Let $\mathcal{S_L}$ and $\mathcal{S_F}$ denote the sets of nodes in the domains of monarchs in $\mathcal{E}$ and $\mathcal{F}$, respectively. Let $f$ be the number of full centers allocated by the algorithm that cover nodes in $\mathcal{S_F}$.

LEMMA 4.2. *The number of centers allocated by our algorithm is at most $C_1 + f + |\mathcal{X}|$.*

*Proof.* No additional centers are used by the algorithm to cover nodes in $\mathcal{S_L}$. Therefore the algorithm uses $|\mathcal{E}| + f + |\mathcal{X}|$ centers. Because the monarchs form an independent set in $G_i^2$, the optimal solution must use at least $|\mathcal{E}|$ centers to cover nodes in $\mathcal{S_L}$. Therefore $C_1 \geq |\mathcal{E}|$, implying the lemma. □

LEMMA 4.3. *A lower bound on $C_2$ is given by $C_2 \geq f + \frac{|\mathcal{X}|}{x}$.*

*Proof.* By Lemma 2.5, the centers in the optimal solution which cover nodes in $\mathcal{S_L}$ cannot cover any nodes not in $\mathcal{S_L}$. Therefore the optimal solution uses at least $\frac{|\mathcal{S_F}|}{L}$ centers to cover nodes in $\mathcal{S_F}$. Since the lightly loaded centers allocated by our algorithm to cover nodes in $\mathcal{S_F}$ each cover at least $\frac{L}{x}$ nodes, it follows that $fL + \frac{|\mathcal{X}|L}{x} \leq |\mathcal{S_F}|$. Therefore $C_2 \geq \frac{|\mathcal{S_F}|}{L} \geq f + \frac{|\mathcal{X}|}{x}$. □

*Proof of Theorem* 4.1. By Lemma 4.2, the number of centers used by our algorithm is at most

$$\begin{aligned}
C_1 + f + |\mathcal{X}| &\leq C_1 + \frac{1+x}{1+x}f + \frac{1+x}{1+x}|\mathcal{X}| \\
&\leq C_1 + \frac{1+x}{1+x}f + \frac{x-1}{1+x}|\mathcal{X}| + \frac{2}{1+x}|\mathcal{X}| \\
&\leq C_1 + \frac{1+x}{1+x}f + \frac{x-1}{1+x}f + \frac{2}{1+x}|\mathcal{X}| \qquad \text{(since } |\mathcal{X}| \leq f\text{)} \\
&\leq C_1 + \frac{2x}{1+x}\left(f + \frac{|\mathcal{X}|}{x}\right) \\
&\leq C_1 + \frac{2x}{1+x}C_2 \qquad \text{(by Lemma 4.3)} \\
&\leq \frac{2x}{x+1}(C_1 + C_2) \qquad \text{(since } x \geq 1\text{)} \\
&\leq \frac{2x}{x+1}K \qquad \text{(because the optimal solution does not overuse centers).}
\end{aligned}$$
□

For the capacitated $K$-center problem, the same approach gives a $\left(\frac{2}{c}K, cL, 4R\right)$ solution.

Results of Lund and Yannakakis [12] and Feige [6] imply that no polynomial time $(c_1K, c_2L, (2-\epsilon)R)$ approximation algorithm is possible unless $P = NP$, since this would imply a constant-factor approximation algorithm for set cover. A sketch of the reduction is as follows: given an instance of set cover, construct a bipartite graph with nodes corresponding to the sets in one bipartition and nodes corresponding to the elements in the other. There is an edge between each set-element node pair such that the element appears in the set. Add two nodes $v$ and $v'$; $v$ has an edge to every node corresponding to a set, and there is an edge between $v$ and $v'$. All edge lengths are 1. Set $L =$ the number of nodes. Now any solution of radius less than $2R$ using $K'$ centers corresponds to a solution to the set cover problem using $K' - 1$ sets.

## Appendix A. Pseudocode for AssignMonarchs.

```
AssignMonarchs($G_i^c$).
1.  Pick an arbitrary vertex $v \in G_i^c$ and set $Q = \{v\}$.
2.  $p(v) =$ nil.
3.  **while** $Q$ has unmarked nodes **do**
4.          Remove an unmarked node $v$ from $Q$.
5.          Make $v$ a monarch and mark it.
6.          **for** all $u \in Adj(v)$ **do**
7.              **if** $u$ is unmarked **then**
8.                  Add $u$ to $E_1(v)$ and mark $u$
9.          **for** all $u \in Adj(v)$ **do**
10.             **for** all $w \in Adj(u)$ **do**
11.                 **if** $w$ is unmarked **then**
12.                     Add $w$ to $E_2(v)$ and mark $w$
13.         **for** all $u \in E_2(v)$ **do**
14.             **for** all $w \in Adj(u)$ **do**
15.                 **if** $w$ is unmarked **and** $w \notin Q$ **then**
16.                     Set $p(w) = v$ and add $w$ to $Q$
17. **end-proc**
```

## Appendix B. Pseudocode for AssignDomains.

```
AssignDomains($G_i^c$).
1. Construct a bipartite graph $G' = (M, V, E')$, where $M$ is the set of monarchs in
   $G_i^c$ and $E' = \{(m, v) \mid m \in M,\ v \in V$, distance from $m$ to $v$ is at most two$\}$.
2. Add vertices $s$ and $t$. Add edges $\{(s, m) \mid m \in M\}$ and $\{(v, t) \mid v \in V\}$.
3. For $m \in M$, $v \in V$, set capacities $u(s, m) = L$, $u(m, v) = 1$ and $u(v, t) = 1$.
4. Cost of edge $c(m, v) = 1$ if $v$ is not in $m$'s empire.
   Cost of all other edges is 0.
5. Compute a min-cost maximum integral flow in $G'$.
6. For each monarch $m$, set
   $domain(m) = \{v \mid v$ receives one unit of flow from $m$ in $G'\}$.
7. For each $v$, if $v \in domain(m)$ then define $\phi(v) = m$.
8. **end-proc**
```

## Appendix C. Pseudocode for first version of ReAssign.

```
ReAssign($G_i^c$).
1. Let $M$ be the set of monarchs in $G_i^c$.
2. For each monarch $m \in M$, set
   $unassigned(m) = (\{m\} \cup E_1(m) \cup E_2(m)) \setminus (\cup_{u \in M} domain(u))$.
3. Let $T$ be the tree of monarchs in $G_i^c$.
4. **for** all nodes $m$ in $T$, set $passed(m) = \emptyset$.
5. **while** $T$ is not empty **do**
```

```
6.         Remove a leaf node $m$ from $T$.
7.         Let $|unassigned(m)| + |passed(m)| = k'L + \ell$, $0 \le \ell < L$.
8.         Allocate $k'$ new centers at monarch $m$ and assign $k'L$ of the nodes
           to them. For each such node $v$ we define $\phi(v) = m$.
9.         Assign the $\ell$ remaining nodes to monarch $m$ and for each such
           node $v$ define $\phi(v) = m$, freeing up to $\ell$ nodes in $domain(m)$.
10.        Add the freed nodes to $passed(p(m))$.
11.        Delete $m$ from $T$.
12. end-proc
```

**Appendix D. Pseudocode for second version of ReAssign.**

```
ReAssign($G_i^c$).
1. Let $M$ be the set of monarchs in $G_i^c$.
2. For each monarch $m \in M$, set
   $unassigned(m) = (\{m\} \cup E_1(m) \cup E_2(m)) \setminus (\cup_{u \in M} domain(u))$.
3. For each monarch $m \in M$, let $pending(m)$ be an ordered list, initially $\emptyset$.
4. Let $T$ be the tree of monarchs in $G_i^c$.
5. for all nodes $m$ in $T$ do
6.     Define $T(m)$ such that the group containing $s(m)$ comes last.
7.     for all leaf nodes $v$ in $T(m))$ do set $passed(v) = \emptyset$.
8. for all nodes $v$ in $G_i^c$, let $\chi(v) = \{v\}$ iff $v$ is unassigned and $\emptyset$ otherwise.
9. for all nodes $m$ in $T$, process them in reverse order of their insertion into $T$:
10.    Set $X =$ null.
11.    for all children $v$ of $m$ in $T(m)$ do
12.        for all children $u$ of $v$ in $T(m)$ do
13.            Append $passed(u)$ to $pending(m)$.
14.            if $X =$ null then set $X = u$.
15.            if $|\chi(X)| + |pending(m)| \ge L$ then
16.                Allocate a center at $X$ and assign $\chi(X)$ to it.
17.                Assign the first $L - |\chi(X)|$ nodes from $pending(m)$ to $X$ and
                   remove them from $pending(m)$.
18.                Set $X =$ null.
19.            else if $X = u$ then set $X =$ null.
20.        if $v \ne s(m)$ and $X =$ null then
21.            if $|\chi(v)| + |pending(m)| = L$ then
22.                Allocate a center at $v$ and assign $\chi(v)$ to it.
23.                Remove all nodes from $pending(m)$ and assign them to $v$.
24.            else set $X = v$.
25.    Let $displaced(m) = |domain(m)| + |pending(m)| - L$.
26.    Assign all nodes in $pending(m)$ to $m$,
       possibly displacing nodes in $domain(m)$.
27.    Let $|unassigned(m)| + displaced(m) = k'L + \ell$.
28.    Allocate $k'$ new centers at nodes in $unassigned(m)$.
29.    Assign $k'L$ of the nodes to them, assigning nodes in $unassigned(m)$ first.
30.    Add the $\ell$ remaining nodes to $passed(s(m))$.
31.    Delete $m$ from $T$.
32. end-proc
```

**Acknowledgments.** We thank Robert Pless and Balaji Raghavachari for useful discussions. We thank the ESA referee for helpful comments.

## REFERENCES

[1] J. Bar-Ilan, G. Kortsarz, and D. Peleg, *How to allocate network centers*, J. Algorithms, 15 (1993), pp. 385–415.

[2] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1989.

[3] M. Dyer and A. M. Frieze, *A simple heuristic for the p-center problem*, Oper. Res. Lett., 3 (1985), pp. 285–288.

[4] J. Edmonds and D. R. Fulkerson, *Bottleneck extrema*, J. Combin. Theory, 8 (1970), pp. 299–306.

[5] T. Feder and D. H. Greene, *Optimal algorithms for approximate clustering*, in Proceedings of the 20th ACM Symposium on the Theory of Computing, Chicago, IL, 1988, pp. 434–444.

[6] U. Feige, *A threshold of* $\ln n$ *for approximating set cover*, in Proceedings of the 28th ACM Symposium on the Theory of Computing, Philadelphia, PA, 1996, pp. 314–318.

[7] T. Gonzalez, *Clustering to minimize the maximum inter-cluster distance*, Theoret. Comput. Sci., 38 (1985), pp. 293–306.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1978.

[9] D. Hochbaum and D. B. Shmoys, *A best possible heuristic for the k-center problem*, Math. Oper. Res., 10 (1985), pp. 180–184.

[10] D. Hochbaum and D. B. Shmoys, *A unified approach to approximation algorithms for bottleneck problems*, J. ACM, 33 (1986), pp. 533–550.

[11] W. L. Hsu and G. L. Nemhauser, *Easy and hard bottleneck location problems*, Discrete Appl. Math., 1 (1979), pp. 209–216.

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

[13] R. Lupton, F. M. Maley, and N. E. Young, *Data collection for the Sloan digital sky survey—A network-flow heuristic*, in Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, Atlanta, GA, SIAM, Philadelphia, PA, 1996, pp. 296–303.

[14] H. L. Morgan and K. D. Levin, *Optimal program and data locations in computer networks*, Communications of the ACM, 20 (1977), pp. 315–322.

[15] K. Murthy and J. Kam, *An approximation algorithm to the file allocation problem in computer networks*, in Proceedings of 2nd ACM Symposium on Principles of Database Systems, Atlanta, GA, 1983, pp. 258–266.

[16] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[17] J. Plesnik, *A heuristic for the p-center problem in graphs*, Discrete Appl. Math., 17 (1987), pp. 263–268.

[18] C. Toregas, R. Swain, C. Revelle, and L. Bergman, *The location of emergency service facilities*, Oper. Res., 19 (1971), pp. 1363–1373.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 4, pp. 419–435

# SUBSET-RESTRICTED INTERCHANGE FOR DYNAMIC MIN-MAX SCHEDULING PROBLEMS*

GEORGE STEINER† AND PAUL STEPHENSON†‡

**Abstract.** The traditional method of pairwise job interchange compares the cost of sequences which differ only in the interchange of two jobs. It assumes that either there are no intermediate jobs (*adjacent pairwise interchange*) or that the interchange can be performed no matter what the intermediate jobs are (*nonadjacent pairwise interchange*). We introduce a generalization that permits the pairwise interchange of jobs *provided* that the intermediate jobs belong to a restricted subset of jobs (*subset-restricted pairwise interchange*).

In general, even if an adjacent interchange relation is a partial order, it need not be a precedence order. We introduce a unified theory of dominance relations based on subset-restricted interchange. This yields a precedence order for the class of unconstrained, regular, single machine scheduling problems $1\,/r_j\,/f_{\max}$. Thus it applies to $1\,/r_j\,/L_{\max}$, $1/r_j, \overline{d_j}/C_{\max}$, $1\,/r_j\,/\max w_j L_j$, $1\,/r_j\,/\max w_j C_j$, and other problems. We also show that these problems remain strongly NP-hard for interval-ordered tasks. The strength of the precedence orders was tested in a large-scale computational experiment for $1\,/r_j\,/\max w_j C_j$. The results show that using the precedence orders in branch and bound algorithms speeds these up on average by about 58 percent.

**Key words.** scheduling, dominance relations, posets

**AMS subject classifications.** 06A06, 06A07, 90B36

**PII.** S0895480198343418

## 1. Introduction.

Interchange arguments, which compare the cost of sequences that differ only in the interchange of two jobs, are quite common in the scheduling literature. Some classic results that established the technique are the *rules* of Johnson [10], Smith [20], and Jackson [9]. If jobs $i$ and $j$ can be interchanged so that $i$ is before $j$ in an optimal sequence, then we will say that $i$ is *preferred* to $j$. If these preference relations are transitive, then we have a *preference order* on the jobs. The above-mentioned rules are examples of problems where the *preference order* for the adjacent interchange of jobs (the *adjacent interchange order*) is a complete order (i.e., a sequence). Johnson's rule establishes such a sequence for the two-machine maximum completion time flow shop problem, whereas the rules of Smith and Jackson state the optimality of the *weighted shortest processing time* and *earliest due date* sequences for $1\,//\sum w_i C_i$ and $1\,//L_{\max}$, respectively. (We use the standard notation to describe scheduling problems and we refer the reader to [13] or [17] for any terminology not defined here.) Another classic interchange result due to Emmons [2] provides a *precedence order* (a partial ordering of the jobs that must be obeyed by at least one optimal sequence) for the total tardiness problem $1\,//\sum T_i$ . In this case, there is a preference order for the interchange of jobs that permits the interchange of any (not necessarily adjacent) pair of jobs in a sequence if they do not appear in preference order. By performing these interchanges in a particular manner, we can restrict our search space to contain only sequences that obey the preference order.

*Received by the editors March 15, 1998; accepted for publication (in revised form) July 5, 2000; published electronically October 6, 2000. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada under grant OGP0001798.

http://www.siam.org/journals/sidma/13-4/34341.html

†Management Science and Information Systems Area, Michael G. DeGroote School of Business, McMaster University, Hamilton, ON, Canada L8S 4M4 (steiner@mcmaster.ca).

‡Current address: Department of Mathematics, Acadia University, Wolfville, NS, Canada B0P IX0 (paul.stephenson@acadiau.ca).

In this paper, we introduce a new technique, a generalization of pairwise interchange that takes into consideration the composition of the intermediate sequence. This yields a preference order which permits the interchange of a pair of jobs *provided* that the intermediate jobs belong to a restricted subset. The traditional methods of pairwise job interchange can be viewed as special cases of this *subset-restricted interchange,* where the subsets are either uniformly the empty set (adjacent interchange) or the entire job set (nonadjacent interchange). In general, an adjacent interchange order is not a complete order and therefore it is *not* a precedence order, as we shall see for the $1/r_j/L_{\max}$ problem, using an example due to Lageweg, Lenstra, and Rinnooy Kan [11]. We prove, however, that using subset-restricted interchange for the class of regular, single machine scheduling problems $1/r_j/f_{\max}$, we can derive a precedence order that is a suborder of the adjacent interchange order. A performance measure for a scheduling problem is called *regular* if it is a nondecreasing function of the job completion times. In such problems, each job $i$ has an associated nondecreasing, real valued cost function $f_i(t)$, the cost of completing $i$ at time $t$, and the objective is to minimize the maximum cost $f_{\max}$. The precedence orders we derive have the property that they are defined *independently* of the processing times. This makes them especially useful in applications with stochastic or ill-defined processing times. We use only certain extreme values of the other job parameters that display a "staircase-like" structure. In addition, the precedence orders derived belong to a special class of partial orders, the interval orders [5]. This also leads to the complexity implication that the above problems are *strongly NP-hard* for interval-ordered tasks. Our results can be viewed as a *unified treatment* of job interchange, which *generalizes* well-known rules for deriving dominance relations and *extends* the pyramid precedence orders of Erschler et al. [3], [4] for $1/r_j/L_{\max}$ to the general problem $1/r_j/f_{\max}$. We generalize the pairwise interchange and insertion operations of Monma [15], and introduce "pyramid-like" structures of higher dimension than two, extending the two-dimensional staircases of [3] and [4]. In our unified theory, $1/r_j/L_{\max}$ represents the most special case.

The paper is organized as follows. In the next section, we introduce the preliminary definitions and notation for sequencing problems, partial orders, and interchange operators. In section 3, we define the adjacent interchange order $\lessdot$ and the interchange regions for $1/r_j/f_{\max}$. In section 4, we derive a precedence order $\prec$ for the problem $1/r_j/f_{\max}$, which includes the problems $1/r_j/L_{\max}$, $1/r_j,\overline{d_j}/C_{\max}$, $1/r_j/\max w_jC_j$, and $1/r_j/\max w_jL_j$ as special cases. Finally, in section 5 we discuss the results of a computational experiment for the problem $1/r_j/\max w_jC_j$.

## 2. Preliminary definitions and notation.

**2.1. Sequencing notation.** We call a scheduling problem a sequencing problem if any schedule can be completely specified by the sequence in which jobs are performed. This is the case for nonpreemptively scheduling a single machine with a regular performance measure. Let $J = \{1, 2, \ldots, n\}$ be the set of jobs to be sequenced on a single machine. Jobs are characterized by a list of parameters (e.g., for the $1/r_j/L_{\max}$ problem each job $j$ possesses a release time $r_j \geq 0$, a processing time $p_j \geq 0$, and a due date $d_j \geq 0$, the lateness for job $j$ is $L_j = C_j - d_j$, where $C_j$ is its completion time). A sequence $s$ on $J$ is a function from $\{1, 2, \ldots, n\}$ to $J$ represented by the $n$-tuple $(s(1), s(2), \ldots, s(n))$, where $s(i)$ is the $i$th job in sequence $s$ (e.g., for the maximum lateness problem $L_{\max}(s) = \max_i(C_{s(i)} - d_{s(i)})$). For the sequencing problems that we study, the adjacent interchange order will be a partial order defined by the parameters of the jobs. Thus we introduce certain definitions for partially ordered sets (posets).

**2.2. Partial orders.** By a partially ordered set we mean a pair $P = (X, \leq_P)$ consisting of a set $X$ together with a binary relation $\leq_P$ on $X \times X$ which is reflexive, antisymmetric, and transitive. For $u, v \in X$, $u \leq_P v$ is interpreted as $u$ is less than or equal to $v$ in $P$. Similarly, $u <_P v$ means that $u \leq_P v$ and $u \neq v$. The usual symbols $\leq$ and $<$ will be reserved for relations between real numbers. A partial order $P = (X, \leq_P)$ is a *linear order* (or *complete order*) if for every pair $(u, v) \in X \times X$, either $u \leq_P v$ or $v \leq_P u$. Given a pair of partial orders $P = (X, \leq_P)$ and $Q = (X, \leq_Q)$ on the same set $X$, we call $Q$ an *extension* of $P$ ($P$ a *suborder* of $Q$) if $u \leq_P v$ implies $u \leq_Q v$ for all $u, v \in X$. A partial order $Q = (X, \leq_Q)$ is a *linear extension* of a partial order $P = (X, \leq_P)$ if $Q$ is a linear order that extends $P$. Given two partial orders $P_1 = (X, \leq_{P_1})$ and $P_2 = (X, \leq_{P_2})$, we can define the partial order $P_1 \cap P_2 = (X, \leq_{P_1 \cap P_2})$, the *intersection* of $P_1$ and $P_2$, where $u \leq_{P_1 \cap P_2} v$ if and only if $u \leq_{P_1} v$ and $u \leq_{P_2} v$ for all $u, v \in X$. The *dimension* of a partial order $P = (X, \leq_P)$, denoted by $\dim(P)$, is the smallest $l$ such that there exists a set $\{Q_1, Q_2, \ldots, Q_l\}$ of linear extensions of $P$ such that $P = \cap_{i=1}^{l} Q_i$. A subset $I \subseteq X$ is an *ideal* of $P$ if for every $v \in I$ and $u \in X$ such that $u \leq_P v$ we have $u \in I$. Similarly, $F \subseteq X$ is a *filter* of $P$ if for every $u \in F$ and $v \in X$ such that $u \leq_P v$ we have $v \in F$. For every $v \in X$ the *principal ideal* $I(v)$ is defined by $I(v) = \{u \in X \,|u \leq_P v\}$ and the *principal filter* $F(v)$ is defined by $F(v) = \{u \in X \,|v \leq_P u\}$.

**2.3. Interchange operators.** We follow Monma [15] in defining our interchange operators. Let $s_1$ be a sequence with job $m$ preceding job $k$. In general, $s_1$ is of the form $s_1 = (AmBkC)$, where $A$, $B$, and $C$ are subsequences of $J$. We define three types of interchanges of jobs $k$ and $m$ that leave $k$ preceding $m$ in the resulting sequence $s_2$.

1. *Pairwise interchange* (*PI*):
   $s_2 = (AkBmC)$,
2. *Backward insertion* (*BI*):
   $s_2 = (ABkmC)$,
3. *Forward insertion* (*FI*):
   $s_2 = (AkmBC)$.

If we let $B$ be the set of jobs in sequence $B$ (we do not distinguish between these), then each of these interchanges reduces to adjacent pairwise interchange in the case when $B = \varnothing$. This leads to the definition of the *adjacent interchange order.*

Definition 2.1. *A partial order $\lessdot$ is an* adjacent interchange order *for a sequencing function $f$ if for all jobs $k$, $m$, and sequences $A$, $C$, $k \lessdot m$ implies $f(AkmC) \leq f(AmkC)$.*

Note that all of the above interchanges involve interchanging $k$, $m$ or both $k$ and $m$ around sequence $B$. Intuitively, whether or not an interchange leads to a reduction in cost (for a given sequencing function $f$ and adjacent interchange order $\lessdot$) should depend on the composition of $B$**.** This involves placing restrictions on certain of the parameters of the jobs in $B$. In the case when interchangeability does not depend on the composition of $B$, then $\lessdot$ is a *precedence order* for the sequencing problem, i.e., there exists an optimal sequence that is a linear extension of $\lessdot$. Such an example is the precedence order $\lessdot$ defined by

$$k \lessdot m \Leftrightarrow p_k \leq p_m \text{ and } d_k \leq d_m$$

for the total tardiness problem on a single machine $1//\sum T_i$ [2]. Note that here $\lessdot$ is the intersection of the $\leq_p$ and $\leq_d$ orders. In general, an adjacent interchange order

$\lessdot$ is not necessarily a precedence order, as it can be demonstrated by the instance of the maximum lateness problem $1/r_j/L_{\max}$, shown in its equivalent delivery time form in Example 1 [11]. The delivery time version is defined by triples $(r_j, p_j, q_j)$, where $q_j = T - d_j$ is the delivery time for job $j \in J$ and $T$ is a constant chosen so that $T \geq \max\{d_j \,|j \in J\}$. If we define $L_j^{'} = C_j + q_j$, then $L_j^{'} = C_j + T - d_j = L_j + T$ and $L_{\max}^{'} = L_{\max} + T$. As we shall see in the next section, the adjacent interchange order $\lessdot$ for $1/r_j/L_{\max}$ is defined by $k \lessdot m \Leftrightarrow r_k \leq r_m$ and $q_k \geq q_m$. For the 5-job example specified below we have $4 \lessdot 2$; however, the unique optimal sequence is $(1, 2, 3, 4, 5)$ with $L_{\max}^{'} = 11$. Thus $\lessdot$ is *not* a precedence order, since the unique optimal sequence is not a linear extension of $\lessdot$.

*Example* 1. A 5-job problem to illustrate that $\lessdot$ is not necessarily a precedence order.

| $j$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $r_j$ | 0 | 2 | 3 | 0 | 7 |
| $p_j$ | 2 | 1 | 2 | 2 | 2 |
| $q_j$ | 5 | 2 | 6 | 3 | 2 |

We consider interchanges that are restricted by conditions on $B$ and define the subset-restricted interchange conditions as follows.

DEFINITION 2.2. *An adjacent interchange order $\lessdot$ together with the collection of subsets $R^{PI} = \{R^{PI}_{k \lessdot m} \,|k \lessdot m\}$ satisfies the* restricted pairwise interchange condition *for a sequencing function $f$ if for all jobs $k$, $m$ and sequences $A$, $B$, $C$, $k \lessdot m$ and $B \subseteq R^{PI}_{k \lessdot m}$ imply $f(AkBmC) \leq f(AmBkC)$.*

DEFINITION 2.3. *An adjacent interchange order $\lessdot$ together with the collection of subsets $R^{BI} = \{R^{BI}_{k \lessdot m} \,|k \lessdot m\}$ satisfies the* restricted backward insertion condition *for a sequencing function $f$ if for all jobs $k$, $m$ and sequences $A$, $B$, $C$, $k \lessdot m$ and $B \subseteq R^{BI}_{k \lessdot m}$ imply $f(ABkmC) \leq f(AmBkC)$.*

DEFINITION 2.4. *An adjacent interchange order $\lessdot$ together with the collection of subsets $R^{FI} = \{R^{FI}_{k \lessdot m} \,|k \lessdot m\}$ satisfies the* restricted forward insertion condition *for a sequencing function $f$ if for all jobs $k$, $m$ and sequences $A$, $B$, $C$, $k \lessdot m$ and $B \subseteq R^{FI}_{k \lessdot m}$ imply $f(AkmBC) \leq f(AmBkC)$.*

In the following sections, we show how to use subset-restricted interchange and the above three conditions to derive a precedence order $\prec$ on the jobs. This precedence order $\prec$ is always a suborder of the adjacent interchange order $\lessdot$.

## 3. Interchange regions.

In this section, we derive the interchange regions (subsets) for the general problem $1/r_j/f_{\max}$. In this problem, each job $j$ has an associated *nondecreasing,* real valued cost function $f_j$, where $f_j(t)$ is the cost of completing job $j$ at time $t$, and the objective is to minimize $f_{\max} = \max_{1 \leq j \leq n} f_j(C_j)$ over all sequences. We order the jobs according to $\geq_f$, where $f_i \geq_f f_j \Leftrightarrow f_i(t) \geq f_j(t)$ for all $t \geq 0$. Note that, in the general case, $\geq_f$ does not order every pair $i$ and $j$, it might be only a partial order. Two special, linearly ordered cases of $\geq_f$ occur for the lateness objective, where $f_j(t) = t + q_j$, and the weighted completion time objective, where $f_j(t) = w_j t$. Hall [8] considered the $\geq_f$ order and noticed that the linear ordering property makes it possible to extend Potts's [18] approximation algorithm for the $1/r_j/L_{\max}$ problem to the $1/r_j/f_{\max}$ problem when $\geq_f$ is a linear order.

The adjacent interchange order and the restricted subsets for $1/r_j/f_{\max}$ are defined below. We note that these definitions use *no processing time information.* This means that all of the subsequent results are true *irrespective* of job processing times.

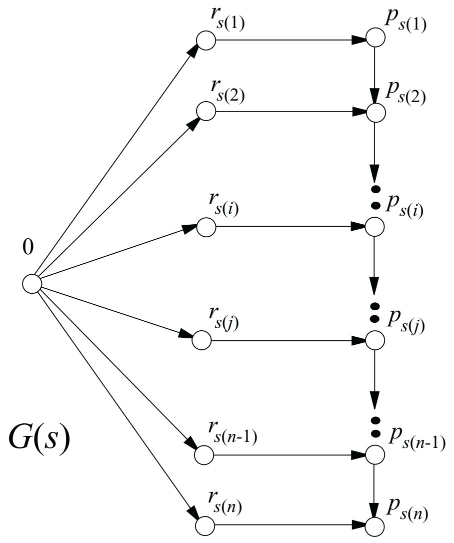


FIG. 3.1. *Directed graph $G(s)$ for sequence $s$.*

DEFINITION 3.1. Adjacent interchange order: $k \lessdot m \Leftrightarrow r_k \le r_m$ *and* $f_k \ge_f f_m$.

Simple arguments show that the $\lessdot$ order defined this way is an adjacent interchange order indeed. Instead of including these in detail here, we just observe that this follows as a special case with $B = \varnothing$ in the proof of Theorem 3.3 below. Note also that this order is not complete in general.

DEFINITION 3.2. *Given $k \lessdot m$, define the following* subsets *of jobs:*

(i) $R^{PI}_{k\lessdot m} = \{j \,|r_j \le r_m, f_k \ge_f f_j\}$,

(ii) $R^{BI}_{k\lessdot m} = \{j \,|r_j \le r_m\}$,

(iii) $R^{FI}_{k\lessdot m} = \{j \,|f_k \ge_f f_j\}$.

THEOREM 3.3. *Partial order $\lessdot$ together with the collection of subsets $R^{PI} = \{R^{PI}_{k\lessdot m} \,|k \lessdot m\}$ satisfies the restricted pairwise interchange condition for $1\,/r_j\,/f_{\max}$.*

*Proof.* Given a sequence $s$, recall that $f_{\max}(s) = \max_{1\le j\le n} f_{s(j)}\left(C_{s(j)}\right)$, where $C_{s(j)}$ is the completion time of job $s(j)$. We construct a directed graph $G(s)$ to evaluate $f_{\max}(s)$ (see Figure 3.1). Each job $s(j)$ is represented by two nodes: the first one has weight $r_{s(j)}$, followed by the second node with weight $p_{s(j)}$. $G(s)$ has the property that the completion time of job $s(j)$ in sequence $s$ is the length of the longest "node-weighted" path from 0 to $s(j)$. Each path goes from node 0 to a node $s(i)$ and ends at a node $s(j)$. Thus we can represent all paths from 0 to $s(j)$ by pairs $(s(i), s(j))$, $1 \le i \le j \le n$. Then by definition

$$\begin{aligned} f_{\max}(s) &= \max_{1\le j\le n}\left[f_{s(j)}\left(\max_{\substack{(s(i),s(j))\\1\le i\le j}}\left(r_{s(i)} + \sum_{l=i}^{j} p_{s(l)}\right)\right)\right] \\ &= \max_{\substack{(s(i),s(j))\\1\le i\le j\le n}}\left[f_{s(j)}\left(r_{s(i)} + \sum_{l=i}^{j} p_{s(l)}\right)\right] \end{aligned}$$

and we can evaluate $f_{\max}(s)$ as the maximum over all such pairs $(s(i), s(j))$ in $G(s)$.

Let $s_1 = (AmBkC)$ be a sequence with the property that $k \lessdot m$ and $B \subseteq R^{PI}_{k\lessdot m}$.

Table 3.1

| $s_2$ | $s_1$ | | | |
|---|---|---|---|---|
| $(AkBmC)$ | $(AmBkC)$ | Proof | | |
| $(a,k)$ | $(a,k)$ | | | $f_k$ *nondecreasing* |
| $(a,b)$ | $(a,k)$ | | $f_k \geq_f f_b$ | $f_k$ *nondecreasing* |
| $(a,m)$ | $(a,k)$ | | $f_k \geq_f f_m$ | $f_k$ *nondecreasing* |
| $(a,c)$ | $(a,c)$ | | | |
| $(k,b)$ | $(m,k)$ | $r_k \leq r_m$ | $f_k \geq_f f_b$ | $f_k$ *nondecreasing* |
| $(k,m)$ | $(m,k)$ | $r_k \leq r_m$ | $f_k \geq_f f_m$ | $f_k$ *nondecreasing* |
| $(k,c)$ | $(m,c)$ | $r_k \leq r_m$ | | $f_c$ *nondecreasing* |
| $(b,m)$ | $(m,k)$ | $r_b \leq r_m$ | $f_k \geq_f f_m$ | $f_k$ *nondecreasing* |
| $(b,c)$ | $(m,c)$ | $r_b \leq r_m$ | | $f_c$ *nondecreasing* |
| $(m,c)$ | $(m,c)$ | | | $f_c$ *nondecreasing* |

Table 3.2

| $s_2$ | $s_1$ | | | |
|---|---|---|---|---|
| $(ABkmC)$ | $(AmBkC)$ | Proof | | |
| $(a,b)$ | $(a,b)$ | | | $f_b$ *nondecreasing* |
| $(a,k)$ | $(a,k)$ | | | $f_k$ *nondecreasing* |
| $(a,m)$ | $(a,k)$ | | $f_k \geq_f f_m$ | $f_k$ *nondecreasing* |
| $(a,c)$ | $(a,c)$ | | | |
| $(b,k)$ | $(b,k)$ | | | |
| $(b,m)$ | $(m,k)$ | $r_b \leq r_m$ | $f_k \geq_f f_m$ | $f_k$ *nondecreasing* |
| $(b,c)$ | $(m,c)$ | $r_b \leq r_m$ | | $f_c$ *nondecreasing* |
| $(k,m)$ | $(m,k)$ | $r_k \leq r_m$ | $f_k \geq_f f_m$ | $f_k$ *nondecreasing* |
| $(k,c)$ | $(m,c)$ | $r_k \leq r_m$ | | $f_c$ *nondecreasing* |
| $(m,c)$ | $(m,c)$ | | | $f_c$ *nondecreasing* |

We apply pairwise interchange to $s_1$ and obtain sequence $s_2 = (AkBmC)$. We demonstrate that $s_2$ is not worse than $s_1$ by showing that for *every* pair of jobs in $s_2$ there exists a *dominating pair* in $s_1$ with a not smaller $f$ value. For example, consider pair $(k,m)$ in $s_2$, then it has the dominating pair $(m,k)$ in $s_1$ (see Figure 3.2), that is,

$$f_m\left(r_k + p_k + \sum_{b\in B} p_b + p_m\right) \leq f_k\left(r_m + p_m + \sum_{b\in B} p_b + p_k\right),$$

which holds since $k \lessdot m$ implies $r_k \leq r_m$ and $f_k(t) \geq f_m(t)$ for all $t$, from which it follows that $f_m\left(r_k + p_k + \sum_{b\in B} p_b + p_m\right) \leq f_k\left(r_k + p_m + \sum_{b\in B} p_b + p_k\right)$, and finally $f_k$ is *nondecreasing.*

Table 3.1 gives a dominating pair in $s_1$ for each pair in $s_2$. (We use lower case letters $a$, $b$, or $c$ to refer to arbitrary generic elements of the subsequences $A$, $B$, or $C$, respectively.) The last three columns contain the arguments that they are dominating pairs. □

Theorem 3.4. *Partial order $\lessdot$ together with the collection of subsets $R^{BI} = \{R^{BI}_{k\lessdot m} \,|k\lessdot m\}$ satisfies the restricted backward insertion condition for* $1/r_j/f_{\max}$.

*Proof.* The proof is totally analogous to that for pairwise interchange. Table 3.2 gives the corresponding dominating pairs. □

Theorem 3.5. *Partial order $\lessdot$ together with the collection of subsets $R^{FI} = \{R^{FI}_{k\lessdot m} \,|k\lessdot m\}$ satisfies the restricted forward insertion condition for* $1/r_j/f_{\max}$.

*Proof.* The proof is totally analogous to that for pairwise interchange. Table 3.3 gives the corresponding dominating pairs. □

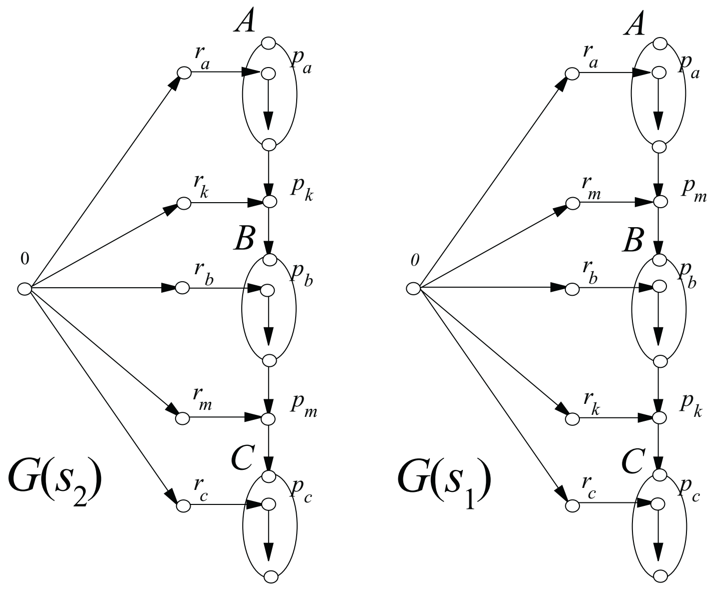


FIG. 3.2. *Directed graphs $G(s_2)$ and $G(s_1)$.*

TABLE 3.3

| $s_2$ | $s_1$ | | | |
|---|---|---|---|---|
| $(AkmBC)$ | $(AmBkC)$ | Proof | | |
| $(a,k)$ | $(a,k)$ | | | $f_k$ *nondecreasing* |
| $(a,m)$ | $(a,k)$ | | $f_k \geq_f f_m$ | $f_k$ *nondecreasing* |
| $(a,b)$ | $(a,k)$ | | $f_k \geq_f f_b$ | $f_k$ *nondecreasing* |
| $(a,c)$ | $(a,c)$ | | | |
| $(k,m)$ | $(m,k)$ | $r_k \leq r_m$ | $f_k \geq_f f_m$ | $f_k$ *nondecreasing* |
| $(k,b)$ | $(m,k)$ | $r_k \leq r_m$ | $f_k \geq_f f_b$ | $f_k$ *nondecreasing* |
| $(k,c)$ | $(m,c)$ | $r_k \leq r_m$ | | $f_c$ *nondecreasing* |
| $(m,b)$ | $(m,b)$ | | | |
| $(m,c)$ | $(m,c)$ | | | $f_c$ *nondecreasing* |
| $(b,c)$ | $(b,c)$ | | | $f_c$ *nondecreasing* |

*Remark* 1. We observe that for any $k \lessdot m$, we have $R^{PI}_{k\lessdot m} = R^{BI}_{k\lessdot m} \cap R^{FI}_{k\lessdot m}$. Thus if $B \subseteq R^{PI}_{k\lessdot m}$, then this implies not only $f(AkBmC) \leq f(AmBkC)$, but also $f(ABkmC) \leq f(AmBkC)$ and $f(AkmBC) \leq f(AmBkC)$.

The preceding theorems could directly be used in branch and bound (B&B) algorithms for restricting the search space on sequences. This, however, would require branching on sequences and storing for all pairs $k \lessdot m$ the subsets of Definition 3.2 and the testing of membership in these, which would be time consuming and inefficient. In the following sections, we show that there is a much more effective way to restrict the search space, by proving that there is a precedence order on the jobs.

We also note that by simply modifying release times and processing times, problems in which the jobs have setup times can be handled as well. Two forms of job setups can be considered: either a setup $\rho_i$ is *attached* to job $i$, i.e., it cannot be performed without the job being present, that is, before $r_i$; or it is *detached,* i.e., it can be performed prior to $r_i$, if the machine is idle and waiting to process job $i$. Attached setups can be dealt with by using modified processing times $p_i^{'} = p_i + \rho_i$, while detached setups can be handled using modified release times $r_i^{'} = \max\{0, r_i - \rho_i\}$ and processing times $p_i^{'} = p_i + \rho_i$. Thus, our theory of precedence constraints applies to the case with setups too.

**4. A general precedence order.** In this section, first we use subset-resticted interchange to derive a precedence order $\prec$ for the general case of $1/r_j/f_{\max}$. We can represent the adjacent interchange order $\lessdot$ using a 3-dimensional structure. The axes of the 3-dimensional space correspond to the $r$, $t$, and $f(t)$ values (see Figure 4.1). Jobs are represented by their $f_j(t)$ curves in this space resting on planes whose height equals their release times. Recall that $k \lessdot m \iff r_k \le r_m$ and $f_k \ge f_m$, that is, if $k$ is on a lower $r$-plane than $m$ and $f_k$ is above $f_m$, i.e., $f_k(t) \ge f_m(t)$ for all $t$. The precedence order $\prec$ is defined in terms of certain "extreme jobs" in $\lessdot$, or, more precisely, certain extreme curves of the 3-dimensional structure. (Some of these extreme curves might not correspond to real jobs.) To identify these jobs, we introduce the *least upper bound* in $\le_f$ for a finite set of jobs $I \subseteq J$, as the nondecreasing function $\text{pmax}_{j\in I} f_j$ defined as the *pointwise maximum* of the functions $f_j$, i.e., with values $(\text{pmax}_{j\in I} f_j(t))(t) = \max_{j\in I} f_j(t)$ for all $t$. A simple greedy procedure to define the set of *boundary jobs* $M = \{M_i \,|i = 1, 2, \ldots, H + 1\}$ and the set of *diagonal jobs* $\Delta = \{D_1, D_2, \ldots, D_H\}$ is presented below. Assume that there are $K$ distinct $r$ values denoted by $r^i$ for $i = 1, 2, \ldots, K$, and $r^1 > r^2 > \cdots > r^K$. Define the function $f^i = \text{pmax}\{f_j \,|r_j = r^i\}$, this is the *least upper bound* in $\le_f$ of the jobs on level $r^i$.

```
ALGORITHM Δ FOR $1/r_j/f_{\max}$.
    Let $(r_{M_1}, f_{M_1}) = (r^1, f^1)$, $l = 1$
    For $i = 2$ to $K$
        If $f_{M_l} \ge_f f^i$ go to Next $i$
        Else
            $(r_{M_{l+1}}, f_{M_{l+1}}) = (r^i, \text{pmax}\{f^i, f_{M_l}\})$ /* This defines $M_{l+1}$ and the
               function $f_{M_{l+1}}$
            $(r_{D_l}, f_{D_l}) = (r^i, f_{M_l})$ /* This defines $D_l$ and the function $f_{D_l}$
            Increase $l$ to $l + 1$
    Next $i$
```

The procedure looks at the $r$-levels $r^i$ $(i = 1, 2, \ldots, K)$ and compares the largest $f$ on this level $(f^i)$ with the largest $f$ obtained so far $(f_{M_l})$. If $f^i$ represents a strict increase compared to $f_{M_l}$, then $(r^i, f^i)$ becomes the new boundary pair $(r_{M_{l+1}}, f_{M_{l+1}})$ and $(r_{D_l}, f_{D_l})$ is the projection of $(r_{M_l}, f_{M_l})$ onto level $r^i$. Note that these extreme jobs are not necessarily real jobs, since they are defined using pointwise maximum. Rather, they are "artificial jobs" used only for ordering purposes. The sets $\Delta$ and $M$ define a "staircase-like" structure that contains all of the curves $f_j$ for $j \in J$ either inside or on its surface (Figure 4.1 shows an example with $|\Delta| = 4$). Later on, we present a numerical example which demonstrates the application of the algorithm $\Delta$ in the case when the $\le_f$ order is linear (see Example 2.)

We augment the partial order $P = (J, \lessdot)$ with the diagonal set $\Delta$ and call it $P_\Delta = (J_\Delta, \lessdot_\Delta)$, where $J_\Delta = J \cup \Delta$ and $\lessdot_\Delta$ is the order $\lessdot$ extended to include the diagonal jobs. Jobs $k \lessdot m$ $(k,m \in J)$ are *separated by* $\Delta$ (are $\Delta$*-separated)* if there exists a $D_i$, $i \in \{1, 2, \ldots, H\}$, such that $k$ is in its principal ideal and $m$ is in its principal filter, i.e., $k \in I(D_i)$ and $m \in F(D_i)$ in $P_\Delta$. $\Delta$ induces a *partition* of $P$ into separable and nonseparable pairs that can be used to define $\prec$.

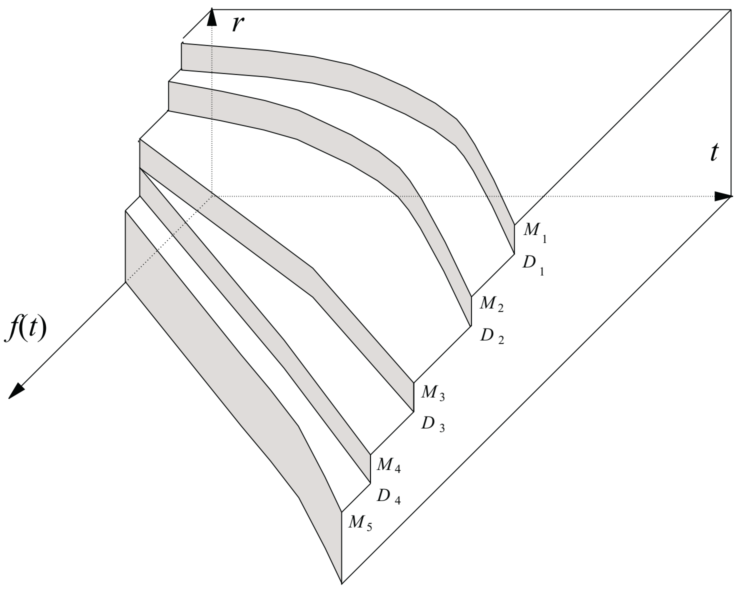


FIG. 4.1. *Staircase structure for the general case.*

DEFINITION 4.1. Precedence order $\prec$: *For $k{<}m$ ($k,m \in J$), we define $k \prec m$ if and only if $k$ and $m$ are separated by $\Delta$.*

It is well known that the main source of difficulty in all $1/r_j/f_{\max}$ problems is the fact that at any time the machine becomes available, it may be better to wait for a yet unreleased job rather than to schedule one of the jobs available. The partition of $P$ by $\Delta$ means that the *only* jobs for which it may be worth waiting are the ones which are *not* separated by $\Delta$ from the currently available jobs, that is, $\prec$ is a precedence order.

THEOREM 4.2. *Partial order $\prec$ is a precedence order for $1/r_j/f_{\max}$.*

*Proof.* We use subset-restricted interchange in the proof. The following observations immediately follow from the construction of $\Delta$:

$$\begin{array}{ccccccc} D_1 & \gtrdot_\Delta & D_2 & \gtrdot_\Delta & \cdots & \gtrdot_\Delta & D_H, \\ F(D_1) & \subset & F(D_2) & \subset & \cdots & \subset & F(D_H), \\ I(D_1) & \supset & I(D_2) & \supset & \cdots & \supset & I(D_H). \end{array}$$

Furthermore, for all $b \in J \setminus F(D_i)$ and $m \in F(D_i)$ we have $r_b \le r_{D_i} \le r_m$ for any $i$. This the *crucial* property used throughout our proof. Let $s$ be any optimal sequence. If every job in $I(D_1)$ is before every job in $F(D_1)$, then all jobs separated by $D_1$ are already in $\prec$ order, and consider $I(D_2)$ and $F(D_2)$. Otherwise, let $k_1 \in I(D_1)$ be the last job in $s$ that is after some job from $F(D_1)$, and let $m_1$ be the last such job from $F(D_1)$ before $k_1$. That is, $s = (A_1 m_1 B_1 k_1 C_1)$, where $C_1 \cap I(D_1) = \varnothing$ and $B_1 \subset J \setminus F(D_1)$. By the above property, we have $r_{b_1} \le r_{D_1} \le r_{m_1}$ for all $b_1 \in B_1$, which implies that $B_1 \subseteq R^{BI}_{k_1 \lessdot m_1}$. Thus, by subset-restricted interchange, we can insert $m_1$ *backward* just after $k_1$ to obtain the alternative optimal sequence $(A_1 B_1 k_1 m_1 C_1)$. Following in this way, inserting the last job in $F(D_1)$ *backward* after $k_1$ until there are no such jobs, we obtain sequence $s_1$ which is an optimal sequence with the property that $I(D_1)$ is before $F(D_1)$. Continuing similarly, if $I(D_2)$ is before $F(D_2)$ in $s_1$, then all jobs separated by $D_2$ are already in $\prec$ order, and consider $I(D_3)$ and $F(D_3)$. Otherwise, let $k_2 \in I(D_2)$ be the last job in $s_1$ after some job from $F(D_2) \setminus F(D_1)$ (since $I(D_2) \subset I(D_1)$ and $I(D_1)$ is before $F(D_1)$ in $s_1$, $k_2$ cannot be after any job from $F(D_1)$), and let $m_2$ be the last such job from $F(D_2) \setminus F(D_1)$. Similarly, we have $s_1 = (A_2 m_2 B_2 k_2 C_2)$, where $C_2 \cap I(D_2) = \varnothing$ and $B_2 \subseteq J \setminus F(D_2)$. As above, we have that $r_{b_2} \leq r_{D_2} \leq r_{m_2}$ for all $b_2 \in B_2$, which implies that $B_2 \subseteq R^{BI}_{k_2 \lessdot m_2}$. Thus we can insert $m_2$ *backward* just after $k_2$ to obtain the sequence $(A_2 B_2 k_2 m_2 C_2)$. When all such jobs in $F(D_2) \setminus F(D_1)$ have been inserted after $k_2$, we obtain sequence $s_2$, an optimal sequence with the property that $I(D_2)$ is before $F(D_2)$ and $I(D_1)$ is before $F(D_1)$. Continuing similarly, we obtain $s_i$ for $i = 3, 4, \ldots, H$. Then $s_H$ is an optimal sequence with the property that $I(D_i)$ is before $F(D_i)$ for $i = 1, 2, \ldots, H$. Thus $s_H$ is a linear extension of $\prec$, and we have that $\prec$ is a precedence order indeed. ☐

Theorem 4.2 also has interesting complexity implications. Let us further augment $P_\Delta$ by adding new least and greatest elements 0 and 1, and call it $P_{0,1} = (J_{0,1}, \lessdot_{0,1})$, where $J_{0,1} = J_\Delta \cup \{0,1\}$ and $\lessdot_{0,1} = \lessdot_\Delta \cup (\{0\} \times J_\Delta) \cup (J_\Delta \times \{1\})$. Then $\prec$ admits an interval representation using intervals $I_j = [x(j), y(j)]$ $(j \in J)$ on the set $S = \Delta \cup \{0,1\}$ linearly ordered by $\lessdot_{0,1}$, where for $j \in J$, $x(j) = \max\{l \in S \,|\, l \lessdot_{0,1} j\}$, and $y(j) = \min\{l \in S \,|\, j \lessdot_{0,1} l\}$. Partial orders $(P, \leq_P)$ that possess such an interval representation on a linearly ordered set (where $k \leq_P m \Leftrightarrow I_k$ lies to the left of $I_m$) are called *interval orders* [5]. This leads to the following corollary for $1/r_j/f_{\max}$.

Corollary 4.3. *Problem $1/r_j, prec/f_{\max}$ remains NP-hard in the strong sense even with interval-order precedence constraints.*

**4.1. Linearly ordered $\geq_{\mathbf{f}}$.** In this section, we consider separately the special case where $\geq_f$ is a linear order. This means that for *each* pair of jobs $i$ and $j$ either $f_i(t) \geq f_j(t)$ for all $t$ or $f_j(t) \geq f_i(t)$ for all $t$. That is, we can *completely* arrange the jobs in nonincreasing $f$ order according to $\geq_f$. We examine an equivalent pyramid representation for $\prec$ (see [3], [4], [14], and [6]) and show how this representation does *not* extend to the nonlinearly ordered general case.

For the linearly ordered case, we are able to represent the adjacent interchange order in the plane using the $r$ and $f$ orders as the $x$ and $y$ axes, respectively. Here jobs are represented by points with preferences toward the origin, i.e., $k \lessdot m$ if $k$ is closer to the origin than $m$ in both the $r$ and $f$ orders. The principal ideals and filters in $\lessdot$ are represented by quadrants through these points. That is, let job $i$ be represented by the point $(r_i, f_i)$. If we divide the plane into quadrants using the lines $r = r_i$ and $f = f_i$, then the $SW$ and $NE$ quadrants correspond to $I(i)$ and $F(i)$, the principal ideal and filter in $\lessdot$ for job $i$ (see Figure 4.2).

This planar representation was used by Merce [14] to derive a precedence order for the problem of minimizing the makespan in the presence of release times and deadlines $(1/r_j, \overline{d_j}/C_{\max})$. Fontan [6] noted that if we consider due dates instead of deadlines, then the same order is a precedence order for the lateness model $1/r_j/L_{\max}$. They did not consider the adjacent interchange order explicitly (see [3] and [4]); rather, they defined their order using certain extreme points in the plane, called summits. These *summits,* represented by $S_i$ ($i = 1$ to $N$), are the jobs that form a staircase boundary in the plane and satisfy the property that their $SE$ quadrant minus the boundary is empty. The summits are completely ordered $S_1 \lessdot S_2 \lessdot \cdots \lessdot S_N$. For each summit $S_i$, Merce and Fontan define a *pyramid* $P(S_i)$, which is its $NW$ quadrant without its boundary lines but including $S_i$. Jobs are classified using pairs that represent their membership in pyramids. For $j \in J$, they define $u(j) = \min\{i \,|\, j \in P(S_i)\}$ and $v(j) = \max\{i \,|\, j \in P(S_i)\}$. With these, they define their partial order $\prec'$ by

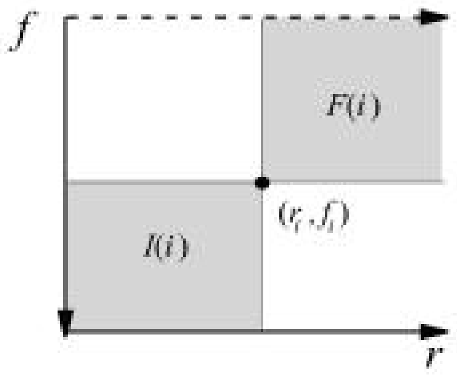


FIG. 4.2. *Ideals and filters for* $\lessdot$.

$k \prec' m \Leftrightarrow v(k) < u(m)$. The planar representation for the 5-job problem of Example 1 is shown in Figure 4.3. Jobs 3 and 5 are the summits $S_1$ and $S_2$, respectively. $P(S_1) = \{1,2,3,4\}$ and $P(S_2) = \{5\}$. We have $v(i) = 1$ for $i = 1,2,3,4$ and $u(5) = 2$, which implies by definition of $\prec'$ that jobs 1, 2, 3, and 4 must precede job 5. Notice that the unique optimal sequence $(1,2,3,4,5)$ is a linear extension of $\prec'$ (as we would expect), but not a linear extension of $\lessdot$ (as we saw earlier). Example 2 is another instance of $1/r_j/L'_{\max}$, this time with $N = 9$ summits. For this instance, the optimal $L'_{\max} = 114$ and the sequence ($1$,$S_1$,$k$,$S_2$,$S_3$,$S_4$,$S_5$,$3$,$S_6$,$S_7$,$m$,$2$,$S_8$,$S_9$,$4$) is an optimal sequence, which is also a linear extension of $\prec'$. To further illustrate how $\prec'$ is defined, consider jobs $k$ and $m$ in Figure 4.4. Here $k \prec' m$, since $v(k) = 5 < 6 = u(m)$.

*Example* 2. A 15-job instance of $1/r_j/L'_{\max}$ with $N = 9$ summits.

| $j$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ | $S_8$ | $S_9$ | $k$ | $m$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $r_j$ | 15 | 26 | 34 | 40 | 48 | 57 | 65 | 65 | 73 | 19 | 51 | 8 | 22 | 30 | 38 |
| $p_j$ | 6 | 8 | 8 | 7 | 6 | 10 | 2 | 5 | 7 | 5 | 3 | 5 | 8 | 8 | 10 |
| $q_j$ | 53 | 46 | 40 | 36 | 36 | 27 | 19 | 13 | 9 | 32 | 17 | 43 | 16 | 22 | 6 |

Let us now apply our general precedence order $\prec$ to this special linearly ordered case: Starting Algorithm $\Delta$, $r^1 = 73$ and $M_1 = S_9$. Note that $f_j(t) = t + q_j$ in this case, meaning that $i \geq_f j \iff q_i \geq q_j$. Thus $f^1(t) = f_{S_9}(t) = t + 9$. Continuing, we have $r^2 = 65$, with both jobs $S_7$ and $S_8$ having this release time. $f^2(t) =$pmax$\{f_{s_7}(t), f_{s_8}(t)\} = t + \max\{q_{s_7}, q_{s_8}\} = t + 19$. This defines $M_2$ by $(r_{M_2}, f_{M_2}) = (65, t+19)$, and $D_1$ by $(r_{D_1}, f_{D_1}) = (r^2, f_{M_1}) = (65, t+9)$. Continuing with the algorithm in this fashion, we obtain the set of corner point boundary jobs $M = \{M_i \,|i = 1,2,\ldots,H+1\}$, and the set of artificial jobs on their inscribed diagonal $\Delta = \{D_1, D_2, \ldots, D_H\}$ (see Figure 4.4). The set $M \subseteq S$ is the subset of boundary jobs with empty $SE$ quadrants, and again we call $\Delta$ the set of diagonal jobs. We augment the partial order $P = (J, \lessdot)$ by these diagonal jobs and call it $P_\Delta = (J_\Delta, \lessdot_\Delta)$, where $J_\Delta = J \cup \Delta$ and $\lessdot_\Delta$ is the planar order with these diagonal jobs included. Jobs $k \lessdot m$ ($k,m \in J$) are separated by $\Delta$ (are $\Delta$-separated) if there exists a $D_i$ ($i = 1,2,\ldots,H$) such that $k \in I(D_i)$ and $m \in F(D_i)$. Notice that $v(k) < u(m)$ when $k$ and $m$ are $\Delta$-separated. It can be easily verified for $k \lessdot m$ ($k,m \in J$) that $k \prec' m$ if and only if $k$ and $m$ are separated by $\Delta$, i.e., $k \prec m$. As an example of this equivalence, consider again Example 1 in Figure 4.3: we see that diagonal job $D_1$ separates job 5 and jobs 1, 2, 3, and 4, and this is the *only* separation present. Thus, by the diagonal representation of $\prec$, jobs 1, 2, 3, and 4 must precede job 5, and $\prec$ and $\prec'$ define the same precedence orders indeed.

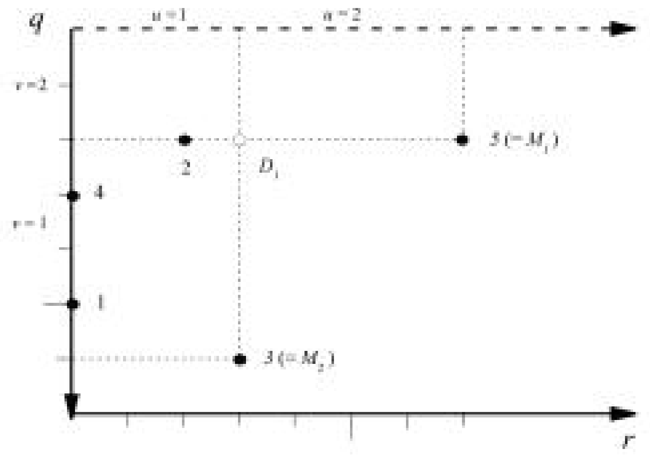


FIG. 4.3. *Planar representation for 5-job problem.*

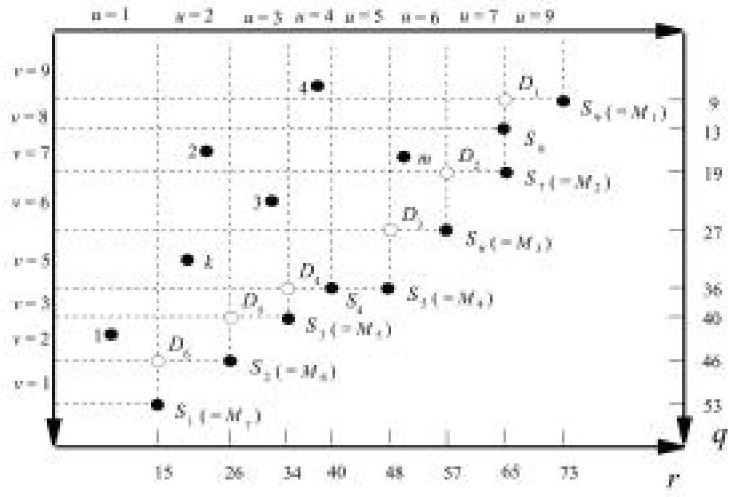


FIG. 4.4. *Planar representation for the 15-job problem.*

The pyramid representation used for $\prec'$ can be reinterpreted in partial order terminology to explain *why it does not* extend to the nonlinearly ordered higher dimensional case. The summits $S_i$ $(i = 1, 2, \ldots, N)$ are *maximal elements* of a related partial order $<^c$, which is defined by $k <^c m \Leftrightarrow r_k < r_m$ and $f_k <_f f_m$, the conjugate of $<$. Two partial orders on the same set are *conjugate* if every pair of distinct elements is comparable in exactly one of these partial orders. This is clearly the case if we compare the principal ideals and filters for $<$ and $<^c$, using the planar representation. For $<$, these are the $SW$ and $NE$ quadrants, respectively, with the boundary lines included. For $\lessdot^c$, these are the $NW$ and $SE$ quadrants minus the boundary lines (compare Figures 4.2 and 4.5). By a well-known theorem of Dushnik and Miller [1] from partial order theory, $\lessdot^c$ exists if and only if $\dim(\lessdot) \leq 2$. In this context, the pyramids are just the *principal ideals* of $\lessdot^c$. The $u(j)$ and $v(j)$ defined in [4] implicitly use the fact that $\lessdot^c$ has an *interval containment order* representation, i.e., there exist intervals $\{I_j | j \in J\}$ such that $i \lessdot^c j$ if and only if $I_i \subset I_j$ for $i, j \in J$. In the pyramid representation of $1/r_j/L_{\max}$, these intervals are just $I_j = [r_j, d_j]$. On the other hand, by the same theorem of Dushnik and Miller [1], a poset has an interval containment representation if and only if its dimension is 2. Thus, the $u(j)$ and $v(j)$ can be defined for *any* problem for which $\dim(\lessdot) \leq 2$, but it can be defined *only* for such problems. Of course, $\dim(\lessdot) \leq 2$ is equivalent to $\geq_f$ being a linear order, so the $u(j)$ and $v(j)$ can be defined *only* in this case. Thus we see that the diagonal representation for $\prec$ has the advantage that it does *not* require that $\lessdot$ possess a conjugate $\lessdot^c$, and thus is *not restricted* by the dimension of $\geq_f$.

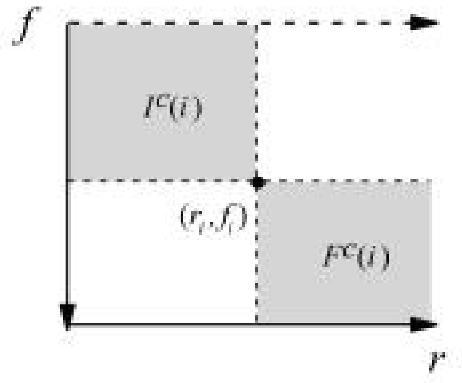


FIG. 4.5. *Ideals and filters for* $\lessdot^c$.

It is interesting to note that the original proof due to Erschler et al. [4], for $1/r_j/L_{\max}$, used *pairwise interchange.* This proof can also be modified to carry over to other cost functions $f$ *when* $\geq_f$ is a linear order. We chose to present a proof using backward insertion, however, because this extends to the (nonlinearly ordered) general case. A proof using forward insertion can also be obtained by proceeding in the opposite direction. This is due to the duality of the operations and regions for the linearly ordered case. However, the proof based on forward insertion is not extendable to the general case either, as the duality of regions no longer holds.

Although this section deals only with linearly ordered $\geq_f$, it covers a number of well-studied scheduling problems. In addition to the ones studied in [3] and [4], we mention one further example.

COROLLARY 4.4. *Partial order $\prec$ is a precedence order for $1/r_j/\max w_j C_j$ and, in this case, $\geq_f$ is the order which orders the jobs in nonincreasing $w_j$ order.*

Theorem 4.2 and Corollary 4.3 also have interesting complexity implications; they yield the following tightening of previously known complexity results [8], [13].

COROLLARY 4.5. *Problems $1/r_j, prec/L_{\max}$ and $1/r_j, prec/\max w_j C_j$ remain NP-hard in the strong sense even with interval-order precedence constraints.*

Corollary 4.5 is interesting, as interval orders have a very special restricted structure [5], which does not seem to help in reducing the complexity of the scheduling problems mentioned. This is in contrast with the result of [16] which shows that $Pm\,/p_j = 1, prec/\,C_{\max}$ is polynomially solvable *for* interval-ordered precedence constraints.

Recall that when the adjacent interchange order $\lessdot$ *itself* is a linear order, it defines an optimal sequence. It can easily be seen that $\prec$ is equivalent to $\lessdot$ in this case, and so it also defines an optimal sequence. This means that $\prec$ also solves some well-known special cases solved by Jackson's rule [9] or Lawler's method [12]: for $1\,/r_j/\,L'_{\max}$, $f_i(t) = t + q_i$ and the jobs are linearly ordered by $f_i \geq_f f_j \Leftrightarrow q_i \geq q_j$. Therefore both $\lessdot$ and $\prec$ are linear orders and define an optimal sequence, for example, *when* the $f$ order is the same as the $r$ order (the agreeably ordered case of $1\,/r_j/\,L'_{\max}$ in which $r_i \leq r_j \Leftrightarrow q_i \geq q_j$), or one of the orders is trivial (i.e., the $f$ order is linear and *all* the jobs have the same release time $1//L_{\max}$), or *all* the jobs have the same cost function $(1\,/r_j, d_j = d/\,L_{\max})$. Similar comments apply to the corresponding special cases of $1\,/r_j/\max w_jC_j$ and $1/r_j/f_{\max}$.

**4.2. Weighted maximum lateness.** For this problem the $\geq_f$ order is *no longer linear*; thus we proceed as in the general case. For the weighted maximum lateness problem, $f_j\,(t) = w_j(t - d_j)$ for all $j \in J$. Let $g_j\,(t) = f_j\,(t) + L$, where the constant $L$ is chosen so that $L \geq \max\{w_jd_j\,|j \in J\}$. That is, $g_j\,(t) = w_jt + q_j$, where $q_j = L - w_jd_j \geq 0$. We see that $f_k \geq_f f_m \iff (w_k \geq w_m)$ and $(q_k \geq q_m)$. Thus, we can represent the adjacent interchange order $\lessdot$ and the interchange regions using this 2-*dimensional* representation of $\geq_f$. This means that our general definitions from before reduce to the following for $1\,/r_j\,/\max w_jL_j$.

$$
\begin{aligned}
k \lessdot m \quad & \Leftrightarrow (r_k \leq r_m) \text{ and } (w_k \geq w_m) \text{ and } (q_k \geq q_m),\\
R^{PI}_{k\lessdot m} \quad & = \{j\,|r_j \leq r_m, w_j \leq w_k, q_j \leq q_k\},\\
R^{FI}_{k\lessdot m} \quad & = \{j\,|w_j \leq w_k, q_j \leq q_k\},\\
R^{BI}_{k\lessdot m} \quad & = \{j\,|r_j \leq r_m\}.
\end{aligned}
$$

We derive the $\Delta$ boundary by modifying the greedy procedure presented earlier. Once again, we assume that there are $K$ distinct $r$ values $r^1 > r^2 > \cdots > r^K$ and let $w^i = \max\{w_j\,|r_j = r^i\}$ and $q^i = \max\{q_j\,|r_j = r^i\}$ for $i = 1, 2, \ldots, K$. We use the same notation as before for the boundary jobs $M_i$ and the diagonal jobs $D_i$.

```
ALGORITHM Δ FOR $1\,/r_j\,/\max w_jL_j$ .
    Let $(r_{M_1}, q_{M_1}, w_{M_1}) = (r^1, q^1, w^1), l = 1$
    For $i = 2$ to $K$
        If $q_{M_l} \geq q^i$ and $w_{M_l} \geq w^i$ then go to Next $i$ /*i.e. $(q_{M_l}, w_{M_l}) \geq_f$
            $(q^i, w^i)$
        Else
            $(r_{M_{l+1}}, q_{M_{l+1}}, w_{M_{l+1}}) = (r^i, \max\{q^i, q_{M_l}\}, \max\{w^i, w_{M_l}\})$ /* This
                defines $M_{l+1}$
            $(r_{D_l}, q_{D_l}, w_{D_l}) = (r^i, q_{M_l}, w_{M_l})$ /* This defines $D_l$
            Increase $l$ to $l + 1$
    Next $i$.
```

If we represent $\lessdot$ in 3 dimensions with $q, w$, and $r$ as the $x, y$, and $z$ axes, respectively, then $(q^i, w^i)$ is the least upper bound according to $\leq_f$ of the jobs on the plane $r = r^i$. (This is well defined by the finiteness of $J$.) Symbolically, $(q^i, w^i) = \max_2\{(q_j, w_j)\,|r_j = r^i\}$, where we define $\max_2\{(q_j, w_j)\,|j \in I\} = (\max_{j\in I} q_j, \max_{j\in I} w_j)$. Taking least upper bounds, we greedily construct the sets of possibly artificial jobs $\Delta$ and $M$. These jobs are on the boundary of a *step pyramid,* which contains all of the original jobs inside or on its surface (see Figure 4.6 for an example with $H = 6$).

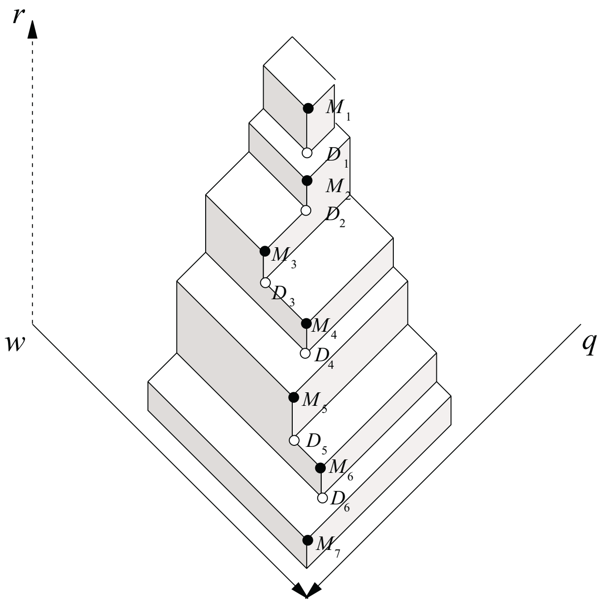


FIG. 4.6. *Staircase structure for weighted maximum lateness.*

The definition of $\prec$ is analogous to the general case: $k \prec m$ if $k$ and $m$ are $\Delta$-separated. From Theorem 4.2 we then have the following corollary for the problem $1\,/r_j\,/\max w_j L_j$.

COROLLARY 4.6. *Partial order $\prec$ is a precedence order for* $1\,/r_j\,/\max w_j L_j$.

## 5. Computational experiment.

We consider a B&B algorithm for $1/r_j/\max w_j C_j$ that follows Potts's implementation [19]. Potts uses an "adaptive" branching technique to *fix* jobs at both ends of the schedule. More precisely, each node of the search tree is represented by a pair $(\sigma_1, \sigma_2)$, where $\sigma_1$ and $\sigma_2$ are the *initial* and *final* partial sequences, respectively. The immediate successor of $(\sigma_1, \sigma_2)$ is either of the form $(\sigma_1 i, \sigma_2)$ for a *type* 1 branching, or $(\sigma_1, i\sigma_2)$ for a *type* 2 branching, where $i$ is an *unfixed* job $i \in J \setminus (\sigma_1 \cup \sigma_2)$. The types of the branchings of the tree are all the same within a level, but in general they will differ between levels. The type for a given level $k$ is fixed on the very first visit to level $k$ according to the following rule: branch in the direction of the fewest number of ties at the minimum lower bound. Let $n_1$ and $n_2$ be the number of ties at the minimum lower bound for type 1 and type 2 branchings, at level $k$. If $n_1 < n_2$, the next branching is of type 1, while if $n_2 < n_1$, then the branching is of type 2. If $n_1 = n_2$, then the branching is the same as on the previous level. Finally, the search strategy is to branch to the newest active node with the smallest lower bound.

The B&B algorithm was coded in Pascal and run on a Sun Sparc5 workstation. The experiment consisted of 24 groups with 100 problems in each group. For each problem, with $n = 100$ or 500 jobs, $3n$ integer data $(r_i, p_i, w_i)$ were generated. Processing times $p_i$ and weights $w_i$ were uniformly distributed between $[1, 100]$ and $[1, W]$ for $W = 10$ and 100, respectively, while release times were uniformly distributed in $[0, n \cdot 50.5 \cdot R]$ for $R \in \{0.2, 0.4, 0.6, 0.8, 1.0, 2.0\}$, following the data generation technique used by Hariri and Potts [7]. Several versions of the algorithm were tested both

Table 5.1
*Results of computational experiment.*

| $n = 100$ | | | | |
|---|---|---|---|---|
| $W = 10$ | CPU time | | Solved | |
| $R$ | with $\prec$ | without $\prec$ | with $\prec$ | without $\prec$ |
| 0.2 | 00:01 | 00:05 | 100 | 100 |
| 0.4 | 12:07 | 43:17 | 100 | 98 |
| 0.6 | 00:18 | 00:34 | 100 | 100 |
| 0.8 | 00:19 | 40:21 | 100 | 99 |
| 1.0 | 00:17 | 00:27 | 100 | 100 |
| 2.0 | 00:09 | 00:14 | 100 | 100 |
| $W = 100$ | CPU time | | Solved | |
| $R$ | with $\prec$ | without $\prec$ | with $\prec$ | without $\prec$ |
| 0.2 | 0:00:42 | 0:01:01 | 100 | 100 |
| 0.4 | 3:16:27 | 4:32:02 | 90 | 89 |
| 0.6 | 0:02:31 | 0:07:16 | 100 | 99 |
| 0.8 | 0:42:20 | 0:56:56 | 92 | 92 |
| 1.0 | 0:00:22 | 0:00:32 | 100 | 100 |
| 2.0 | 0:00:11 | 0:00:14 | 100 | 100 |

| $n = 500$ | | | | |
|---|---|---|---|---|
| $W = 10$ | CPU time | | Solved | |
| $R$ | with $\prec$ | without $\prec$ | with $\prec$ | without $\prec$ |
| 0.2 | 00:44 | 00:35 | 100 | 100 |
| 0.4 | 20:33 | 35:19 | 100 | 100 |
| 0.6 | 12:51 | 26:59 | 100 | 100 |
| 0.8 | 13:08 | 28:09 | 100 | 100 |
| 1.0 | 12:51 | 33:52 | 100 | 100 |
| 2.0 | 09:07 | 27:02 | 100 | 100 |
| $W = 100$ | CPU time | | Solved | |
| $R$ | with $\prec$ | without $\prec$ | with $\prec$ | without $\prec$ |
| 0.2 | 0:00:27 | 0:00:35 | 100 | 100 |
| 0.4 | 0:57:54 | 7:50:25 | 99 | 98 |
| 0.6 | 0:49:48 | 1:26:49 | 100 | 99 |
| 0.8 | 0:39:46 | 0:57:51 | 100 | 100 |
| 1.0 | 0:39:44 | 0:56:32 | 100 | 100 |
| 2.0 | 0:22:08 | 0:31:32 | 100 | 100 |

with and without the precedence order $\prec$; for each group we present only the best results. An upper limit of 1,000,000 was used for the maximum number of nodes in the B&B tree. The data in Table 5.1 shows that this limit was exceeded only for a few problems, i.e., very few problems remained unsolved. Most of these seem to be concentrated in the groups defined by $n = 100$, $W = 100$, and $R = 0.4$ or 0.8. Inspection of the results reveals that the number of problems solved is roughly the same in each case. Nonetheless, the precedence order seems to *significantly improve* the running time of the algorithm. The times shown in Table 5.1 are the *total* times used for solving the 100 problems in the group, using the format (min:sec) or (hrs:min:sec). The average CPU time for the version with the precedence order is 42 percent of that for its counterpart without it, resulting in a savings of 58 percent. This saving from using $\prec$ appears to be due to the reduction in the number of branchings and lower bounds that need to be computed at each node. Note that if the precedence order $\prec$ is used, then lower bounds for type 1 and type 2 branchings need only be calculated for the minimal and maximal jobs in $\prec$, and *not all* unfixed jobs. In summary, the B&B algorithm with the precedence order $\prec$ appears to be a very effective and fast solution method for large problems.

## REFERENCES

[1] B. Dushnik and E.W. Miller, *Partially ordered sets,* Amer. J. Math., 63 (1941), pp. 600–610.
[2] H. Emmons, *One machine sequencing to minimize certain functions of job tardiness,* Oper. Res., 17 (1969), pp. 701–715.
[3] J. Erschler, G. Fontan, C. Merce, and F. Roubellat, *Applying new dominance concepts to job schedule optimization*, European J. Oper. Res., 11 (1982), pp. 60–66.
[4] J. Erschler, G. Fontan, C. Merce, and F. Roubellat, *A new dominance concept in scheduling n jobs on a single machine with ready times and due dates,* Oper. Res., 31 (1983), pp. 114–127.
[5] P.C. Fishburn, *Interval Orders and Interval Graphs*, John Wiley, New York, 1985.
[6] G. Fontan, *Notion de dominance et son application à l'étude de certains problèmes d'ordonnancement*, Thèse de Doctorat ès Sciences, Université Paul Sabatier, Toulouse, France, 1980.
[7] A.M. Hariri and C.N. Potts, *An algorithm for single machine sequencing with release dates to minimize total weighted completion time,* Discrete Appl. Math., 5 (1983), pp. 99–109.
[8] L.A. Hall, *A note on generalizing the maximum lateness criterion for scheduling,* Discrete Appl. Math., 47 (1993), pp. 129–137.
[9] J.R. Jackson, *Scheduling a Production Line to Minimize Maximum Tardiness*, Research Report 43, Management Science Research Project, University of California, Los Angeles, 1955.
[10] S.M. Johnson, *Optimal two- and three-stage production schedules with setup times included,* Naval Res. Logist. Quart., 1 (1954), pp. 61–68.
[11] B.J. Lageweg, J.K. Lenstra, and A.H.G. Rinnooy Kan, *Minimizing maximum lateness on one machine: Computuational experience and some applications,* Statistica Neerlandica, 30 (1976), pp. 25–41.
[12] E.L. Lawler, *Optimal sequencing of a single machine subject to precedence constraints,* Management Science, 19 (1973), pp. 544–546.
[13] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, *Sequencing and Scheduling: Algorithms and Complexity,* in Handbooks in Operations Research and Management Science, Vol. 4, S.C. Graves et al., eds., North-Holland, Amsterdam, 1993.
[14] C. Merce, *Etude de l'existence de solutions pour certains problèmes d'ordonnancement,* Thèse de Doctorat Ingénieur, Université Paul Sabatier, Toulouse, France, 1979.
[15] C.L. Monma, *Properties and Efficient Algorithms for Certain Classes of Sequencing Problems*, Ph.D. thesis, Cornell University, Ithaca, NY, 1978.
[16] C.H. Papadimitriu and M. Yannakakis, *Scheduling interval-ordered tasks,* SIAM J. Comput., 8 (1979), pp. 405–409.
[17] M. Pinedo, *SCHEDULING: Theory, Algorithms and Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
[18] C.N. Potts, *Analysis of a heuristic for one machine sequencing with release dates and delivery times,* Oper. Res., 28 (1980), pp. 1436–1441.
[19] C.N. Potts, *An adaptive branching rule for the permutation flow-shop problem,* European J. Oper. Res., 5 (1980), pp. 19–25.
[20] W.E. Smith, *Various optimizers for single-stage production,* Naval Res. Logist. Quart., 3 (1956), pp. 59–66.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 4, pp. 436–453

# EXACT ANALYSIS OF EXACT CHANGE: THE $k$-PAYMENT PROBLEM[*]

BOAZ PATT-SHAMIR[†], YIANNIS TSIOUNIS[‡], AND YAIR FRANKEL[§]

**Abstract.** We introduce the $k$-payment problem: given a total budget of $N$ units, the problem is to represent this budget as a set of coins, so that any $k$ exact payments of total value at most $N$ can be made using $k$ disjoint subsets of the coins. The goal is to minimize the number of coins for any given $N$ and $k$, while allowing the actual payments to be made on-line, namely without the need to know all payment requests in advance. The problem is motivated by the electronic cash model, where each coin is a long bit sequence, and typical electronic wallets have only limited storage capacity. The $k$-payment problem has additional applications in other resource-sharing scenarios.

Our results include a complete characterization of the $k$-payment problem as follows. First, we prove a necessary and sufficient condition for a given set of coins to solve the problem. Using this characterization, we prove that the number of coins in any solution to the $k$-payment problem is at least $kH_{N/k}$, where $H_n$ denotes the $n$th element in the harmonic series. This condition can also be used to efficiently determine $k$ (the maximal number of exact payments) which a given set of coins allows in the worst case. Secondly, we give an algorithm which produces, for any $N$ and $k$, a solution with a minimal number of coins. In the case that all denominations are available, the algorithm finds a coin allocation with at most $(k+1)H_{N/(k+1)}$ coins. (Both upper and lower bounds are the best possible.) Finally, we show how to generalize the algorithm to the case where some of the denominations are not available.

**Key words.** $k$-payment problem, electronic cash, exact change, coin allocation, change-making

**AMS subject classifications.** 05A17, 05B40, 11P81, 68R05

**PII.** S0895480197318118

## 1. Introduction.

Consider the following everyday scenario. You want to withdraw $N$ units of money from your bank. The teller asks you, "How would you like to have it?" Let us assume that you need to have "exact change," i.e., given any payment request $P \leq N$, you should be able to choose a subset of your "coins" whose sum is precisely $P$. Let us further assume that you would like to withdraw your $N$ units with the least possible number of coins. In this case, your answer depends on your estimate of how many payments you are going to make. In the worst case, you may be making $N$ payments of 1 unit each, forcing you to take $N$ coins of denomination 1. On the other extreme, you may need to make only a single payment $P$. In this case, even if you don't know $P$ in advance, $\log_2(N+1)$ coins are sometimes sufficient (as we explain later). In this article, we provide a complete analysis of the general question, which we call the *$k$-payment problem:* what is the smallest set of coins which enables one to satisfy any $k$ exact payment requests of total value up to $N$?

*Motivation.* In any payment system, be it physical or electronic, some transactions require payments of exact amounts. Forcing shops to provide change to a customer, if she does not possess the exact change, simply shifts the problem from the customers to the shops. The number of coins is particularly important in electronic cash (see, e.g., [2, 3, 12]), because electronic coins are inherently long bit sequences

[*]Received by the editors March 10, 1997; accepted for publication (in revised form) May 19, 2000; published electronically October 6, 2000.
http://www.siam.org/journals/sidma/13-4/31811.html

[†]Department of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel (boaz@eng.tau.ac.il).

[‡]InternetCash Corp., 90 William St., New York, NY 10038 (yiannis@internetcash.com).

[§]ECash.com, 3600 S. Harbor Blvd., Oxnard, CA 93035 (yfrankel@cryptographers.com).

and their handling is computationally intensive, while the typical "smart-card" used to store them has small memory space and computational power [7].

Another interesting application of the problem arises in the context of resource sharing. For concreteness, consider a communication link whose total bandwidth is $N$. When the link is shared by time-multiplexing, there is a fixed schedule which assigns the time-slots (say, cells in ATM lines) to the different connections. Typical schedules have small time-slots, since assigning big slots to small requests entails underutilization. It is important to note, however, that there is an inherent fixed overhead associated with each time-slot (e.g., the header of an ATM cell). Therefore, it would be desirable to have a multiplexing schedule (with time-slots of various sizes) which can accommodate any set of requests with the least number of slots. Similarly to the withdrawal scenario, an improvement upon the trivial $N$ unit-slot solution can be achieved if we know how many connections might be running in parallel. When we know a bound $k$ on this number, and if we restrict ourselves to long-lived connections, the problem of designing a schedule naturally reduces to the $k$-payment problem.

*The $k$-payment problem: Definition.* Formally, the problem is as follows. There are two parameters: the *budget*, denoted $N$, and the *number of payments*, denoted $k$. The problem is to find, for each $i \geq 1$, the *number of $i$-coins* (also called *coins of denomination $i$*), denoted $c_i$, such that the following two requirements are satisfied.

*Budget compliance: $N = \sum_{i=1}^{\infty} ic_i$ and $k$-partition.* For any sequence of $k$ *payment requests*, denoted $P_1, P_2, \ldots, P_k$, with $\sum_{j=1}^{k} P_j \leq N$, there exists a way to exactly satisfy these payments using the coins. That is, there exist nonnegative integers $p_{ij}$ (where $p_{ij}$ represents the number of $i$-coins used in the $j$th payment), such that

- $\sum_i ip_{ij} = P_j$ for each $1 \leq j \leq k$, and
- $\sum_j p_{ij} \leq c_i$ for each $i \geq 1$.

The problem can thus be broken into two parts as follows. The *coin allocation* problem is to partition $N$ into coins given $N$ and $k$, i.e., determining the $c_i$'s. The *coin dispensing* problem is how to actually make a payment, i.e., given the $c_i$'s and a $P_j$ as input, produce the $p_{ij}$ values.

There are a few possible variants of the $k$-payment problem. First, in many systems, not all denominations are available. (For example, we do not know of any system with 3-coins.) Even in the electronic cash realm, denominations may be an expensive resource. (This is because each denomination requires a distinct pair of secret/public keys of the central authority; see, e.g., [1, 8].) Thus an interesting variant of the allocation problem is the *restricted denominations* version, where the set of possible solutions is restricted to only those in which $c_i = 0$ if $i \notin \mathcal{D}$ for a given *allowed denomination set* $\mathcal{D}$.

Also, one may consider the *on-line* coin dispensing problem, where the algorithm is required to dispense coins after each payment request, without knowledge of future requests, or the *off-line* version, where the value of all $k$ payments is assumed to be known before the first coin is dispensed.

*Results.* It turns out that the coin dispensing problem is easy: the greedy strategy works (in the worst-case sense), even in the on-line setting. Henceforth we consider only the on-line problem. Most of our results concern the coin allocation problem, and we sometimes refer to this part of the problem as the $k$-payment problem. Our basic result is a simple necessary and sufficient condition for a sequence $c_1, c_2, \ldots$ of coins to solve the allocation problem (Theorem 3.1). Using this characterization, we prove (in Theorem 3.6) a lower bound of $kH_{N/k} \approx k \ln(N/k)$ on the number of coins in any solution for all $N$ and $k$, where $H_n$ denotes the $n$th element of the harmonic series. The lower bound is the best possible in the sense that it is met with equality for infinitely many $N$'s and $k$'s (Theorem 3.8). The characterization can be used to efficiently determine the maximal number $k$ for which a given collection of coins solves the $k$ payment problem (Corollary 3.9). Our next major result is an efficient algorithm which finds a solution for *any* $N$ and $k$ using the least possible number of coins. We first deal with the case where all denominations are allowed (Theorem 4.1). In this case, the number of coins is never more than $(k+1)H_{N/(k+1)}$ (Theorem 4.7). Similarly to our lower bound, the upper bound is the best possible in general (Theorem 4.8). Finally, using the same ideas in a slightly more refined way, we extend the algorithm to the general case of restricted denomination sets (Theorem 5.1).

*Related work.* To the best of our knowledge, the current work is the first to formulate the general $k$-payment problem, and hence the first to analyze it. One related classic combinatorial problem is *$k$-partition*, where the question is in how many ways a natural number $N$ can be represented as a sum of $k$ positive integers. This problem is less structured than the $k$-payment problem and can be used to derive lower bounds. (However, these bounds are suboptimal; see section 2.) The *postage-stamp problem* is also closely related. Cast in our terms, the postage-stamp problem is to find a set of denominations which will allow one to pay any request of value $1, 2, 3, \ldots, N$ using at most $h$ coins, so that $N$ is maximized. The postage-stamp problem can be viewed as an inverse of the 1-payment problem: there is one payment to make, the number of coins is given, and the goal is to find a denomination set of a given size which will maximize the budget. We remark that the postage-stamp problem is considered a difficult problem even for a very small number of denominations. See, e.g., [10, 11, 4]. Another related question is *change-making* [6], which is the problem of how to represent a given budget with the least number of coins from a given allowed denomination set. General change-making is (weakly) NP-hard. Kozen and Zaks [5], Verma and Xu [14], and Pearson [9] study the question of which denomination sets allow one to use the greedy strategy for optimal change-making.

*Organization.* In section 2 we introduce notation, give some preliminary observations, and briefly discuss a few suboptimal results. In section 3 we prove a characterization of the $k$-payment problem and a lower bound on the number of coins in any solution. In section 4 we present and analyze an optimal algorithm for the unrestricted denomination case. In section 5 we extend the algorithm for the case of restricted denominations. We conclude in section 6 with a few open problems.

## 2. Notation and simple results.

In this section we develop some intuition for the $k$-payment problem by presenting a few simple upper and lower bounds on the number of coins required. The notation we shall use throughout this article is summarized in Figure 2.1. The solution $\mathcal{S}$ to which the $c_i, T_i$, and $m$ symbols refer should be clear from the context.

For the remainder of this article, fix $N$ and $k$ to be arbitrary given positive integers. Note that we may assume without loss of generality that $k \leq N$, since payment requests of value 0 can be ignored.

Let us now do some rough analysis of the $k$-payment problem. As already mentioned above, the case $k = N$ is trivial to solve. Take $c_1 = k$ and $c_i = 0$ for $i \neq 1$. Clearly, no better solution is possible since $c_1 < k$ would not satisfy $k$ payments of value 1 each. The case of $k = 1$ is also quite simple, at least when $N = 2^g - 1$ for an integer $g \geq 1$; we can solve it with $g = \log_2(N+1)$ coins of denominations $1, 2, 4, \ldots, 2^{g-1}$. Given any request $P$, we can satisfy it by using the coins which correspond to the ones in the binary representation of $P$.

**Parameters of problem specification:**

- $N$: the total budget.
- $k$: the number of payments.
- $\mathcal{D}$: the set of allowed denominations. In the unrestricted case, $\mathcal{D} = \mathbf{N}$.

**Quantities related to solution specification $\mathcal{S}$:**

- $c_i$ $(i \geq 0)$: the number of coins of denomination $i$ (a.k.a. $i$-coins) in $\mathcal{S}$. By convention, $c_0 = 0$.
- $m$: the largest denomination of a coin in $\mathcal{S}$. Formally, $m = \max\{i \mid c_i > 0\}$.
- $T_i$ $(i \geq 0)$: the budget allocated in $\mathcal{S}$ using coins of denomination $i$ or less. Formally, $T_i = \sum_{j=1}^{i} jc_j$.
- It is required that $T_m = N$.

**Quantities related to making a payment:**

- $c'_i, T'_i, m'$: refer to the respective quantities after a payment has been made.

**Standard quantities:**

- $H_n = \sum_{i=1}^{n} \frac{1}{i}$. By convention, $H_0 = 0$.

FIG. 2.1. *Glossary of notation.*

However, it is not immediately clear how can one generalize this solution to arbitrary $N$ and $k$. For an arbitrary $N$, the usual technique of "rounding up" to the next power of 2 does not seem appropriate in the $k$-payment problem: can we ask the teller of the bank to round up the amount we withdraw just because it is more convenient for us? But let us ignore this point for the moment, and consider the problem of general $k$. If we were allowed to make the dubious assumption that we may enlarge $N$, then one simple solution would be to duplicate the solution for 1-payment $k$ times, and let each payment use its dedicated set of coins. Specifically, this means that we allocate $k$ 1-coins, $k$ 2-coins, $k$ 4-coins, and so on, up to $k$ $2^{\lceil \log_2(N+1) \rceil - 1}$-coins. The result is approximately $k \log_2 N$ coins, and the guarantee we have based on this simplistic construction is that we can pay $k$ payments, but for all we know *each* of these payments must be of value at most $N$. However, the total budget allocated in this solution is in fact $kN$, and thus it does not seem to solve the $k$-payment problem as stated, where the only limit on the value of payments is placed on their *sum*, rather than on individual values.

As an aside, we remark that one corollary of our work (specifically, Theorem 3.1) is that if coin dispensing is done using the greedy strategy (see below), then the above "binary" construction for coin allocation indeed solves the general $k$-payment problem. More precisely, assume that $(N/k) + 1$ is a power of 2. Then the coin allocation algorithm allocates $k$ coins of each denomination $1, 2, 4, \ldots, 2^{\log_2((N/k)+1)-1}$. Clearly, the number of coins is $k \log_2((N/k)+1)$, and their sum is $N$. Coin dispensing is made *greedily*: at each point, the largest possible coin is used. (A formal description of the greedy dispensing algorithm is presented in Figure 2.2.) However, one should note that, perhaps surprisingly, our results also indicate that the binary algorithm is *not* the right generalization for $k > 1$: the best algorithm (described in section 4) yields a factor of about $(1 - \ln 2)$ improvement, i.e., roughly 30% fewer coins.

Let us now reconsider the question of a general $N$. Once we have an algorithm for infinitely many values of $k$ and $N$, generalizing to arbitrary $N$ and $k$ is easy: find a solution for $N'$ and $k'$, with $N' \geq N$ and $k' \geq k+1$, and then dispense a payment of value $N' - N$. The remaining set of coins is a coin allocation of total budget $N$, and it can be used for $k$ additional payments since the original set solved the $(k+1)$-payment

```
GreedyDispense($P$)
1 while $P > 0$
2     do $i \leftarrow \max\{l \mid l \leq P, c_l > 0\}$        ▷ $i$ is highest possible denomination
3         $j \leftarrow \min(\lfloor P/i \rfloor, c_i)$        ▷ use as many $i$-coins as possible
4         dispense $j$ $i$-coins
5         $c_i \leftarrow c_i - j$
6         $P \leftarrow P - j \cdot i$
```

Fig. 2.2. *The greedy algorithm for coin dispensing. $P$ is the amount to be paid.*

problem.

We close this section with a simple lower bound on the number of coins required in any solution to the $k$-payment problem. The bound is based on a counting argument, and we only sketch it here. For $k = 1$, the number of distinct possible payment requests is $N+1$ (0 is allowed). Observe that the algorithm must dispense a different set of coins in response to each request. It follows that the number of coins in any solution must be at least $\log_2(N+1)$. This argument can be extended to a general $k$, using the observation that the number of distinct responses of the algorithm (disregarding order), is at least $p_k(N)$, the number of ways to represent $N$ as a sum of $k$ positive integers. Using standard bounds for partitions (see, e.g., [13]), and since the number of responses is exponential in the number of coins, one can conclude that the number of coins is $\Omega(k \log(N/k^2))$.

## 3. Problem characterization.

In this section we first prove a simple condition to be necessary and sufficient for a set of coins to correctly solve the $k$-payment problem. Using this result, we obtain a sharp lower bound on the number of coins in any solution to the $k$-payment problem. Finally, we outline an efficient algorithm which, given a set of coins $\mathcal{S}$, determines the maximal $k$ for which $\mathcal{S}$ is a solution of the $k$-payment problem. Please refer to Figure 2.1 for notation.

### 3.1. A necessary and sufficient condition.

Theorem 3.1. *Let $\mathcal{S}$ be a solution as in Figure 2.1. Then $\mathcal{S}$ solves the $k$-payment problem if and only if $T_i \geq ki$ for all $0 \leq i < m$.* The theorem is proven by a series of lemmas below. The necessity proof is not hard; the intuition is that the "hardest" cases are when all payment requests are equal. The more interesting part is the sufficiency proof. We start by proving an upper bound on $m$, the largest denomination in a solution.

Lemma 3.2. *If $\mathcal{S}$ solves the $k$-payment problem, then $m \leq \lceil N/k \rceil$.*

*Proof.* Suppose, for contradiction, that $m > \lceil N/k \rceil$, and consider $k$ payments of values $\lfloor N/k \rfloor$ and $\lceil N/k \rceil$ such that their total sum is $N$. Clearly, none of these payments can use $m$-coins, and since $c_m \geq 1$ by definition, the total budget available for these payments is at most $N - m$, which is a contradiction. ◻

The following lemma is slightly stronger than the condition in Theorem 3.1. We use this version in the proof of Theorem 3.6.

Lemma 3.3. *If $\mathcal{S}$ solves the $k$-payment problem, then $T_i \geq ki$ for all $0 \leq i \leq \lfloor N/k \rfloor$.*

*Proof.* Let $i \leq \lfloor N/k \rfloor$. Then $ki \leq N$. Consider $k$ payments of value $i$ each. Each such payment can be done only with coins of denomination at most $i$, and hence $T_i \geq ki$. ◻

We now turn to sufficiency. We start by showing that if the condition of Theorem 3.1 holds for $k = 1$, then any single payment of value up to $N$ can be satisfied by $\mathcal{S}$ under the greedy algorithm.

LEMMA 3.4. *Let $P$ be a payment request, and suppose that in $\mathcal{S}$ we have that for some $j$,*

*(1) $P \leq T_j$, and*

*(2) $T_i \geq i$ for all $0 \leq i < j$.*

*Then $P$ can be satisfied by the greedy algorithm using only coins of denomination $j$ or less.*

*Proof.* We prove, by induction on $j$, that the claim holds for $j$ and any $P$. The base case is $j = 1$; then, by (1), $T_1 \geq P$ and hence there are at least $P$ 1-coins, which can be used to pay any amount up to their total sum. For the inductive step, assume that the claim holds for $j$ and all $P$, and consider $j + 1$. Let $a$ be the number of $(j + 1)$-coins dispensed by the greedy algorithm, namely,

$$a = \min\left(\left\lfloor \frac{P}{j+1} \right\rfloor, c_{j+1}\right).$$

Let $R$ denote the remainder of the payment after the algorithm dispenses the $(j+1)$-coins, i.e., $R = P - a(j + 1)$. Note that (2) trivially holds after dispensing the $(j + 1)$-coins; we need to show that (1) holds as well. We consider two cases. If $a = c_{j+1}$, then using (1) we get

$$\begin{aligned} T_j &= T_{j+1} - c_{j+1}(j+1) \\ &= T_{j+1} - a(j+1) \\ &\geq P - a(j+1) \\ &= R\ , \end{aligned}$$

and we are done for this case.

If $a < c_{j+1}$, then since the algorithm is greedy, it must be the case that $R < j+1$. On the other hand, by (2) we have that $T_j \geq j$, and hence $T_j \geq R$ and we are done in this case too. □

The following lemma is the key invariant preserved by the greedy algorithm. It is interesting to note that while the algorithm proceeds from larger coins to smaller ones, the inductive proof goes in the opposite direction. Recall that "primed" quantities refer to the value after a payment is done.

LEMMA 3.5. *If $T_i \geq ki$ holds for all $0 \leq i < m$, then, after the greedy algorithm dispenses any amount up to $T_m$, $T'_i \geq (k-1)i$ holds for all $0 \leq i < m'$.*

*Proof.* We use induction on $i$. For $i = 0$ the claim is trivial. Assume that the claim holds for all $l < i$, and consider $i$. Let $S_i = T_i - T'_i$ ($S_i$ is the amount dispensed using coins of denomination at most $i$).

Define $j = \min\left\{j \mid j > i, c'_j > 0\right\}$; namely, $j$ is the smallest remaining denomination which is larger than $i$. Note that $j$ is well defined since $i < m'$; namely, $i$ is not the largest remaining coin. Next, note that since all the coins of denomination $i + 1, i + 2, \ldots, j - 1$ (whose sum is $T_{j-1} - T_i$) were used by the algorithm, we have that

$$S_{j-1} = T_{j-1} - T_i + S_i. \tag{3.1}$$

Now, observe that since the algorithm is greedy, and since at least one $j$-coin was not used by the algorithm, it must be the case that the total amount dispensed using coins of denomination smaller than $j$ is less than $j$, i.e., $S_{j-1} \le j-1$. Using (3.1), we get that $T_i \ge T_{j-1} - j + 1 + S_i$. Finally, using the assumption applied to $j-1$, we obtain

$$\begin{aligned} T_i' &= T_i - S_i \\ &\ge T_{j-1} - j + 1 \\ &\ge (j-1)k - j + 1 \\ &= (j-1)(k-1) \\ &\ge i(k-1)\ . \qquad \square \end{aligned}$$

We now complete the proof of the characterization.

*Proof of Theorem* 3.1. The necessity of the condition follows directly from Lemmas 3.2 and 3.3. For the sufficiency, assume that $T_i \ge ik$ for all $0 \le i < m$, and consider a sequence of up to $k$ requests of total value at most $N$. After the $l$th request is served by the greedy algorithm, we have, by inductive application of Lemma 3.5, that $T_i' \ge (k-l)i$ for all $0 \le i < m$. Moreover, by Lemma 3.4, any amount up to the total remainder can be paid from $\mathcal{S}$, so long as $k - l > 0$, which completes the proof. $\square$

**3.2. A lower bound on the number of coins.** Using Theorem 3.1, we derive a lower bound on the number of coins in any solution to the $k$-payment problem.

THEOREM 3.6. *The number of coins in any solution to the k-payment problem is at least* $kH_{\lfloor N/k \rfloor} \approx k \ln \frac{N}{k}$. We first prove a little lemma we use again in Theorem 4.7.

LEMMA 3.7. *For any number j,* $\sum_{i=1}^{j} c_i = \sum_{i=1}^{j-1} \frac{T_i}{i(i+1)} + \frac{T_j}{j}$ .

*Proof.*

$$\begin{aligned} \sum_{i=1}^{j} c_i &= \sum_{i=1}^{j} \frac{T_i - T_{i-1}}{i} \\ &= \sum_{i=1}^{j-1} \left( \frac{T_i}{i} - \frac{T_i}{i+1} \right) + \frac{T_j}{j} \\ &= \sum_{i=1}^{j-1} \frac{T_i}{i(i+1)} + \frac{T_j}{j}\ . \qquad \square \end{aligned}$$

*Proof of Theorem* 3.6. By Lemmas 3.7 and 3.3,

$$\begin{aligned} \sum_{i=0}^{\lfloor N/k \rfloor} c_i &= \sum_{i=1}^{\lfloor N/k \rfloor - 1} \frac{T_i}{i(i+1)} + \frac{T_{\lfloor N/k \rfloor}}{\lfloor N/k \rfloor} \\ &\ge \sum_{i=1}^{\lfloor N/k \rfloor - 1} \frac{ki}{i(i+1)} + k \\ &= k \sum_{i=1}^{\lfloor N/k \rfloor - 1} \frac{1}{i+1} + k \\ &= k(H_{\lfloor N/k \rfloor} - 1) + k \\ &= kH_{\lfloor N/k \rfloor}\ . \qquad \square \end{aligned}$$

```
ALLOC($N, k$)
1  $c_i \leftarrow 0$ for all $i$                                  ▷ initialize
2  $t \leftarrow 0$
3  $i \leftarrow 0$
4  while $N - t > i$                 ▷ consider denominations in increasing order
5      do $i \leftarrow i + 1$
6          if $t < ki$
7            then $c_i \leftarrow \min\left(\left\lceil \frac{ki-t}{i} \right\rceil, \left\lfloor \frac{N-t}{i} \right\rfloor\right)$      ▷ add $i$-coins but don't overflow
8                 $t \leftarrow t + ic_i$
9  if $N > t$                                                  ▷ add remainder
10   then $c_{N-t} \leftarrow c_{N-t} + 1$
```

FIG. 4.1. *Algorithm for optimal solution of the k-payment problem with unrestricted denominations.*

The lower bound of Theorem 3.6 is the best possible in general, as shown in the following theorem.

THEOREM 3.8. *For any natural numbers $n_1, n_2$, there are infinitely many $N > n_1, k > n_2$, such that there exists a solution for the $k$-payment problem with budget $N$ with exactly $kH_{\lfloor N/k \rfloor}$ coins.*

*Proof.* Choose a natural number $m$ such that $m! > n_2$ and $m \cdot m! > n_1$, and set $k = m!$ and $N = km$. In this case the solution with $c_i = k/i$ for $1 \le i \le N/k$ solves the $k$-payment problem by Theorem 3.1, and its total number of coins is precisely $kH_{\lfloor N/k \rfloor}$. ∎

**3.3. Determining *k* for a given set.** Consider the "inverse problem," in which we are given a set of coins, with $c_i$ coins of denomination $i$ for each $i$, and the question is how many payments we can make using these coins, i.e., find $k$. Note that $k$ is well defined: We say that $k$ is 0 if there is a payment request which cannot be satisfied by the set, and $k$ is never more than the total budget. Theorem 3.1 can be directly applied to answer such a question efficiently, as implied by the following simple corollary.

COROLLARY 3.9. *Let $\mathcal{S}$ be a given coin allocation. Then $\mathcal{S}$ solves the $k$-payment problem if and only if $k \le \min\{\lfloor T_i/i \rfloor \mid 1 \le i < m\}$.*

**4. An optimal algorithm for unrestricted denominations.** We now present a coin allocation algorithm which finds a minimal solution to the $k$-payment for arbitrary $N$ and $k$, assuming that all denominations are available. We first prove the optimality of the algorithm, and then we give an upper bound on the number of coins it allocates. In section 5 we generalize the algorithm to handle a restricted set of denominations.

*The algorithm.* Given arbitrary integers $N \ge k > 0$, the algorithm presented in Figure 4.1 finds an optimal coin allocation. Intuitively, the algorithm works by scanning all possible denominations $i = 1, 2, 3, \ldots$ and adding, for each $i$, the least number of $i$-coins which suffices to make $T_i \ge ki$. The number of coins $c_i$ is thus an approximation of the $i$th harmonic element multiplied by $k$. When the budget is exhausted, the remainder is added simply as a single additional coin.

**4.1. Optimality of** ALLOC.

THEOREM 4.1. *Let $\mathcal{S}$ denote the solution produced by* ALLOC. *Then $\mathcal{S}$ solves the $k$-payment problem using the least possible number of coins.*

*Proof.* The result follows from Lemmas 4.3 and 4.5, below. ∎

The important properties of the algorithm are stated in the following loop invariant.

LEMMA 4.2. *Whenever* ALLOC *executes line* 4, *the following assertions hold.*

(i) $t = \sum_{j=1}^{i} jc_j \leq N$.

(ii) *If* $N - t > i$, *then* $t \geq ik$.

(iii) *If* $i > 0$, *then* $t < i(k+1)$.

*Proof.* Line 4 can be reached after executing lines 1–3 or after executing the loop of lines 5–8. In the former case, we have $t = i = 0$, and the lemma holds trivially.

Suppose now that line 4 is reached after an iteration of the loop. We denote by $t_0$ the value of the variable $t$ *before* the last execution of the loop. If lines 7 and 8 are not executed, then (4.2) holds trivially. If they are executed, then we have that $t = t_0 + ic_i$, and $t \leq \left\lfloor \frac{N-t_0}{i} \right\rfloor \cdot i + t_0 \leq N$, and hence (4.2) holds after the loop is executed. Also note that $t \leq t_0 + \left\lceil \frac{ki-t_0}{i} \right\rceil \cdot i < t_0 + (ki - t_0 + i) = i(k+1)$, and therefore (4.2) holds true after the execution of the loop.

Finally, we prove that (4.2) holds after executing the loop. If lines 7–8 are not executed, (4.2) holds trivially. Suppose that lines 7–8 are executed and that $i < N - t$. In this case, by line 8, $i < N - t_0 - ic_i$, which means that

$$c_i \ < \ \frac{N-t_0}{i} - 1 \ < \ \left\lfloor \frac{N-t_0}{i} \right\rfloor .$$

Therefore, by line 7, $c_i = \left\lceil \frac{ki-t}{i} \right\rceil$, and hence $t = t_0 + \left\lceil \frac{ki-t_0}{i} \right\rceil \cdot i \geq t_0 + ki - t_0 = ki$, and we are done. ∎

Using Lemma 4.2 and Theorem 3.1, correctness is easily proven. We introduce some notation to facilitate separate handling of the remainder.

- $c_i^*$, for all $i$, is the value of the $c_i$ variable when line 9 is executed.
- $T_i^* = \sum_{j=1}^{i} jc_j^*$ .

LEMMA 4.3. $\mathcal{S}$ *solves the* $k$*-payment problem.*

*Proof.* By (4.2) of Lemma 4.2, in conjunction with lines 9–10 of the code, we have that upon completion of the algorithm, $\sum_i ic_i = N$. By (4.2) of Lemma 4.2, and since (by lines 4 and 10) the largest denomination is $m \leq N - T_m^*$, we have that $T_i \geq T_i^* \geq ki$ for all denominations $i < m$, and therefore, by Theorem 3.1, $\mathcal{S}$ solves the $k$-payment problem. ∎

Proving optimality takes more work. We first deal with the the coins allocated before line 9 is reached, and then we show that even if the remainder was nonzero (and an additional coin was allocated in line 10), the solution $\mathcal{S}$ produced by ALLOC is still optimal. To this end, we fix an arbitrary solution $\mathcal{T}$ for the $k$-payment problem and use the following additional notation.

- $d_i$ is the number of $i$-coins in $\mathcal{T}$.
- $U_i$ is the budget allocated in $\mathcal{T}$ using coins of denomination $i$ or less: $U_i = \sum_{j=1}^{i} id_i$.
- $n$ is the largest denomination of a coin in $\mathcal{T}$. Formally, $n = \max\{i \mid d_i > 0\}$.
- $\Delta_0 = 0$ and $\Delta_i = \Delta_{i-1} + d_i - c_i^*$ for $i > 0$.

Note that by the definitions, the following holds for all $i \geq 0$:

$$\Delta_i = \sum_{j=1}^{i} d_j - \sum_{j=1}^{i} c_j^*, \tag{4.1}$$

$$d_i = c_i^* + \Delta_i - \Delta_{i-1}. \tag{4.2}$$

The lemma shows that ALLOC is optimal—ignoring the remainder.

LEMMA 4.4. *$\sum_{i=1}^{j} d_i \geq \sum_{i=1}^{j} c_i^*$ for all $1 \leq j < n$.*

*Proof.* We proceed by contradiction. Suppose $\sum_{i=1}^{l} d_i < \sum_{i=1}^{l} c_i^*$ for $l < n$, and suppose that $l$ is the smallest such index, i.e., using (4.1), $\Delta_l < 0$, and $\Delta_j \geq 0$ for all $0 \leq j < l$. Hence, by (4.2), we have that $d_l < c_l^* - \Delta_{l-1}$, which implies, by integrality, that $d_l \leq c_l^* - \Delta_{l-1} - 1$. Therefore, since $T_l^* < (k+1)l$ by (4.2) of Lemma 4.2, we have

$$\begin{aligned}
U_l &= U_{l-1} + l d_l \\
&\leq \sum_{i=1}^{l-1} i d_i + l c_l^* - l\Delta_{l-1} - l \\
&= \sum_{i=1}^{l-1} i\left(c_i^* + \Delta_i - \Delta_{i-1}\right) + l c_l^* - l\Delta_{l-1} - l \\
&= \sum_{i=1}^{l-1} i c_i^* + \sum_{i=0}^{l-2} (i\Delta_i - (i+1)\Delta_i) \\
&\qquad + (l-1)\Delta_{l-1} + l c_l^* - l\Delta_{l-1} - l \\
&= T_l^* - \sum_{i=1}^{l-1} \Delta_i - l \\
&\leq T_l^* - l \\
&< (k+1)l - l \\
&= kl\ ,
\end{aligned}$$

i.e., $U_l < kl$, which is a contradiction to Theorem 3.1, since $l < n$. □

We now prove that $\mathcal{S}$ is optimal even when considering the remainder.

LEMMA 4.5. $\sum_{i=1}^{m} c_i \leq \sum_{i=1}^{n} d_i$.

*Proof.* We consider two cases. If $n > m$, then, using Lemma 4.4 and the fact that $d_n \geq 1$ by definition of $n$, we get

$$\sum_{i=1}^{m} c_i \ \leq\ \sum_{i=1}^{m} c_i^* + 1 \ \leq\ \sum_{i=1}^{n-1} d_i + 1 \ \leq\ \sum_{i=1}^{n} d_i\ ,$$

and we are done for this case.

So suppose $n \leq m$. Let $b = \sum_{i=n}^{m} c_i - d_n$. We prove the lemma by showing that $\sum_{i=1}^{n-1}(d_i - c_i) \geq b$. First, note that since by Lemma 4.4 we have $\Delta_i \geq 0$ for $i \leq n-1$, and using (4.2) we get

$$\begin{aligned}
U_{n-1} - T_{n-1}^* &= \sum_{i=1}^{n-1} i(d_i - c_i^*) \\
&= \sum_{i=1}^{n-1} i(\Delta_i - \Delta_{i-1}) \\
&= (n-1)\Delta_{n-1} - \sum_{i=0}^{n-2} \Delta_i \\
&\leq (n-1)\Delta_{n-1}\ .
\end{aligned}$$

On the other hand, since $U_n = N = T_m$, we get

$$\begin{aligned}U_{n-1} &= U_n - nd_n \\ &= T_m - n\left(\sum_{i=n}^{m} c_i - b\right) \\ &\geq T_m - \sum_{i=n}^{m} ic_i + bn \\ &= T_{n-1} + bn \ .\end{aligned}$$

Therefore, it follows from (4.1) that

$$\begin{aligned}\sum_{i=1}^{n-1}(d_i - c_i^*) &= \Delta_{n-1} \\ &\geq \left\lceil \frac{U_{n-1} - T^*_{n-1}}{n-1} \right\rceil \\ &\geq \left\lceil \frac{T_{n-1} + bn - T^*_{n-1}}{n-1} \right\rceil . \qquad (4.3)\end{aligned}$$

We now consider two subcases. If $T_{n-1} = T^*_{n-1}$, then $c_i = c_i^*$ for $1 \leq i \leq n-1$, and (4.3) reduces to

$$\sum_{i=1}^{n-1}(d_i - c_i) \geq \left\lceil \frac{nb}{n-1} \right\rceil \geq b \ ,$$

and we are done for this subcase.

Otherwise, $T_{n-1} - T^*_{n-1} \geq 1$, and therefore

$$\begin{aligned}\sum_{i=1}^{n-1}(d_i - c_i) &\geq \sum_{i=1}^{n-1}(d_i - c_i^*) - 1 \\ &\geq \left\lceil \frac{T_{n-1} + bn - T^*_{n-1}}{n-1} \right\rceil - 1 \\ &\geq \left\lceil \frac{bn+1}{n-1} \right\rceil - 1 \\ &\geq b \ ,\end{aligned}$$

and the proof of Lemma 4.5 is complete. ◻

**4.2. The number of coins allocated by** Alloc**.** Theorem 4.1 proved that the number of coins in the solution produced by Alloc is optimal. We now give a tight bound on that number in terms of $N$ and $k$. There is a simple interpretation to the bound: the worst penalty for having $N$ and $k$ which are not "nice" is equivalent to requiring an extra payment (i.e., solving the $(k+1)$-payment problem) for "nice" $N$ and $k$. We remark that we do not know of any direct reduction which proves this result.

First, we prove an upper bound on $m$, which is slightly sharper than the general bound of Lemma 3.2.

Lemma 4.6. *The largest denomination of a coin generated by* Alloc *satisfies* $m \leq \lceil \frac{N}{k+1} \rceil$.

*Proof.* By (4.2) of Lemma 4.2 we have $T^*_i < (k+1)i$, and, in particular,

$$m > \frac{T^*_m}{k+1}. \tag{4.4}$$

By (4.2) of Lemma 4.2 we have that $T^*_{m-1} \geq k(m-1)$, or that $m \leq \frac{T^*_{m-1}}{k} + 1$. Since $c_m \geq 1$, we have that $T^*_{m-1} \leq T^*_m - m$. Also using (4.4), we get that

$$\begin{aligned} m &\leq \frac{T^*_{m-1}}{k} + 1 \\ &\leq \frac{T^*_m - m}{k} + 1 \\ &< \frac{T^*_m - \frac{T^*_m}{k+1}}{k} + 1 \\ &= \frac{T^*_m}{k+1} + 1 \\ &\leq \frac{N}{k+1} + 1\ , \end{aligned}$$

i.e., $m < \frac{N}{k+1} + 1$, and by integrality, $m \leq \lceil \frac{N}{k+1} \rceil$. ☐

THEOREM 4.7. *The number of coins in the solution produced by* ALLOC *is at most* $(k+1)H_{\lceil N/(k+1)\rceil} \approx (k+1)\ln\frac{N}{k+1}$.

*Proof.* By (4.2) of Lemma 4.2, and by integrality, $T^*_i \leq (k+1)i - 1$ for all $1 \leq i \leq m$. This, in conjunction with Lemmas 3.7 and 4.6, implies that

$$\begin{aligned} \sum_{i=1}^{m} c^*_i &= \sum_{i=1}^{m-1} \frac{T^*_i}{i(i+1)} + \frac{T^*_m}{m} \\ &\leq \sum_{i=1}^{m-1} \frac{i(k+1)-1}{i(i+1)} + k + 1 - \frac{1}{m} \\ &= \sum_{i=1}^{m-1} \frac{i(k+1)}{i(i+1)} - \sum_{i=1}^{m-1} \frac{1}{i(i+1)} + k + 1 - \frac{1}{m} \\ &= (k+1)(H_m - 1) + k \\ &\leq (k+1)H_{\lceil N/(k+1)\rceil} - 1\ , \end{aligned}$$

and therefore, $\sum_i c_i \leq \sum_i c^*_i + 1 \leq (k+1)H_{\lceil N/(k+1)\rceil}$, as required. ☐

The upper bound given by Theorem 4.7 is the best possible in general, as proven in the following theorem.

THEOREM 4.8. *For any natural numbers $n_1, n_2$, there are infinitely many $N > n_1, k > n_2$, such that any solution for the k-payment problem for budget N requires at least $(k+1)H_{\lceil N/(k+1)\rceil}$ coins.*

*Proof.* Choose a natural number $m$ such that $m! - 1 > n_2$ and $m \cdot m > n_1$, and set $k = m! - 1$ and $N = 1 + \sum_{i=1}^{m} \lceil k/i \rceil \cdot i = m(k+1)$. For these $N$ and $k$, we have that the algorithm produces the largest denomination $m = \lceil N/(k+1) \rceil$, and $c_i = (k+1)/i$ for $1 \leq i \leq m$, and $c_i = 0$ otherwise. It follows that the number of coins in this case is precisely $(k+1)H_{\lceil N/(k+1)\rceil}$. ☐

**5. Generalization to a restricted set of denominations.** We now turn our attention to the restricted denomination case. In this setting, we are given a set $\mathcal{D}$ of natural numbers, and the requirement is that in any solution, $c_i = 0$ if $i \notin \mathcal{D}$.

```
GAlloc($N, k, \mathcal{D}$)
1  $c_i \leftarrow 0$ for all $i$
2  $t \leftarrow 0$
3  $i \leftarrow 0$
4  while $N - t \geq \bar{i}$
5      do $i \leftarrow \bar{i}$
6         $j \leftarrow \bar{i} - 1$            $\triangleright$ $j$ is the largest den. smaller than the next one in $\mathcal{D}$
7         if $t < kj$                          $\triangleright$ ensure $T_l \geq kl$ for all $l$ up to $j$
8              then if $\lfloor \frac{N-t}{i} \rfloor \geq \lceil \frac{kj-t}{i} \rceil$     $\triangleright$ check if budget is exhausted
9                     then $c_i \leftarrow \lceil \frac{kj-t}{i} \rceil$              $\triangleright$ if not, add coins
10                         $t \leftarrow t + ic_i$
11                    else goto 12
12 if $N > t$
13   then $\mathcal{D}' \leftarrow \{d \mid d \in \mathcal{D}, d \leq i\}$
14        add MakeChange($N - t, \mathcal{D}'$)
```

FIG. 5.1. *Algorithm for optimal solution of the k-payment problem with allowed denominations $\mathcal{D}$.*

To get an optimal algorithm for an arbitrary set of denominations which allows for a solution,[1] we follow the idea of Algorithm ALLOC. Ensure, with the least number of coins, that $T_i \geq ik$ for all $1 \leq i < m$. The treatment of the remainder is more complicated here but has the same motivation: add the remainder with the least possible number of coins. We also have to make sure that the invariant maintained by the main loop is not broken, so we restrict the remainder allocation to use only denominations which were already considered. In fact, remainder allocation is precisely the *change-making* problem, solvable by dynamic programming. (For some allowed denomination sets [5, 14], the greedy strategy works too.) The main algorithm, GALLOC, is given in Figure 5.1. For completeness, we also include, in Figure 5.2, a description of a dynamic programming algorithm for optimal change-making. In GALLOC, and in the analysis, we use the following additional notation.

- $\underline{i} = \max_{j \in \mathcal{D}} \{j \mid j < i\}$, the largest allowed denomination smaller than $i$.
- $\bar{i} = \min_{j \in \mathcal{D}} \{j \mid j > i\}$, the smallest denomination larger than $i$.

We now prove that GALLOC is correct and that the number of coins it allocates is optimal. We remark that the general arguments are similar to those for the unrestricted case but are somewhat more refined here.

THEOREM 5.1. *Let $\mathcal{S}$ denote the solution produced by* GALLOC. *Then $\mathcal{S}$ solves the k-payment problem with allowed denominations $\mathcal{D}$ using the least possible number of coins.*

*Proof.* The result follows from Lemmas 5.3 and 5.8. □

We again use a loop invariant to capture the important properties of the algorithm.

LEMMA 5.2. *Whenever* GALLOC *executes line* 4, *the following assertions hold.*

(i) $t = \sum_{j=1}^{i} jc_j \leq N$.

[1]Observe that the problem is solvable if and only if $1 \in \mathcal{D}$. If there are no 1-coins, then certainly we cannot satisfy any payment request of value 1, and if 1 is allowed, then the trivial solution of $N$ 1-coins works for all $k$.

```
MakeChange($R, \mathcal{D}$)
1  for $i \leftarrow 1$ to $R$
2      do if $i \in \mathcal{D}$
3            then $M[i] \leftarrow \{i\}$
4            else $M[i] \leftarrow \emptyset$
5  while $M[R] = \emptyset$
6      do for $i \leftarrow R$ downto 1
7        do if $M[i] = \emptyset$
8            then for all $j \in \mathcal{D}$ s.t. $j < i$
9                 do if $M[i-j] \neq \emptyset$
10                     then $M[i] \leftarrow M[i-j] \cup \{j\}$
11 return $M[R]$
```

Fig. 5.2. *Dynamic programming algorithm for finding the least number of coins whose sum is $R$ units using denomination set $\mathcal{D}$. $M$ is an array of multisets.*

(ii) *If $N - t \geq \bar{i}$, then $t \geq jk$, where $j = \bar{i} - 1$.*

(iii) *If $i > 0$, then $t < jk + i$.*

*Proof.* Line 4 can be reached after executing lines 1–3 or after executing the loop of lines 5–10. In the former case $t = i = 0$ and the lemma holds trivially.

In the latter case, let $t_0$ denote the value of the variable $t$ before the last execution of the loop. If lines 9 and 10 are not executed, then (5.2) holds trivially. If they are executed, then $t = t_0 + ic_i$ and $t \leq \lfloor \frac{N-t_0}{i} \rfloor \cdot i + t_0 \leq N$; hence (5.2) holds after the loop is executed. Then, whether or not lines 9 and 10 are executed, $t \leq t_0 + \lceil \frac{kj-t_0}{i} \rceil \cdot i < t_0 + (kj - t_0 + i) = jk + i$, and thus (5.2) is true after the execution of the loop.

Finally, we show (5.2) holds after executing the loop. If line 8 is not executed, or if line 11 is executed, then (5.2) holds trivially. Otherwise lines 9–10 are executed, and $c_i = \lceil \frac{kj-t}{i} \rceil$, hence $t = t_0 + \lceil \frac{kj-t_0}{i} \rceil \cdot i \geq t_0 + kj - t_0 = kj$, as required. ∎

The correctness of GAlloc is proven in the next lemma. Following the conventions of section 4, we denote by $\mathcal{S}^*$ the allocation produced by GAlloc before line 12. We denote by $c_i^*$ and $T_i^*$ the values in $\mathcal{S}^*$ corresponding to $c_i$ and $T_i$ in $\mathcal{S}$, respectively.

Lemma 5.3. *$\mathcal{S}$ solves the $k$-payment problem with allowed denominations $\mathcal{D}$.*

*Proof.* By (5.2) of Lemma 5.2, in conjunction with lines 12–14 of the code, we have that upon completion of the algorithm, $\sum_i ic_i = N$. Let $m$ be the highest denomination allocated by the algorithm. Then, by (5.2) of Lemma 5.2, and the fact that MakeChange does not use coins with higher denominations (line 13), we have that for all $1 \leq i < m$, $T_i \geq T_i^* \geq k(\bar{i} - 1) \geq ki$. Therefore, by Theorem 3.1, $\mathcal{S}$ solves the $k$-payment problem. ∎

We now turn to prove optimality of the algorithm. The analysis proceeds similarly to that of section 4: We first show that the allocation of the coins up to the remainder (i.e., just before line 12 is executed) is optimal. We then prove that the handling of the remainder is optimal as well. The proof here is complicated by the fact that in order to ensure correctness, GAlloc allocates the remainder by using only denominations which were already considered in the main loop, and hence it is not immediately clear that the remainder is allocated optimally.

For the remainder of this section, fix an arbitrary "competitor" solution $\mathcal{T}$. We define for $\mathcal{T}$ notions analogous to those defined for the solution $\mathcal{S}$ produced by GAlloc.

- $d_i$ is the number of $i$-coins in $\mathcal{T}$.

- $U_i$ is the budget allocated in $\mathcal{T}$ using coins of denomination $i$ or less: $U_i = \sum_{j=1}^{i} id_i$.
- $n$ is the largest denomination of a coin in $\mathcal{T}$. Formally, $n = \max\{i \mid d_i > 0\}$.

To facilitate treatment of the remainder, we fix an arbitrary subsolution $\mathcal{T}^*$ of $\mathcal{T}$ which solves the $k$-payment problem for budget $T^*_m$. (This is possible since a solution for budget $N$ is also a solution for any budget smaller than $N$.) We use the following additional definitions.

- $U^*_i = \displaystyle\sum_{j=1}^{i} jd^*_j$ . (This is analogous to $T^*_i$ in $\mathcal{S}$).
- $n^*$ is the largest denomination in $\mathcal{T}^*$.
- $m^*$ is the largest denomination allocated up to line 12 by GALLOC. Hence, $T^*_{m^*} = T^*_m$.

Finally, we define $\Delta_0 = 0$ and $\Delta_i = \Delta_{i-1} + d^*_i - c^*_i$ for $i > 0$. As before, by the definitions, for all $i \geq 0$ we have

$$\Delta_i = \sum_{j=1}^{i} d^*_j - \sum_{j=1}^{i} c^*_j, \tag{5.1}$$

$$d^*_i = c^*_i + \Delta_i - \Delta_{i-1} \tag{5.2}$$

We start by proving that GALLOC produces an optimal solution—ignoring the remainder.

LEMMA 5.4. *For all* $1 \leq j < n^*$, $\sum_{i=1}^{j} d^*_i \geq \sum_{i=1}^{j} c^*_i$.

*Proof.* Suppose $\sum_{i=1}^{l} d^*_i < \sum_{i=1}^{l} c^*_i$ for $l < n^*$, and that $l$ is the smallest such index, i.e., using (5.1), $\Delta_l < 0$ and $\Delta_j \geq 0$ for all $0 \leq j < l$. Hence, by (5.2), we have that $d^*_l < c^*_l - \Delta_{l-1}$, which implies, by integrality, that $d^*_l \leq c^*_l - \Delta_{l-1} - 1$. Also, $T^*_l < k(\bar{l} - 1) + l$ by (5.2) of Lemma 5.2. Therefore, since $\bar{l} - 1 < n^*$, we get

$$\begin{aligned}
U^*_{\bar{l}-1} &= U^*_{l-1} + ld^*_l \\
&\leq \sum_{i=1}^{l-1} id^*_i + lc^*_l - l\Delta_{l-1} - l \\
&= \sum_{i=1}^{l-1} i\left(c^*_i + \Delta_i - \Delta_{i-1}\right) + lc^*_l - l\Delta_{l-1} - l \\
&= \sum_{i=1}^{l-1} ic^*_i + \sum_{i=0}^{l-2} (i\Delta_i - (i+1)\Delta_i) \\
&\qquad + (l-1)\Delta_{l-1} + lc^*_l - l\Delta_{l-1} - l \\
&= T^*_l - \sum_{i=1}^{l-1} \Delta_i - l \\
&\leq T^*_l - l \\
&< k(\bar{l} - 1) + l - l \\
&= k(\bar{l} - 1) \ ,
\end{aligned}$$

or $U^*_{\bar{l}-1} < k(\bar{l} - 1)$, a contradiction to Theorem 3.1, since $\bar{l} - 1 < n^*$. ☐

COROLLARY 5.5. $n^* \leq m^*$.

*Proof.* Suppose $n^* > m^*$. Then, by Lemma 5.4, $U^*_{n^*} = \sum_{i=1}^{n^*} id^*_i > \sum_{i=1}^{m^*} id^*_i \geq \sum_{i=1}^{m^*} ic^*_i = T^*_{m^*}$, which is a contradiction to the fact that $U^*_{n^*} = T^*_m = T^*_{m^*}$. ☐

Next, we prove that no solution can use a denomination larger than the largest one used by $\mathcal{S}$.

LEMMA 5.6. *$n \le m$.*

*Proof.* We consider two cases. Suppose first that line 12 is reached after testing the condition on line 4. Then $m^* = m$, and hence the remainder allocated by the dynamic algorithm is $N - T^*_m < \overline{m}$; hence $N - T^*_m \le m$. Note that $n \le \max(n^*, N - T^*_m)$. By Corollary 5.5, $n^* \le m^*$, and therefore, $n \le \max(m^*, m) = m$, and we are done for this case.

Suppose next that line 12 is reached from line 11. In this case we have

$$\left\lfloor \frac{N - T^*_{m^*}}{m} \right\rfloor < \left\lceil \frac{k(\overline{m}-1) - T^*_{m^*}}{m} \right\rceil ,$$

and since $T^*_{m^*} = U^*_{n^*}$, we get

$$\left\lfloor \frac{N - U^*_{n^*}}{m} \right\rfloor < \left\lceil \frac{k(\overline{m}-1) - U^*_{n^*}}{m} \right\rceil ,$$

and hence

$$\begin{aligned} \frac{N - U^*_{n^*} - m}{m} &= \frac{N - U^*_{n^*}}{m} - 1 \\ &< \frac{k(\overline{m}-1) - U^*_{n^*}}{m} , \end{aligned}$$

or

$$N - m \quad < \quad k(\overline{m}-1) \; . \tag{5.3}$$

Now suppose for contradiction that $n > m$. Then at least one coin of denomination $n > m$ is allocated by $\mathcal{T}$; hence $N \ge n + U_m > m + U_m$, or $U_m < N - m$. Since $U_{\overline{m}-1} = U_m$, we get from (5.3) that $U_{\overline{m}-1} < k(\overline{m}-1)$, a contradiction to Theorem 3.1 since $\overline{m} - 1 < \overline{m} \le n$. ☐

It follows that the remainder is allocated optimally (for any choice of a subsolution $\mathcal{T}^*$).

COROLLARY 5.7. *$\sum_{i=1}^n d_i - \sum_{i=1}^n d^*_i \;\; \ge \;\; \sum_{i=1}^m c_i - \sum_{i=1}^m c^*_i$.*

*Proof.* By Lemma 5.6, the amount of $N - T^*_m$ must be allocated in $\mathcal{T}$ using only denominations used in $\mathcal{S}$. The statement therefore follows from the optimality of dynamic programming used in MAKECHANGE. ☐

We can now prove optimality of the full solution $\mathcal{S}$.

LEMMA 5.8. *$\sum_{i=1}^m c_i \le \sum_{i=1}^n d_i$.*

*Proof.* By Corollary 5.7, it is sufficient to prove that the number of coins in $\mathcal{T}^*$ is at least as much as in $\mathcal{S}^*$. If $n^* \ge m^*$, then we are done by Lemma 5.4. It remains to consider the case of $n^* < m^*$. We do it similarly to Lemma 4.5. Define $b = \sum_{i=n^*}^{m^*} c^*_i - d^*_{n^*}$. With this notation, it is sufficient to show that $\sum_{i=1}^{n^*}(d^*_i - c^*_i) \ge b$.

By Lemma 5.4 we have $\Delta_i \ge 0$ for $i \le \underline{n}^*$, and using (5.2) we get

$$\begin{aligned}
U^*_{\underline{n}^*} - T^*_{\underline{n}^*} &= \sum_{i=1}^{\underline{n}^*} i(d^*_i - c^*_i) \\
&= \sum_{i=1}^{\underline{n}^*} i(\Delta_i - \Delta_{i-1}) \\
&= \underline{n}^* \Delta_{\underline{n}^*} - \sum_{i=0}^{\underline{n}^*-1} \Delta_i \\
&\le \underline{n}^* \Delta_{\underline{n}^*} \ .
\end{aligned}$$

Also, since $U^*_{n^*} = T^*_{m^*}$, we have

$$\begin{aligned}
U^*_{\underline{n}^*} &= U^*_{n^*} - n^* d_{n^*} \\
&= T^*_{m^*} - n^* \left( \sum_{i=n^*}^{m^*} c^*_i - b \right) \\
&\ge T^*_{m^*} - \sum_{i=n^*}^{m^*} i c^*_i + bn^* \\
&= T^*_{\underline{n}^*} + bn^* \ .
\end{aligned}$$

Hence, $n^* \Delta_{\underline{n}^*} \ge \underline{n}^* \Delta_{\underline{n}^*} \ge U^*_{\underline{n}^*} - T^*_{\underline{n}^*} \ge bn^*$, i.e., $\Delta_{\underline{n}^*} \ge b$. Thus, from (5.1) we get

$$\sum_{i=1}^{\underline{n}^*} (d^*_i - c^*_i) \quad = \quad \Delta_{\underline{n}^*} \quad \ge \quad b \ ,$$

as required. □

**6. Conclusion.** Motivated by electronic cash, we have specified and analyzed the *k-payment problem* in this paper. It is obvious that the problem is valid in conventional cash models as well. The main requirement is that the payments are made exactly, with no change given back (i.e., no cash given back from the vendor to the customer). We have given a simple "binary" solution, where the denominations used are powers of 2, and an exact optimal solution for the general case, where the set of allowed denominations is arbitrary. If all denominations are allowed, the number of coins is about $k \ln(N/k)$.

We close this paper with a few open problems we find interesting.

- In the course of the analysis we have used a special kind of an approximation of the harmonic series, in which the sum of the first $i$ elements, for each $i$, is the smallest possible value above $k \cdot i$. Are there other applications of the harmonic approximation used in this paper?
- Can the results be extended to the case where denominations may be negative?
- It is easy to see that the running time of Algorithms Alloc and GAlloc—without the invocation of MakeChange—is proportional to the largest denomination they allocate, and hence it is $O(N/k)$ by Lemma 3.2. For Algorithm MakeChange, it is not difficult to see that its running time is bounded by $O(N^2/k)$. Can the computational complexity of the algorithms in this paper be improved?
- Is there a direct reduction proving the results of section 4.2?

**Acknowledgments.** We thank Mike Saks, Richard Stanley, and Yishay Mansour for helpful discussions, Eric Bach for bringing the postage-stamp problem to our attention, Rakesh Verma for providing us with a copy of [14], and Agnes Chan and the anonymous referees for valuable suggestions and comments.

## REFERENCES

[1] S. Brands, *Untraceable off-line cash in wallets with observers*, in Advances in Cryptology: CRYPTO. '93, Lecture Notes in Comput. Sci. 773, Springer-Verlag, Berlin, 1994, pp. 302–318.

[2] D. Chaum, *Blind signatures for untraceable payments*, in Advances in Cryptology: CRYPTO. '82, Plenum Press, New York, 1983, pp. 199–203.

[3] D. Chaum, A. Fiat, and M. Naor, *Untraceable electronic cash*, in Advances in Cryptology: CRYPTO. '88, Springer-Verlag, Berlin, 1990, pp. 319–327.

[4] D. F. Hsu and X.-D. Jia, *External problems in the construction of distributed loop networks*, SIAM J. Disc. Math., 7 (1994), pp. 57–71.

[5] D. Kozen and S. Zaks, *Optimal bounds for the change-making problem*, Theoret. Comput. Sci., 123 (1994), pp. 377–388.

[6] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, Wiley-Intersci. Ser. Discrete Math. Optim., John Wiley and Sons, New York, 1990.

[7] D. Naccache and D. M'Raïhi, *Cryptographic smart cards*, IEEE Micro., 16 (1996), pp. 14–23.

[8] T. Okamoto, *An efficient divisible electronic cash scheme*, in Advances in Cryptology: CRYPTO. '95, Lecture Notes in Comput. Sci. 963, Springer-Verlag, Berlin, 1995, pp. 438–451.

[9] D. Pearson, *A Polynomial-Time Algorithm for the Change-Making Problem*, Technical report TR94-1493, Cornell University, Ithaca, NY, 1994.

[10] E. S. Selmer, *On the postage stamp problem with* 3 *denominations*, Math. Scand., 56 (1985), pp. 105–116.

[11] E. S. Selmer, *Associate bases in the postage stamp problem*, J. Number Theory, 42 (1992), pp. 320–336.

[12] Y. Tsiounis, *Efficient Electronic Cash: New Notions and Techniques*, Ph.D. thesis, College of Computer Science, Northeastern University, Boston, MA, 1997. See http://www.ccs.neu.edu/home/yiannis for information on availability.

[13] J. van Lint and R. Wilson, *A Course in Combinatorics*, Cambridge University Press, Cambridge, UK, 1992.

[14] R. Verma and J. Xu, *On Optimal Greedy Change Making*, Technical report UH-CS-94-03, Department of Computer Science, University of Houston, Houston, TX, 1994.

SIAM J. DISCRETE MATH.
Vol. 13, No. 4, pp. 454–464

# GRAPH SEARCHING AND INTERVAL COMPLETION*

FEDOR V. FOMIN† AND PETR A. GOLOVACH‡

**Abstract.** In the early studies on graph searching a graph was considered as a system of tunnels in which a fast and clever fugitive is hidden. The "classical" search problem is to find a search plan using the minimal number of searchers. In this paper, we consider a new criterion of optimization, namely, the search cost. First, we prove monotone properties of searching with the smallest cost. Then, making use of monotone properties, we prove that for any graph $G$ the search cost of $G$ is equal to the smallest number of edges of all interval supergraphs of $G$. Finally, we show how to compute the search cost of a cograph and the corresponding search strategy in linear time.

**Key words.** graph searching, search cost, vertex separation, interval graph completion, linear layout, profile, cograph

**AMS subject classifications.** 05C78, 05C50, 68R10

**PII.** S0895480199350477

## 1. Introduction.

Search problems on graphs were introduced by Parsons [29] and Petrov [30, 31] (see also [27]). These problems attract the attention of researchers from different fields of mathematics and computer science for a variety of reasons. In the first place, this is the resemblance of graph searching to certain pebble games [21] which model sequential computation. The second motivation of the interest to graph searching arises from VLSI theory. The use of game-theoretic approaches to some important parameters of graph layouts such as cutwidth [26], topological bandwidth [25], and vertex separation number [13] is very useful for constructions of efficient algorithms. Yet another reason is connections between graph searching, pathwidth, and treewidth. These parameters play a very important role in the theory of graph minors developed by Robertson and Seymour (see [1, 12, 33]). Also, graph searching has applications in motion coordinations of multiple robots [34] and in problems of privacy in distributed environments with mobile eavesdroppers ("bugs") [15]. One can find more information on graph searching and related problems in the surveys [1, 4, 14, 28, 32].

In the "classical" node-search version of searching (see, e.g., [21]), at every move of searching a searcher is placed at a vertex or is removed from a vertex. Initially all edges are contaminated (uncleared). A contaminated edge is cleared once both its endpoints are occupied by searchers. A clear edge $e$ is recontaminated if there is a path without searchers leading from $e$ to a contaminated edge. The "classical" search problem is to find the search program such that the maximal number of searchers used at any move is minimized. In this paper we introduce an alternative criterion of optimization. We are looking for node-search programs with the minimal sum of numbers of searchers (the sum is taken over all moves of the search program). We call this criterion the search cost of a graph. Loosely speaking, the cost of a search

*Received by the editors December 30, 1998; accepted for publication (in revised form) July 5, 2000; published electronically October 6, 2000.
http://www.siam.org/journals/sidma/13-4/35047.html

†Faculty of Mathematics and Mechanics, St. Petersburg State University, Bibliotechnaya sq.2, St. Petersburg, 198904, Russia (fomin@gamma.math.spbu.ru). The research of this author was partially supported by the RFBR grant N98-01-00934.

‡Department of Applied Mathematics, Faculty of Mathematics, Syktyvkar State University, Oktyabrsky pr., 55, Syktyvkar, 167001, Russia (golovach@ssu.komi.com). The research of this author was partially supported by the RFBR grant N96-00-00285.

program is the total number of “man-steps” used in this program, and the search cost is the cost of an optimal program. The reader is referred to section 2 for formal definitions of searching and its cost.

One of the most important issues concerning searching is that of recontamination. In some search problems (see [2, 23]) recontamination does not help to search a graph, i.e., if searchers can clear the graph, then they can do it without recontamination of previously cleared edges. We establish the monotonicity of search programs of the smallest cost. To prove the monotonicity we use special constructions called clews. Clews are closely related to crusades used by Bienstock and Seymour in [2] and the notion of measure of clew is related to the notion of linear width introduced by Thomas in [35].

This paper is organized as follows. In section 2 we give necessary definitions. In section 3 we introduce clews and prove the monotonicity of graph searching. In section 4 it is proved that for any graph $G$ the search cost of $G$ is equal to the smallest number of edges of an interval supergraph of $G$. In section 5 it is shown that for any graph $G$ the search cost of $G$ is equal to the vertex separation sum and profile of $G$. In section 6 we show how to compute the search cost of the product of graphs. In section 7 we prove that the search cost of a cograph and the corresponding search program can be obtained in linear time.

## 2. Statement of the problem.

We use the standard graph-theoretic terminology compatible with [7], to which we refer the reader for basic definitions. Unless otherwise specified, $G$ is an undirected, simple (without loops and multiple edges), and finite graph with the vertex set $V(G)$ and the edge set $E(G)$; $n$ denotes the order of $G$, i.e., $|V(G)| = n$.

A *search program* $\Pi$ on a graph $G$ is the sequence of pairs

$$(A_0^1, Z_0^1), (A_0^2, Z_0^2), (A_1^1, Z_1^1), (A_1^2, Z_1^2), \dots, (A_m^1, Z_m^1), (A_m^2, Z_m^2)$$

such that the following hold.

I. For $i \in \{0, \dots, m\}$, $j \in \{1, 2\}$, $A_i^j \subseteq E(G)$, and $Z_i^j \subseteq V(G)$.

II. For $i \in \{0, \dots, m\}$, $j \in \{1, 2\}$, any vertex incident to an edge in $A_i^j$ and to an edge in $E(G) - A_i^j$ is in $Z_i^j$.

III. For $j \in \{1, 2\}$, $A_0^j = \emptyset$, and $A_m^j = E(G)$.

IV (*placing new searchers and clearing edges*). For $i \in \{1, \dots, m\}$, there is $v \in V(G) \setminus Z_{i-1}^2$ such that $Z_i^1 = Z_{i-1}^2 \cup \{v\}$ and $A_i^1 = A_{i-1}^2 \cup E_v$, where $E_v$ is the set of all incident to $v$ edges having one end in $Z_{i-1}^2$.

V (*removing searchers and possible recontamination*). For $i \in \{1, \dots, m\}$, $Z_i^2 \subseteq Z_i^1$ and $A_i^2$ is the set of all edges $e \in A_i^1$ such that every path containing $e$ and an edge of $E(G) - A_i^1$ has an internal vertex in $Z_i^2$.

We call these the *search axioms*. It is useful to treat $Z_i^1$ as the set of vertices occupied by searchers immediately after placing a new searcher at the $i$th step, $Z_i^2$ as the set of vertices occupied by searchers immediately before making the $(i+1)$th step, and $A_i^1$, $A_i^2$ as the sets of cleared edges.

The well-known node search problem [21] is to find $\Pi$ with the smallest $\max_{i \in \{0, \dots, m\}} |Z_i^1|$. (This maximum can be treated as the maximum number of searchers used in one step.) Let us suggest an alternative measure of search. We define the *cost* of $\Pi$ to be $\sum_{i=0}^m |Z_i^2|$. One can interpret the cost of a search program as the total number of “man-steps” used for the search or as the total sum that searchers earn for doing their job. The *search cost* of a graph $G$, denoted by $\gamma(G)$, is the minimum cost of a search program where the minimum is taken over all search programs on $G$.

A search program $(A_0^1, Z_0^1), (A_0^2, Z_0^2), \dots, (A_m^1, Z_m^1), (A_m^2, Z_m^2)$ is *monotone* if, for each $i \in \{0, \dots, m\}$, $A_i^1 = A_i^2$. (Recontamination does not occur when searchers are removed at the $i$th step.) The *monotone search cost* of $G$, $\gamma_m(G)$, is the cost of the minimal (over all monotone search programs) search program on $G$.

Notice that search programs can be defined not only for simple graphs but for graphs with loops and multiple edges as well. Adding loops and multiple edges does not change the search cost. (We suppose that a loop edge is cleared once a searcher is placed on its endpoint.)

**3. Monotone programs and clews.** Let $G$ be a graph. For $X \subseteq E(G)$ we define $V(X)$ to be the set of vertices which are endpoints of $X$, and we let $\delta(X) = V(X) \cap V(E(G) - X)$.

We consider only clews in graphs of a special structure. Let $G^0$ be obtained by adding a loop at each vertex of a graph $G$. A *clew* in $G^0$ is the sequence $(X_0, X_1, \dots, X_m)$ of subsets of $E(G^0)$ such that the following hold.

1. $X_0 = \emptyset$ and $X_m = E(G^0)$.
2. For $i \in \{1, \dots, m\}$, $|V(X_i) - V(X_{i-1})| \leq 1$.
3. For $i \in \{1, \dots, m\}$, if $v \in V(X_i)$, then the loop at $v$ also belongs to $X_i$.

The *measure of the clew* is $\sum_{i=0}^{m} |\delta(X_i)|$. A clew $(X_0, X_1, \dots, X_m)$ is *progressive* if $X_0 \subseteq X_1 \subseteq \cdots \subseteq X_m$ and for any $i \in \{1, \dots, m\}$, $|V(X_i) - V(X_{i-1})| = 1$. Notice that if a clew $(X_0, X_1, \dots, X_m)$ is progressive, then $m = n$.

THEOREM 3.1. *For any graph $G$ and $k \geq 0$ the following assertions are equivalent.*

(i) $\gamma(G) \leq k$.

(ii) *Let $G^0$ be obtained by adding a loop at each vertex of $G$. There is a clew in $G^0$ of measure $\leq k$.*

(iii) *Let $G^0$ be obtained by adding a loop at each vertex of $G$. There is a progressive clew in $G^0$ of measure $\leq k$.*

(iv) $\gamma_m(G) \leq k$.

*Proof.* (i) $\Rightarrow$ (ii). As mentioned above, $\gamma(G) = \gamma(G^0)$. Let

$$(A_0^1, Z_0^1), (A_0^2, Z_0^2), (A_1^1, Z_1^1), (A_1^2, Z_1^2), \dots, (A_m^1, Z_m^1), (A_m^2, Z_m^2)$$

be a search program on $G^0$ with the cost $\leq k$. We prove that $A_0^2, A_1^2, \dots, A_m^2$ is the clew of measure $\leq k$ in $G^0$. The third search axiom implies $A_0^2 = \emptyset$, $A_m^2 = E(G)$. The second search axiom says that for every $i \in \{0, \dots, m\}$, $\delta(A_i^2) \subseteq Z_i^2$, and hence $\sum_{i=1}^{m} |\delta(A_i^2)| \leq k$. Thus $A_0^2, A_1^2, \dots, A_m^2$ is the clew if $|V(A_i^2) - V(A_{i-1}^2)| \leq 1$ for every $i \in \{1, \dots, m\}$. Suppose that for some $i \in \{1, \dots, m\}$ this inequality does not hold. Then there are vertices $u \neq v$ of $V(A_i^2) - V(A_{i-1}^2)$. Notice that the loops $e_u, e_v$ at $u$ and $v$ belong to $A_i^2 - A_{i-1}^2$. From the fifth search axiom $A_i^1 \supseteq A_i^2$ it follows that $e_u, e_v \in A_i^1 - A_{i-1}^2$. The latter is in contradiction with the fourth axiom.

(ii)$\Rightarrow$ (iii). The proof of this implication is closely related to the crusades monotonicity proof by Bienstock and Seymour [2]. Let us choose a clew $(X_0, X_1, \dots, X_m)$ in $G^0$ such that

$$\sum_{i=0}^{m} |\delta(X_i)| \text{ is minimum} \tag{3.1}$$

and, subject to (3.1),

$$\sum_{i=0}^{m} (|X_i| + 1) \text{ is minimum.} \tag{3.2}$$

First we prove that $X_{i-1} \subseteq X_i$ for $i \in \{1, \ldots, m\}$.

Because $V(X_{i-1} \cup X_i) - V(X_{i-1}) = V(X_i) - V(X_{i-1})$ and $V(X_{i+1}) - V(X_{i-1} \cup X_i) \subseteq V(X_{i+1}) - V(X_i)$, it follows that $(X_0, X_1, \ldots, X_{i-1}, X_{i-1} \cup X_i, X_{i+1}, \ldots, X_m)$ is the clew. Using (3.1), we obtain

$$|\delta(X_{i-1} \cup X_i)| \geq |\delta(X_i)|. \tag{3.3}$$

It is easy to check that $|\delta|$ satisfies the submodular inequality

$$|\delta(X_{i-1} \cup X_i)| + |\delta(X_{i-1} \cap X_i)| \leq |\delta(X_{i-1})| + |\delta(X_i)|. \tag{3.4}$$

Combining (3.3) and (3.4), we obtain

$$|\delta(X_{i-1} \cap X_i)| \leq |\delta(X_{i-1})|. \tag{3.5}$$

If $v \in V(X_{i-1}) \cap V(X_i)$, then the loop at $v$ belongs to $X_{i-1} \cap X_i$ by the third search axiom; therefore, $v \in V(X_{i-1} \cap X_i)$ and, consequently, $V(X_i) - V(X_{i-1} \cap X_i) \subseteq V(X_i) - V(X_{i-1})$. Thus, $V(X_{i-1} \cap X_i) - V(X_{i-2}) \subseteq V(X_{i-1}) - V(X_{i-2})$, and $(X_0, X_1, \ldots, X_{i-2}, X_{i-1} \cap X_i, X_i, X_{i+1}, \ldots, X_m)$ is a clew. Taking into account (3.5), (3.1), and (3.2), we get $|X_{i-1} \cap X_i| \geq |X_{i-1}|$. Thus we have $X_{i-1} \subseteq X_i$.

If $|V(X_i) - V(X_{i-1})| = 0$, then $(X_0, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m)$ is the clew contradicting (3.2). Hence, $(X_0, X_1, \ldots, X_m)$ is progressive.

(iii)$\Rightarrow$ (iv). Let $(X_0, X_1, \ldots, X_n)$ be a progressive clew of measure $\leq k$ in $G^0$. We define the search program on $G^0$ setting $Z_0^1 = Z_0^2 = \emptyset$ and $Z_i^1 = \delta(X_i) \cup \{V(X_i) - V(X_{i-1})\}$, $Z_i^2 = \delta(X_i)$ for $i \in \{1, \ldots, n\}$. Suppose that at the $i$th step searchers are placed at vertices of $Z_i^1$ and that all edges of $X_i$ are cleaned. Obviously, no recontamination occurs by removing all searchers from vertices of $Z_i^1 - Z_i^2$.

Define $v = V(X_{i+1}) - V(X_i)$. Every edge of $X_{i+1} - X_i$ either is the loop at $v$, or is incident to $v$ and to a vertex of $\delta(X_i) = Z_i^2$. Then at the $(i+1)$th step the searcher placed at $v$ cleans all edges of $X_{i+1} - X_i$. Finally, $X_0 = \emptyset$, $X_n = E(G)$ with the result that $\gamma_m(G) = \gamma_m(G^0) \leq k$.

(iv)$\Rightarrow$ (i). This part of the proof is obvious. $\square$

## 4. Interval graphs.

**4. Interval graphs.** A graph $G$ is an *interval graph* if and only if one can associate with each vertex $v \in V(G)$ an open interval $I_v = (l_v, r_v)$ on the real line, such that for all $v, w \in V(G)$, $v \neq w$: $(v, w) \in E(G)$ if and only if $I_v \cap I_w \neq \emptyset$. The set of intervals $\mathcal{I} = \{I_v\}_{v \in V}$ is called an (interval) *representation* of $G$.

It is easy to check that every interval graph has an interval representation in which the left endpoints are distinct integers $1, 2, \ldots, n$. Such a representation is said to be *canonical*.

A graph $G$ is a *supergraph* of the graph $G'$ if $V(G') = V(G)$ and $E(G') \subseteq E(G)$. Let $G$ be an interval graph, and let $\mathcal{I} = \{I_v\}_{v \in V(G)}$ be a canonical representation of $G$. The length of $G$ with respect to $\mathcal{I}$, denoted by $l(G, \mathcal{I})$, is

$$\sum_{v \in V} \lfloor r_v - l_v \rfloor.$$

We define the *length* $l(G)$ of an interval graph $G$ as the minimum length over all canonical representations of $G$. For any graph $G$ we define the *interval length* of $G$, denoted by $il(G)$, as the smallest length over all interval supergraphs of $G$.

In the proof of Theorem 4.2, we shall use the following property of canonical representation.

LEMMA 4.1. *Let $I$ be an interval graph of $n$ vertices and $\mathcal{I} = \{I_v = (l_v, r_v)\}_{v \in V(G)}$, $l_v < r_v$, be its canonical representation such that*

$$\sum_{v \in V} \lfloor r_v - l_v \rfloor \text{ is minimum.} \tag{4.1}$$

*For $i \in \{1, \ldots, n\}$ let $P(i)$ be the set of intervals $I_v$, $v \in V(I)$, containing $i$. Then*

$$\sum_{i=1}^{n} |P(i)| = \sum_{v \in V} \lfloor r_v - l_v \rfloor = |E(I)|.$$

*Proof.* (4.1) implies that every interval $I_v = (l_v, r_v)$, $v \in V(I)$, contains $\lfloor r_v - l_v \rfloor$ integers. Every $i \in \{1, \ldots, n\}$ belongs to $|P(i)|$ intervals of $\mathcal{I}$; therefore,

$$\sum_{i=1}^{n} |P(i)| = \sum_{v \in V} \lfloor r_v - l_v \rfloor. \tag{4.2}$$

Let $deg(v)$ be the degree of a vertex $v$ in $I$. Then for every $v \in V(I)$

$$deg(v) = |P(l_v)| + \lfloor r_v - l_v \rfloor \tag{4.3}$$

(the number of intervals $I_u = (l_u, r_u)$ such that $l_u < l_v < r_u$ plus the number of intervals $I_w = (l_w, r_w)$ such that $l_v < l_w < r_v$). By (4.3),

$$2|E(I)| = \sum_{v \in V(I)} deg(v) = \sum_{i=1}^{n} |P(i)| + \sum_{v \in V(I)} \lfloor r_v - l_v \rfloor,$$

which, combined with (4.2), proves Lemma 4.1. □

THEOREM 4.2. *For any graph $G$ and $k > 0$ the following assertions are equivalent.*

(i) *$\gamma(G) \leq k$.*

(ii) *$il(G) \leq k$.*

(iii) *There is an interval supergraph $I$ of $G$ such that $|E(I)| \leq k$.*

*Proof.* (i)⇒ (ii). If $\gamma(G) \leq k$, then by Theorem 3.1 there is a monotone search program

$$(A_0^1, Z_0^1), (A_0^2, Z_0^2), (A_1^1, Z_1^1), (A_1^2, Z_1^2), \ldots, (A_n^1, Z_n^1), (A_n^2, Z_n^2)$$

on $G$ with cost $\leq k$. We choose $\varepsilon < 1$ and assign to each vertex $v$ of $G$ the interval $(l_v, r_v + \varepsilon)$, where a searcher is placed on $v$ at the $l_v$th step and this searcher is removed from $v$ at the $r_v$th step, i.e., $l_v = \min\{i \in \{1, \ldots, n\} | v \in Z_i^1\}$ and $r_v = \max\{i \in \{1, \ldots, n\} | v \in Z_i^1\}$. After the $n$th step of the search program all edges of $G$ are cleared, and hence for every edge $e$ of $G$ there is a step such that both ends of $e$ are occupied by searchers. Therefore, the interval graph $I$ with the canonical representation $\mathcal{I} = \{I_v = (l_v, r_v + \varepsilon)\}_{v \in V(G)}$ is the supergraph of $G$. Because for sufficiently small $\varepsilon$

$$\sum_{v \in V} \lfloor r_v + \varepsilon - l_v \rfloor = \sum_{v \in V} \lfloor r_v - l_v \rfloor = \sum_{i=0}^{n} |Z_i^2|$$

(in the left-hand side equality each vertex $v$ is counted $\lfloor r_v - l_v \rfloor$ times), we see that $il(G) \leq l(I) \leq k$.

(ii)⇒ (iii). This part of the proof follows immediately from Lemma 4.1.

(iii)⇒ (i). Let $\mathcal{I} = \{I_v = (l_v, r_v)\}_{v\in V(G)}$, $l_v < r_v$, be a canonical representation of the minimal length of an interval supergraph $I$ of $G$. It is clear that $r_v < n+1$, $v \in V(G)$.

Let us describe the following search program on $G$.

- $Z_0^1 = \emptyset$.
- For $i \in \{1, \dots, n\}$, we put $Z_i^1 = Z_{i-1}^2 \cup \{v\}$, where $v$ is the vertex assigned to the interval with the left endpoint $i$.
- For $i \in \{0, \dots, n-1\}$, $Z_i^2 = P(i+1)$.
- $Z_n^2 = \emptyset$.

Such actions of searchers do not imply recontamination because each path in $I$ (and hence in $G$) from a vertex $w$, $l_w > i+1$ to a vertex $u$, $l_u < i+1$ contains a vertex of $P(i+1)$. For each edge $e$ of $I$ (and hence for each edge of $G$) there is $i \in \{1, \dots, n\}$ such that both ends of $e$ belong to $Z_i^1$. Then at the $n$th step all edges are cleared.

The cost of the program is

$$\sum_{i=0}^{n} |Z_i^2| = \sum_{i=1}^{n} |P(i)|.$$

By Lemma 4.1 $\gamma(G) \leq |E(I)|$. □

Notice that by Theorem 4.2 for any graph $G$ on $n$ vertices and $e$ edges

$$e \leq \gamma(G) \leq \frac{1}{2}n(n-1).$$

Also, $\gamma(G) = e$ if and only if $G$ is an interval graph, and $\gamma(G) = \frac{1}{2}n(n-1)$ if and only if $G$ is a complete graph.

**5. Linear layouts.** A *linear layout* of a graph $G$ is a one-to-one mapping $f : V(G) \to \{1, \dots, n\}$. There are various interesting parameters associated with linear layouts like Bandwidth, Sum Bandwidth, Cutwidth, etc. (see [10] and [26] for further references).

Define the *profile* [10] of a symmetric $n \times n$ matrix $A = (a_{ij})$ as the minimum value of the sum

$$\sum_{i=1}^{n} (i - \min\{j : a_{ij} \neq 0\})$$

taken over all symmetric permutations of $A$, it being assumed that $a_{ii} = 1$, $i \in \{1, \dots, n\}$. Profile reduction is relevant to the speedup of matrix computations; see [10] for further references on profile. The profile may be redefined as a graph invariant $p(G)$ by finding a linear layout $f$ of $G$ which minimizes the sum

$$\sum_{u\in V(G)} (f(u) - \min\{f(v) : v \in V(G), v \cong u\}),$$

where $\cong$ stands for "is adjacent or equal to."

For $U \subseteq V(G)$ we define

$$\partial U = \{u : u \in U \text{ and there exists } v \in V(G) \setminus U \text{ such that } (u, v) \in E(G)\}.$$

Ellis, Sudborough, and Turner [13] (see also [24]) introduced the following graph parameter. Let $f$ be a linear layout of a graph $G$. Denote by $S_i(G, f)$, $i \in \{1, \dots, n\}$, the set of vertices $\{v : v \in V(G), f(v) \leq i\}$. The *vertex separation with respect to layout $f$* is

$$vs(G, f) = \max_{i \in \{1,\ldots,n-1\}} |\partial S_i(G, f)|,$$

and the *vertex separation $vs(G)$* of a graph $G$ is the minimum vertex separation over all linear layouts of $G$. Let us define an alternative "average norm" ("cost version") of vertex separation. The *vertex separation sum with respect to layout $f$* is

$$vs_{sum}(G, f) = \sum_{i=1}^{n-1} |\partial S_i(G, f)|,$$

and the *vertex separation sum $vs_{sum}(G)$* of a graph $G$ [18] is the minimum vertex separation sum over all linear layouts of $G$.

The following result is due to Billionnet [3].

THEOREM 5.1 (Billionnet). *For any graph $G$ the profile of $G$ is equal to the smallest number of edges, where minimum is taken over all interval supergraphs of $G$.*

The next theorem is related to the theorem of Kirousis and Papadimitriou [20] about node search number and interval width.

THEOREM 5.2. *For any graph $G$ and $k > 0$, the following assertions are equivalent.*

(i) $\gamma(G) \leq k$.
(ii) $vs_{sum}(G) \leq k$.
(iii) $p(G) \leq k$.

*Proof.* Because the application of Theorems 4.2 and 5.1 yields (i)$\Leftrightarrow$ (iii), it remains to prove (i)$\Leftrightarrow$ (ii).

(i)$\Rightarrow$ (ii). Let

$$(A_0^1, Z_0^1), (A_0^2, Z_0^2), (A_1^1, Z_1^1), (A_1^2, Z_1^2), \ldots, (A_n^1, Z_n^1), (A_n^2, Z_n^2)$$

be a monotone search program on $G$ with cost $\leq k$. Define a layout $f : V(G) \to \{1, \ldots, n\}$ so that $f(u) < f(v)$ if and only if $u$ accepts a searcher before $v$ does. By the second search axiom $|Z_i^2| \leq |\partial S_i(G, f)|$ for each $i \in \{1, \ldots, n\}$, and we conclude that $vs_{sum}(G, f) \leq k$.

(ii)$\Rightarrow$ (i). Let $f : V(G) \to \{1, \ldots, n\}$ be a linear layout such that $vs_{sum}(G, f) \leq k$. We define the following subsets of $V(G)$.

- $Z_0^1 = Z_0^2 = \emptyset$.
- For $i \in \{1, \ldots, n\}$, we put $Z_i^1 = Z_{i-1}^2 \cup \{v\}$, where $v = f^{-1}(i)$.
- For $i \in \{1, \ldots, n\}$, $Z_i^2 = \partial S_i(G, f)$.

For $i \in \{0, \ldots, n\}$ and $j = 1, 2$, let $A_i^j$ be the set of edges induced by $\cup_{k=0}^{i} Z_k^1$. We observe that for each edge $(u, v) \in E(G)$ there is $i \in \{1, \ldots, n-1\}$ such that $u, v \in Z_i^1$. (If $f(u) < f(v)$, then $i = f^{-1}(v)$.) On this basis it is straightforward to prove (as in Theorem 4.2) that the sequence

$$(A_0^1, Z_0^1), (A_0^2, Z_0^2), (A_1^1, Z_1^1), (A_1^2, Z_1^2), \ldots, (A_n^1, Z_n^1), (A_n^2, Z_n^2)$$

is the (monotone) search program of cost $\leq k$ on $G$. ∎

**5.1. Complexity remark.** The problem of INTERVAL GRAPH COMPLETION is as follows.

Instance: A graph $G$ and an integer $k$.

Question: Is there an interval supergraph $I$ of $G$ such that $|E(I) - E(G)| \leq k$?

This problem is NP-complete even when $G$ is stipulated to be an edge graph (see [16, Problem GT35]). INTERVAL GRAPH COMPLETION arises in computational biology (see, e.g., [5]) and is known to be *fixed parameter tractable (FPT)* [9, 19].

From Theorem 4.2 it follows immediately that the problem of SEARCH COST, deciding given a graph $G$ and an integer $k$ whether $\gamma(G) \leq k$ or not, is NP-complete even for edge graphs and that finding the search cost is $FPT$ for a fixed $k$.

An $O(n^{1.722})$ time algorithm was given in [22] for the profile problem for the case that $G$ is a tree with $n$ vertices.

**6. Product of graphs.** Let $G$ and $H$ be disjoint graphs, i.e., $V(G) \cap V(H) = \emptyset$. The disjoint union of disjoint graphs $G$ and $H$ is the graph $G \dot{\cup} H$ with the vertex set $V(G) \cup V(H)$ and the edge set $E(G) \cup E(H)$.

We use $G \times H$ to denote the following type of "product" of disjoint graphs $G$ and $H$: $G \times H$ is the graph with the vertex set $V(G) \cup V(G)$ and the edge set $E(G) \cup E(H) \cup \{(u, w) : v \in V(G), w \in V(H)\})$.

The following theorem is similar to results on edge-search and node-search numbers [17, 28] and on the pathwidth and the treewidth of a graph [6].

THEOREM 6.1. *Let $G_1, G_2$ be disjoint graphs, $|V(G_1)| = n_1$, $|V(G_2)| = n_2$. Then*

(i) $\gamma(G_1 \dot{\cup} G_2) = \gamma(G_1) + \gamma(G_2)$.

(ii) $\gamma(G_1 \times G_2) = \min\{\frac{1}{2}n_1(n_1-1) + \gamma(G_2), \frac{1}{2}n_2(n_2-1) + \gamma(G_1)\} + n_1 n_2$.

*Proof.*

(i) This part of the proof is trivial.

(ii) Let $f$ be an optimal layout of $G_1 \times G_2$, i.e.,

$$vs_{sum}(G_1 \times G_2) = \sum_{i=1}^{n_1+n_2-1} |\partial S_i(G_1 \times G_2, f)|.$$

Let $k$ be the smallest number ensuring that $V(G_1) \subseteq S_k(G_1 \times G_2, f)$ or $V(G_2) \subseteq S_k(G_1 \times G_2, f)$. For clarity's sake, without loss of generality we suppose that $V(G_1) \subseteq S_k(G_1 \times G_2, f)$. Obviously, $k \geq n_1$. Let $g : V(G_2) \to n_2$ be the "restriction" of $f$ to $V(G_2)$, i.e., for any $u, v \in V(G_2)$, $g(u) < g(v)$ if and only if $f(u) < f(v)$.

Putting $S_0(G_2, g) = \emptyset$, we see that $|\partial S_i(G_1 \times G_2, f)| = i$ for $i \in \{1, \ldots, k-1\}$ and $|\partial S_i(G_1 \times G_2, f)| = n_1 + |\partial S_{i-n_1}(G_2, g)|$ for $i \in \{k, \ldots, n_1 + n_2 - 1\}$. In addition, for any $i \in \{1, \ldots, n_2\}$, $|\partial S_i(G_2, g)| \leq i$. Consequently,

$$\begin{aligned} vs_{sum}(G_1 \times G_2) &= \sum_{i=1}^{n_1+n_2-1} |\partial S_i(G_1 \times G_2, f)| \\ &= \sum_{i=1}^{k-1} i + \sum_{i=k}^{n_1+n_2-1} (n_1 + |\partial S_{i-n_1}(G_2, g)|) \\ &= \sum_{i=1}^{n_1} i + \sum_{i=1}^{k-n_1-1} i + \sum_{i=k}^{n_1+n_2-1} |\partial S_{i-n_1}(G_2, g)| + n_1(n_2 - 1) \\ &= \frac{1}{2} n_1(n_1 - 1) + \sum_{i=1}^{k-n_1-1} i + \sum_{i=k-n_1}^{n_2-1} |\partial S_i(G_2, g)| + n_1 n_2 \end{aligned}$$

$$\geq \frac{1}{2}n_1(n_1-1) + \sum_{i=1}^{n_2-1} |\partial S_i(G_2,g)| + n_1n_2$$
$$\geq \frac{1}{2}n_1(n_1-1) + vs_{sum}(G_2) + n_1n_2.$$

Finally,

$$vs_{sum}(G_1 \times G_2) \geq \min\left\{\frac{1}{2}n_1(n_1-1) + vs_{sum}(G_2), \frac{1}{2}n_2(n_2-1) + vs_{sum}(G_1)\right\} + n_1n_2.$$

For the other direction, let $f$ be an arbitrary layout of $G_1$ and let $g$ be a layout of $G_2$ such that $vs_{sum}(G_2) = \sum_{i=1}^{n_2-1} |\partial S_i(G_2,g)|$. Define the layout $h$ of $G_1 \times G_2$ by the rule

$$h(v) = \begin{cases} f(v), & \text{if } v \in V(G_1), \\ n_1 + g(v), & \text{if } v \in V(G_2). \end{cases}$$

It follows easily that $|\partial S_i(G_1 \times G_2, h)| = i$ whenever $i \in \{1,\ldots,n_1\}$ and that $|\partial S_i(G_1 \times G_2, h)| = n_1 + |\partial S_{i-n_1}(G_2,g)|$ if $i \in \{n_1+1,\ldots,n_1+n_2-1\}$. We conclude that

$$vs_{sum}(G_1 \times G_2) \leq \sum_{i=1}^{n_1+n_2-1} |\partial S_i(G_1 \times G_2, h)|$$
$$= \sum_{i=1}^{n_1} i + \sum_{i=n_1+1}^{n_1+n_2-1} (n_1 + |\partial S_{i-n_1}(G_2,g)|)$$
$$= \frac{1}{2}n_1(n_1+1) + n_1(n_2-1) + \sum_{i=1}^{n_2-1} |\partial S_i(G_2,g)|$$
$$= \frac{1}{2}n_1(n_1-1) + vs_{sum}(G_2) + n_1n_2.$$

Similarly,

$$vs_{sum}(G_1 \times G_2) \leq \frac{1}{2}n_2(n_2-1) + vs_{sum}(G_1) + n_1n_2. \qquad \square$$

**7. Cographs.** Theorem 6.1 can be used for obtaining linear time algorithms for the search cost and the corresponding search strategy on cographs. Recall that a graph $G$ is a cograph if and only if one of the following conditions is fulfilled:

1. $|V(G)| = 1$.
2. There are cographs $G_1,\ldots,G_k$ and $G = G_1 \dot\cup G_2 \dot\cup \cdots \dot\cup G_k$.
3. There are cographs $G_1,\ldots,G_k$ and $G = G_1 \times G_2 \times \cdots \times G_k$.

(See [8] for references on cographs and different graph classes.)

Linear time algorithms computing the search cost and optimal search program on cographs follow rather straightforwardly from the theory developed in the preceding sections. A similar algorithm for the treewidth and the pathwidth of cographs was described in [6]. The main idea of the algorithms is in constructing a sequence of operations $\dot\cup$ and $\times$ producing the cograph $G$. With each cograph $G$ one can associate a binary labeled tree which is called the *cotree* $T_G$. $T_G$ has the following properties.

1. Each internal vertex $v$ of $T_G$ has label$(v) \in \{0,1\}$.
2. There is a bijection $\tau$ between the set of leaves of $T_G$ and $V(G)$.

3. To each vertex $v \in (T_G)$ we assign the subgraph $G_v$ of $G$ as follows.
   (a) If $v$ is a leaf, then $G_v = \tau(v)$.
   (b) If $v$ is an internal vertex and label($v$)= 0, then $G_v = G_u \dot{\cup} G_w$, where $u, w$ are the sons of $v$.
   (c) If $v$ is an internal vertex and label($v$)= 1, then $G_v = G_u \times G_w$, where $u, w$ are the sons of $v$.

Notice that if $r$ is the root of $T_G$, then $G_r = G$. Corneil, Perl, and Stewart [11] gave an $O(|V(G)| + |E(G)|)$ algorithm for determining whether a given graph $G$ is a cograph and, if so, for constructing the corresponding cotree.

We omit the detailed proofs of the following theorems.

THEOREM 7.1. *The search cost of a cograph given with a corresponding cotree can be computed in $O(n)$ time.*

THEOREM 7.2. *Let $G$ be a cograph of $n$ vertices and $e$ edges. The optimal search program on $G$ can be constructed in $O(n + e)$ time.*

## 8. Concluding remarks.

In this paper, we have introduced a game-theoretic approach to the problem of interval completion with the smallest number of edges. There are similar approaches to the pathwidth and treewidth parameters. The interesting problem is whether there is a graph-searching "interpretation" of the fill-in problem (see also [36] for monotonicity proofs of another variant of graph searching).

## REFERENCES

[1] D. BIENSTOCK, *Graph searching, path-width, tree-width and related problems (a survey)*, DIMACS Ser. Discrete Mathematics and Theoretical Computer Science 5, Amer. Math. Soc., Providence, RI, 1991, pp. 33–49.
[2] D. BIENSTOCK AND P. SEYMOUR, *Monotonicity in graph searching*, J. Algorithms, 12 (1991), pp. 239–245.
[3] A. BILLIONNET, *On interval graphs and matrice profiles*, RAIRO Rech. Opér., 20 (1986), pp. 245–256.
[4] H. L. BODLAENDER, *A partial k-arboretum of graphs with bounded treewidth*, Theoret. Comput. Sci., 209 (1998), pp. 1–45.
[5] H. L. BODLAENDER, R. G. DOWNEY, M. R. FELLOWS, M. T. HALLETT, AND H. T. WAREHAM, *Parameterized complexity analysis in computational biology*, Computer Applications in the Biosciences, 11 (1995), pp. 49–57.
[6] H. L. BODLAENDER AND R. H. MÖHRING, *The pathwidth and treewidth of cographs*, SIAM J. Discrete Math., 6 (1993), pp. 181–188.
[7] J. A. BONDY, *Basic graph theory: Paths and circuits*, in Handbook of Combinatorics, 1, R. L. Graham, M. Grötschel, and L. Lovász, eds., Elsevier, Amsterdam, 1995, pp. 3–110.
[8] A. BRANDSTÄDT, V. B. LE, and J. P. SPINRAD, *Graph classes: A survey*, SIAM Monogr. Discrete Math. Appl., SIAM, Philadelphia, PA, 1999.
[9] L. CAI, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Technical report, Department of Computer Science, The Chinese University of Hong Kong, Shatin New Territories, Hong Kong, 1995.
[10] P. Z. CHINN, J. CHVÁTALOVÁ, A. K. DEWDNEY, AND N. E. GIBBS, *The bandwidth problem for graphs and matrices—a survey*, J. Graph Theory, 6 (1982), pp. 223–254.
[11] D. G. CORNEIL, Y. PERL, AND L. K. STEWART, *A linear recognition algorithm for cographs*, SIAM J. Comput., 14 (1985), pp. 926–934.
[12] N. D. DENDRIS, L. M. KIROUSIS, AND D. M. THILIKOS, *Fugitive-search games on graphs and related parameters*, Theoret. Comput. Sci., 172 (1997), pp. 233–254.
[13] J. A. ELLIS, I. H. SUDBOROUGH, AND J. TURNER, *The vertex separation and search number of a graph*, Inform. and Comput., 113 (1994), pp. 50–79.
[14] F. V. FOMIN AND N. N. PETROV, *Pursuit-evasion and search problems on graphs*, Congr. Numer., 122 (1996), pp. 47–58.
[15] M. FRANKLIN, Z. GALIL, AND M. YUNG, *Eavesdropping games: A graph-theoretic approach to privacy in distributed systems*, in Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, Palo Alto, CA, 1993, pp. 670–679.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[17] P. A. Golovach, *Node-search and search number of a combination of graphs*, Vestnik Leningrad. Univ. Math. Mekh. Astronom., 1990, OVYR vyp. 2, 90–91, 122 (in Russian); translation in Vestnik Leningrad. Univ. Math., 23 (1990), pp. 53–55.

[18] P. A. Golovach, *Vertex separation sum of a graph*, Diskret. Mat., 9 (1997), pp. 86–91 (in Russian).

[19] H. Kaplan, R. Shamir, and R. E. Tarjan, *Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping*, in Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1994, pp. 780–791.

[20] L. M. Kirousis and C. H. Papadimitriou, *Interval graphs and searching*, Discrete Math., 55 (1985), pp. 181–184.

[21] L. M. Kirousis and C. H. Papadimitriou, *Searching and pebbling*, Theoret. Comput. Sci., 47 (1986), pp. 205–218.

[22] D. Kuo and G. J. Chang, *The profile minimization problem in trees*, SIAM J. Comput., 23 (1994), pp. 71–81.

[23] A. S. LaPaugh, *Recontamination does not help to search a graph*, J. ACM, 40 (1993), pp. 224–245.

[24] T. Lengauer, *Black-white pebbles and graph separation*, Acta Inform., 16 (1981), pp. 465–475.

[25] F. S. Makedon, C. H. Papadimitriou, and I. H. Sudborough, *Topological bandwidth*, SIAM J. Algebraic Discrete Methods, 6 (1985), pp. 418–444.

[26] F. S. Makedon and I. H. Sudborough, *On minimizing width in linear layouts*, Discrete Appl. Math., 23 (1989), pp. 243–265.

[27] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou, *The complexity of searching a graph*, J. ACM, 35 (1988), pp. 18–44.

[28] R. H. Möhring, *Graph problems related to gate matrix layout and PLA folding*, in Computational Graph Theory, Computing Suppl. 7, E. Mayr, H. Noltemeier, and M. Sysło, eds., Springer-Verlag, Berlin, 1990, pp. 17–51.

[29] T. D. Parsons, *Pursuit-evasion in a graph*, in Theory and Application of Graphs, Y. Alavi and D. R. Lick, eds., Springer Verlag, Berlin, 1976, pp. 426–441.

[30] N. N. Petrov, *Some extremal search problems on graphs*, Differential Equations, 18 (1982), pp. 591–595. Translation from Differ. Uravn 18, (1982) pp. 821–827.

[31] N. N. Petrov, *Pursuit problems with no information concerning the evader*, Differential Equations, 18 (1983), pp. 944–948. Translation from Differ. Uravn., 18 (1982), pp. 1345–1352.

[32] B. Reed, *Treewidth and tangles: A new connectivity measure and some applications*, in Surveys in Combinatorics, R. A. Bailey, ed., Cambridge University Press, Cambridge, UK, 1997, pp. 87–162.

[33] N. Robertson and P. D. Seymour, *Graph minors—a survey*, in Surveys in Combinatorics, I. Anderson, ed., Cambridge University Press, Cambridge, UK, 1985, pp. 153–171.

[34] K. Sugihara and I. Suzuki, *Optimal algorithms for a pursuit-evasion problem in grids*, SIAM J. Discrete Math., 2 (1989), pp. 126–143.

[35] R. Thomas, *Tree decompositions of graphs*, Lecture notes, Georgia Institute of Technology, Atlanta, GA, 1996.

[36] Y. S. Stamatiou and D. M. Thilikos, *Monotonocity and Inert Fugitive Search Games*, Technical Report LSI-99-35-R, Departament de Llenguatges i Sistemes Informátics, Universitat Politécnica de Catalunya, Spain, 1999.

SIAM J. DISCRETE MATH.
Vol. 13, No. 4, pp. 465–491

# LEARNING DETERMINISTIC FINITE AUTOMATA FROM SMALLEST COUNTEREXAMPLES[*]

ANDREAS BIRKENDORF[†], ANDREAS BÖKER[‡], AND HANS ULRICH SIMON[†]

**Abstract.** We show in this paper (which appeared in a preliminary form as an extended abstract in [*Proceedings of the 9th International ACM–SIAM Symposium on Discrete Algorithms*, ACM, 1998]) that deterministic finite automata (DFAs) with $n$ states and input alphabet $\Sigma$ can efficiently be learned from less than $|\Sigma|n^2$ smallest counterexamples. This improves on an earlier result of Ibarra and Jiang who required $|\Sigma|n^3$ smallest counterexamples. We present a general strategy which learns a finite concept class $\mathcal{F}$ from $\lfloor \log \mathcal{F} \rfloor$ smallest counterexamples (but not necessarily efficiently). An application to DFAs with at most $n$ states shows that $(1 + o(1))|\Sigma|n \log n$ smallest counterexamples are sufficient (if efficiency is not an issue). We show next that the special DFAs operating on input words of an arbitrary but fixed length (the so-called leveled DFAs) are efficiently learnable from $(1 + o(1))|\Sigma|n \log n$ smallest counterexamples. This improves on an earlier result of Ibarra and Jiang who required $|\Sigma|n^2$ smallest counterexamples. Furthermore, we present a general lower bound on the number of smallest counterexamples (required by any learning algorithm). This bound can be stated in terms of a (new) combinatorial dimension associated with the target class. A computation of this dimension for leveled or arbitrary DFAs leads to a lower bound of the form $(\frac{1}{4} + o(1))|\Sigma|n \log n$. This bound matches the aforementioned upper bounds modulo a constant of approximately 4. Finally, we present a general conversion of algorithms learning from smallest counterexamples into algorithms performing self-directed learning. For the particular classes of leveled or arbitrary DFAs, this conversion leads to self-directed learners making the smallest possible number of mistakes (modulo a constant of approximately 4). A similar remark is valid for the class of multiplicity automata (MAs).

**Key words.** learning from smallest counterexamples, self-directed learning, deterministic finite automata, multiplicity automata

**AMS subject classification.** 68Q32

**PII.** S0895480198340943

**1. Introduction.** In section 1.1, we define the model of learning from smallest counterexamples and some related models. Section 1.2 recalls earlier results on the learning complexity of deterministic finite automata. We mention, in particular, the work of Ibarra and Jiang [16], which is closely related to our work. The main results of this paper, including our improvements on the work of Ibarra and Jiang, are summarized in section 1.3. A detailed outline of the paper is given in section 1.4.

**1.1. The learning model.** Let $\mathcal{F}$ be a *concept class* over *domain* $X$, i.e., a class containing functions of the form $f : X \rightarrow \{0, 1\}$. In what follows, the elements of $\mathcal{F}$ are called *concepts*, and the elements of $X$ are called *instances*. In this paper, we are concerned with the problem of "learning" an unknown target concept $f_* \in \mathcal{F}$. The learning model we consider is related to the query-learning model of Angluin (see [3]). In this model, the learning algorithm $A$ gathers information about $f_*$ by asking a teacher queries. The most popular queries are equivalence-queries (EQs) and membership-queries (MQs). An EQ is issued by $A$ along with a hypothesis

[*]Received by the editors June 19, 1998; accepted for publication (in revised form) February 28, 2000; published electronically October 6, 2000. This research was supported by the Deutsche Forschungsgemeinschaft, grant Si 498/3-2, and by the German–Israeli Foundation for Scientific Research and Development, grant I-403-001.06/95.

http://www.siam.org/journals/sidma/13-4/34094.html

[†]Department of Mathematics, Ruhr-University Bochum, D-44780 Bochum, Germany (birkendo@lmi.ruhr-uni-bochum.de, simon@lmi.ruhr-uni-bochum.de).

[‡]Department of Computer Science, University of Dortmund, D-44221 Dortmund, Germany.

$f$. If $f$ is correct, the teacher answers "YES." Otherwise, the teacher presents a counterexample, that is, an instance $x \in X$ on which $f$ and $f_*$ disagree. An MQ is issued by $A$ along with an instance $x \in X$ and answered with $f_*(x)$. If $A$ has written a description of hypothesis $f$ or instance $x$ on a special query tape, the corresponding answer is given in one time step. We say that $A$ *learns $\mathcal{F}$ from $m$ queries* if, for each target concept $f_* \in \mathcal{F}$, $A$ asks at most $m$ queries to the teacher and finally comes up with a description of $f_*$. In addition, $A$ is called a *polynomial (or efficient) learning algorithm* when $m$ and the run time of $A$ are both bounded by a polynomial depending on parameters[1] describing the complexity of $f_*$ and on the length of the longest counterexample ever presented by the teacher.[2] If only EQs are used, we speak of *learning from counterexamples*. If both EQs and MQs are used, we speak of *learning from a minimum adequate teacher*.

In this paper, we focus on a special variant of learning from counterexamples, where each EQ is answered deterministically. We speak of *learning from smallest counterexamples* (*SC-learning*, for short), if the teacher always presents the "smallest" instance $x \in X$ where hypothesis $f$ and target function $f_*$ disagree. The word "smallest" refers to some given linear ordering on domain $X$, known by the learning algorithm.

The following linear orderings on the set $\Sigma^\star$ of words over a finite alphabet $\Sigma$ are relevant in this paper. For notational convenience, we assume that $\Sigma = \{0, \ldots, |\Sigma| - 1\}$. The letters (digits) have their natural ordering. We denote the empty word by $\lambda$. Word $x \in \Sigma^\star$ is called a *prefix* of word $w \in \Sigma^\star$ if $w$ can be written as $xz$ for some $z \in \Sigma^\star$. If additionally $z \neq \lambda$, $x$ is called a *proper prefix* of $w$. Two words $v, w$ can always be written as $v = xv'$ and $w = xw'$, where $x \in \Sigma^\star$ is the greatest common prefix of $v$ and $w$. We say that $v$ is *lexicographically smaller* than $w$ (denoted as $v <_{\mathrm{LEX}} w$) if either $v$ is a proper prefix of $w$ or $v'$ starts with a smaller letter than $w'$ in the above decomposition. (That's the ordering used in a lexicon.) We say that $v$ is *canonically smaller* than $w$ (denoted as $v < w$) if $v$ is shorter than $w$ or, in the case of equal length, $v$ is lexicographically smaller than $w$. We note that the canonical ordering is more natural for learning applications, because short counterexamples are likely to be more helpful than long ones.

## 1.2. Learnability of deterministic finite automata.

A particular class $\mathcal{F}$ that was intensively studied in the past is the class of concepts representable by deterministic finite automata (DFAs), i.e., the class of characteristic functions of regular languages. The complexity of the target DFA is measured by two parameters: the number $n$ of its states and the size of input alphabet $\Sigma$. As shown in [3], DFAs cannot be learned polynomially from MQs alone. It was furthermore shown in [4] (using the technique of "approximate fingerprints") that also EQs alone are not sufficient. However, Angluin has presented an efficient algorithm that learns DFAs with a minimum adequate teacher (see [2]). This algorithm was later improved by Rivest and Schapire whose algorithm is simpler and needs fewer MQs (see [21]).

DFAs are also polynomially SC-learnable: In [16], Ibarra and Jiang develop an algorithm, which efficiently learns DFAs with at most $n$ states and input alphabet $\Sigma$ from $\Theta(|\Sigma|n^3)$ smallest counterexamples. Ibarra and Jiang's algorithm contains a

[1] These parameters will be formally specified for the concrete learning problem that we consider.

[2] If the teacher always presents a counterexample of shortest length, this additional parameter is not needed.

subroutine that SC-learns so-called *leveled* DFAs, [3] which operate on input words of an arbitrary but fixed length. This subroutine requires $|\Sigma|n^2$ smallest counterexamples.

**1.3. Main results.** In this paper (which appeared in a preliminary form as extended abstract in [9]), we improve on the results of Ibarra and Jiang as follows.

- We show that general DFAs are efficiently SC-learnable from less than $|\Sigma|n^2$ smallest counterexamples. The progress is achieved mainly by avoiding the inefficient reduction of Ibarra and Jiang, who use the procedure for leveled DFAs as a subroutine to solve the general case. We introduce the concept of Nerode diagrams which allows us to solve the general case directly.
- We show that leveled DFAs are efficiently SC-learnable from $(1 + o(1)) \cdot |\Sigma| \cdot n \log n$ smallest counterexamples. This is accomplished by a new method (dealing with so-called selection and elimination operators) which makes the binary search paradigm applicable to SC-learners.

These results are complemented by various other new results.

- We present a general strategy which learns a finite concept class $\mathcal{F}$ from $\lfloor \log \mathcal{F} \rfloor$ smallest counterexamples (but not necessarily efficiently). An application to DFAs with at most $n$ states shows that $(1+o(1))|\Sigma|n \log n$ smallest counterexamples are sufficient (if efficiency is not an issue).
- We present a general lower bound on the number of smallest counterexamples (required by any learning algorithm). This bound can be stated in terms of a (new) combinatorial dimension associated with the target class. A computation of this dimension for leveled or arbitrary DFAs leads to a lower bound of the form $(\frac{1}{4} + o(1))|\Sigma|n \log n$. This bound matches the aforementioned upper bounds modulo a constant of approximately 4.
- We present a general conversion of algorithms learning from smallest counterexamples into algorithms performing self-directed learning (SD-learning). For the particular classes of leveled or arbitrary DFAs, this conversion leads to SD-learners making the smallest possible number of mistakes (modulo a constant of approximately 4). A similar remark is valid for the class of multiplicity automata (MAs).

**1.4. Outline of the paper.** The first part of the paper, comprising sections 2, 3, and 4, is devoted to the aforementioned improvements on the work of Ibarra and Jiang.

In section 3, we present a new algorithm `REDIRECT` which efficiently learns arbitrary DFAs with at most $n$ states and input alphabet $\Sigma$ from less than $|\Sigma|n^2$ smallest counterexamples.

Furthermore, we present in section 4 a new algorithm `LEVELED-REDIRECT` for SC-learning leveled DFAs. `LEVELED-REDIRECT` requires $(1 + o(1))|\Sigma|n \log n$ smallest counterexamples.

In order to design `REDIRECT` and `LEVELED-REDIRECT`, we introduce in section 2 a new data structure for regular languages: *Nerode diagrams* and *partially correct subdiagrams*. It is a very efficient and useful tool when dealing with smallest counterexamples. `REDIRECT` will "grow" an (initially trivial) partially correct subdiagram until it is isomorphic to the full transition diagram of the target DFA. The correct edges in the diagram are found by a sort of exhaustive search in the case of arbitrary

[3]Leveled DFAs, as defined in section 4, are basically equivalent to a data structure known as ordered binary decision diagrams (OBDDs) in the literature (see [19, 1, 11]). In this paper, it is more convenient to use the same terminology as for DFAs.

DFAs, and by a sort of binary search in the case of leveled DFAs. In order to prove the correctness of our algorithms, we present a series of structural results concerning Nerode diagrams and their partially correct subdiagrams. The new data structure and the structural insights gained by its analysis may be of independent interest.

In the second part of the paper, comprising sections 5, 6, 7, and 8, the results of the first part are complemented in many ways.

We present in section 5 a general lower bound on the number of smallest counterexamples (required by any learning algorithm). This bound can be stated in terms of a (new) combinatorial dimension associated with the target class. A computation of this dimension for leveled or arbitrary DFAs leads to a lower bound of the form $(\frac{1}{4} - o(1))|\Sigma|n \log n$. The new dimension may be of interest beyond its application to DFAs.

In section 6, we present a general strategy which learns a finite concept class $\mathcal{F}$ from $\lfloor \log \mathcal{F} \rfloor$ smallest counterexamples (but not necessarily efficiently). The idea of this strategy is to perform a clever majority vote on the instance space, tailored to the conditions valid in the SC-learning scenario. In contrast to the well-known halving strategy in the model of learning from (arbitrary) counterexamples, our strategy realizes proper learning (i.e., all hypotheses are members of $\mathcal{F}$). An application to leveled or arbitrary DFAs with at most $n$ states and input alphabet $\Sigma$ shows that $(1 + o(1))|\Sigma|n \log n$ smallest counterexamples are sufficient (if efficiency is not an issue). This upper bound matches the aforementioned lower bound modulo a constant of approximately 4.

In section 7, we describe a general conversion of SC-learning algorithms into algorithms performing SD-learning.[4] For the particular classes of leveled or arbitrary DFAs, this conversion leads to SD-learners making the smallest possible number of mistakes (modulo a constant of approximately 4).

Recently (and building on our results from [9]), Forster presented an efficient algorithm which SC-learns the class of MAs with at most $n$ states and alphabet $\Sigma$ from $2 + |\Sigma|n(n + 1)$ smallest counterexamples [18]. As outlined in section 8, we can convert this algorithm into an SD-learner for MAs making the smallest possible number of mistakes (up to a constant factor).

## 2. Nerode diagrams and partially correct subdiagrams.

We assume some familiarity with basic concepts in formal language theory as given by any standard book (e.g., [15]). In what follows, we recall some classical notions concerning DFAs and define a new data structure for their representation.

Let $M = (Q, \Sigma, \delta, q_0, Q_+)$ be a DFA, where $Q$ denotes the set of states, $\Sigma$ the finite input alphabet, $\delta$ the transition function, $q_0$ the designated initial state, and $Q_+$ the set of accepting states. We will represent $M$ by its state diagram which visualizes the states as nodes and the transitions as edges. An example is shown in Figure 2.1. As usual, we say that $M$ *accepts* a word $w$ if the path $P_M(w)$, which starts in the initial state and follows the edges labeled by the letters of $w$, ends at an accepting state of $M$. Let $L(M)$ denote the language of words accepted by $M$. We associate with $M$ the indicator function $M(w)$ with value 1 if $w \in L(M)$, and value 0 otherwise.

Two DFAs are called *equivalent* if they accept the same language. Let $M$ be a minimum DFA for $L$, i.e., $L = L(M)$ and no other DFA with this property has fewer states than $M$. Then $M$ is uniquely determined (up to isomorphism), and its states are in a one-to-one correspondence to the classes given by the following equivalence

[4]A brief definition of this learning model is given in section 7.

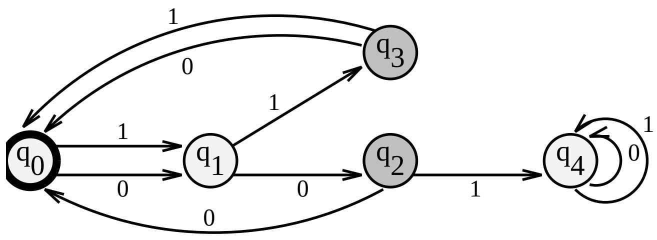


FIG. 2.1. *State diagram of a DFA* $M = (\{q_0, q_1, \dots, q_4\}, \{0,1\}, \delta, q_0, \{q_2, q_3\})$ *accepting language* $L = (\{0,1\}\cdot\{00, 1\cdot\{0,1\}\})^*\cdot\{0,1\}\cdot\{0,1\}$. *Accepting states are drawn darker than nonaccepting states. The initial state is encircled with a thick line.*

relation $\equiv_L$, called the *Nerode relation*:

$$\forall s,t \in \Sigma^\star : \quad [s \equiv_L t \quad \Longleftrightarrow \quad (\forall\, z \in \Sigma^\star : sz \in L \iff tz \in L)] .$$

We call a word $r \in \Sigma^\star$ the *minimum representative* of its Nerode class $[r]_L = \{s \in \Sigma^\star \mid s \equiv_L r\}$ if $r$ is the smallest word in $[r]_L$. The set of all minimum representatives is denoted by $R(L)$.

It is easy to see that $ra \in R(L)$, where $r \in \Sigma^\star$ and $a \in \Sigma$, implies that $r \in R(L)$. Inductively, we get the following lemma.

LEMMA 2.1. *$R(L)$ is prefix-closed, i.e., if $r \in R(L)$, then $r' \in R(L)$ for each prefix $r'$ of $r$.*

The *Nerode diagram* $D_L$ of $L$ is defined as follows. Its nodes are in one-to-one correspondence to the words in $R(L)$ and denoted by $q_r$ for $r \in R(L)$. We call $q_r$ "accepting" if $r \in L$ and "nonaccepting" otherwise. The edge is directed from $q_r$ and labeled by $a$ points to $q_s$, where $s$ is the minimum representative of $[ra]_L$. It is denoted by $q_r \xrightarrow{a} q_s$ in what follows. An example is shown in Figure 2.2(a).

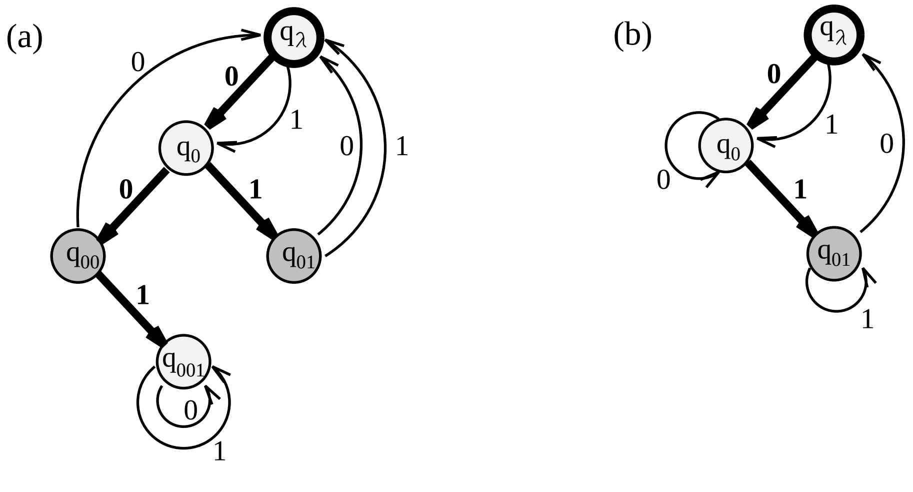


FIG. 2.2. (a) *The Nerode diagram of language L from Figure* 2.1. $R(L) = \{\lambda, 0, 00, 01, 001\}$. *$F$-edges are drawn thick, and $B$-edges are drawn thin.* (b) *Partially correct subdiagram of the Nerode diagram in* (a).

Obviously, $D_L$ represents the minimum DFA for $L$. The edges of $D_L$ decompose according to the following definition. Edge $q_r \xrightarrow{a} q_s$ is called *backward edge* (or $B$-edge) if $s < ra$. Otherwise, it is called *forward edge* (or $F$-edge). In the latter case,

$s = ra$. Since $R(L)$ is prefix-closed, it follows that the $F$-edges form the prefix-tree of $R(L)$ (as illustrated in Figure 2.2(a)). Formally, the *prefix-tree* associated with a prefix-closed set $R \subseteq \Sigma^*$ is the tree with node set $R$ and an edge directed from $r$ to $s$ and labeled by $a \in \Sigma$ if $s = ra$.

Given a prefix-closed subset $R \subseteq R(L)$, a *partially correct subdiagram* of $D_L$ with respect to $R$ consists of the prefix-tree of $R$ and additional $B$-edges not necessarily identical to the $B$-edges of $D_L$. The partitioning of nodes $q_r$ into "accepting" and "nonaccepting" nodes is done as for $D_L$. Figure 2.2(b) shows an example.

A DFA $M$, represented by a partially correct subdiagram of $D_L$ with respect to $R$, is called *partially correct* for $L$ with respect to $R$.[5] (In particular, $M$ is correct for each $r \in R$, i.e., $M(r) = 1$ iff $r \in L$.)

If path $P_M(w)$ ends in state $q_r$, we say that $q_r$ *is visited by* $w$ and call $w$ a *visitor* of $q_r$. We close this section with the following obvious result.

LEMMA 2.2 (visitor lemma). *All visitors $w$ of $q_r$ satisfy $r \le w$ with equality iff $P_M(w)$ contains $F$-edges exclusively.*

## 3. The SC-learning algorithm for DFAs.

Let $M_*$ be the unknown target DFA, accepting language $L_* = L(M_*)$. Throughout the remainder of this section, let $M$ be a DFA, partially correct for $L_*$ with respect to a prefix-closed subset $R \subseteq R(L_*)$, but $L(M) \neq L_*$. We denote the smallest counterexample for $M$ by $\mathrm{mincex}(M, M_*)$. In order to convert the state diagram of $M$ into the correct Nerode diagram of $L_*$, one has to overcome two difficulties.

1. The prefix-tree of $R$ has to be extended to the complete prefix-tree of $R(L_*)$.
2. *Wrong $B$-edges*, i.e., $B$-edges of the form $q_r \xrightarrow{a} q_s, r, s \in R, s < ra, s \not\equiv_{L_*} ra$ must be redirected such as to point to the correct node $q_{s'}$ with $s' \equiv_{L_*} ra$.

The remainder of this section is organized as follows. We first present two auxiliary results (Lemmas 3.1 and 3.2) describing how wrong $B$-edges can be detected. A description of the SC-learning algorithm `REDIRECT` follows. Finally, we prove that `REDIRECT` is correct and analyze its time complexity, where the more troublesome details of the correctness proof and of the time analysis are encapsulated in sections 3.1 and 3.2, respectively. The reader who is already satisfied with the high-level analysis may decide to skip these two subsections without loss of continuity.

LEMMA 3.1. *Let $c = \mathit{mincex}(M, M_*)$. Then the following holds.*

(i) *$P_M(c)$ contains at least one $B$-edge.*

(ii) *The first $B$-edge $q_r \xrightarrow{a} q_s$ on $P_M(c)$ is wrong, i.e., $s \not\equiv_{L_*} ra$.*

*Proof.*

(i) If $P_M(r)$ contains $F$-edges exclusively, then $r \in R$. Because $M$ is correct on $R$, $r$ cannot be a counterexample. Thus, $P_M(c)$ must contain at least one $B$-edge.

(ii) If $q_r \xrightarrow{a} q_s$ is the first $B$-edge on $P_M(c)$, $c$ can be written as $c = raz$ for some $z \in \Sigma^\star$. By the definition of $B$-edges, $s < ra$. Thus $sz < raz$, and $sz$ (being smaller than the smallest counterexample) cannot be a counterexample itself. Thus, $M_*(sz) = M(sz)$. Since $M$ does not distinguish between $ra$ and $s$, $M(sz) = M(raz)$. Since $raz$ is a counterexample, $M(raz) \neq M_*(raz)$. Thus, $M_*(sz) = M(sz) = M(raz) \neq M_*(raz)$, which implies that $ra \not\equiv_{L_*} s$. $\square$

Let $c$ be the counterexample which (by means of Lemma 3.1) showed that the $B$-edge directed from $q_r$ to $q_s$ and labeled by $a$ is wrong. It is a straightforward idea to redirect this $B$-edge (by way of trial and error) to another state $q_{s'}$, where

[5]We will sometimes omit the expression "for $L$ with respect to $R$" when the sets $L$ and $R$ are clear from context.

$s' \in R, s' < ra$. We call the resulting DFA the TEST-DFA for $B$-edge $q_r \xrightarrow{a} q_{s'}$ and denote it by $M(r,a,s')$. Note that the state diagrams of $M$ and $M(r,a,s')$ share the same prefix-tree structure and differ only on a single $B$-edge. Thus, the partial correctness of $M$ carries over to $M(r,a,s')$. The following result shows that the redirected $B$-edge is still wrong unless the TEST-DFA receives a counterexample $c'$ that is greater than $c$.

LEMMA 3.2. *Let $c = mincex(M, M_*)$, $q_r \xrightarrow{a} q_s$ be the first $B$-edge on $P_M(c)$, $q_{s'}$ be a state of $M$ with $s' < ra$, $M' = M(r,a,s')$ be the TEST-DFA for $B$-edge $q_r \xrightarrow{a} q_{s'}$, and $c' = mincex(M', M_*)$. If $c' \leq c$, then $q_r \xrightarrow{a} q_{s'}$ is the first $B$-edge on $P_{M'}(c')$ and $s' \not\equiv_{L_*} ra$.*

*Proof.* If $c' = c$, then $q_r \xrightarrow{a} q_{s'}$ is certainly the first $B$-edge on $P_{M'}(c')$. If $c' < c = \text{mincex}(M, M_*)$, then $M(c') = M_*(c') \neq M'(c')$. $P_{M'}(c')$ must therefore contain $B$-edge $q_r \xrightarrow{a} q_{s'}$. (Otherwise, $M$ and $M'$ would coincide on $c'$.) The following considerations are illustrated in Figure 3.1.

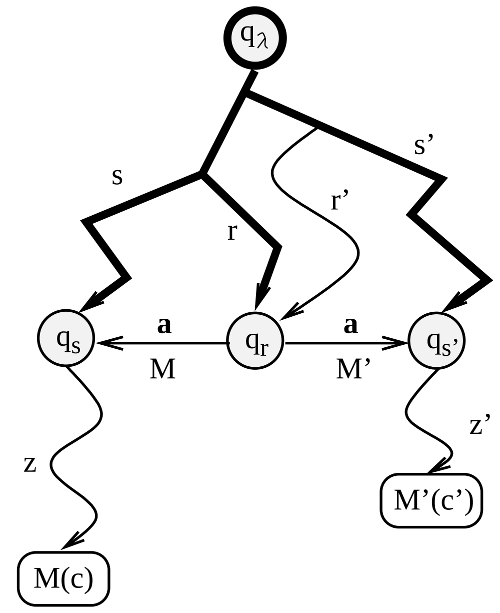


FIG. 3.1. *Illustration for the proof of Lemma* 3.2.

Let us assume for the sake of contradiction that $q_r \xrightarrow{a} q_{s'}$ is not the first $B$-edge on $P_{M'}(c')$. Hence, $c'$ can be written as $c' = r'az'$, where $r' > r$ is a visitor of $q_r$. Let $c = raz$ be the corresponding decomposition of $c$ along the first $B$-edge $q_r \xrightarrow{a} q_s$ on $P_M(c)$. From $r' > r$ and $r'az' = c' < c = raz$, we obtain $z' < z$. We consider the word $raz'$. Since $raz' < r'az' < raz$, $M$ and $M'$ are correct on $raz'$, and $M$ is correct on $r'az'$. On the other hand, we obtain

$$M(raz') = M(r'az') = M_*(r'az') \neq M'(r'az') = M'(raz'),$$

because neither $M$ nor $M'$ can distinguish between $r$ and $r'$. It follows that either $M$ or $M'$ must be wrong on $raz'$—a contradiction. An application of Lemma 3.1 yields $s' \not\equiv_{L_*} ra$. □

TABLE 3.1
*Round $k$ of Algorithm* `REDIRECT`.

```
Step 1  Find the first $B$-edge on $P_{M_k}(c_k)$, say $q_r \xrightarrow{a} q_s$.
Step 2  Eliminate $s$ from $C(r,a)$. (* cf. Lemma 3.1 *)
Step 3  SAMEROUND $\leftarrow$ TRUE.
Step 4  while $C(r,a) \neq \emptyset$ and SAMEROUND do
        Step 4.1  $s' \leftarrow$ next element in $C(r,a)$
        Step 4.2  $M'_k \leftarrow M_k(r,a,s')$.
        Step 4.3  $c'_k \leftarrow \mathrm{mincex}(M'_k, M_*)$.
        Step 4.4  if $c'_k \le c_k$ then eliminate $s'$ from $C(r,a)$. (* cf. Lemma 3.2 *)
                                  else SAMEROUND $\leftarrow$ FALSE.
Step 5  if SAMEROUND then
        Step 5.1  $M_{k+1} \leftarrow$ EXTEND$(M_k, ra)$.
        Step 5.2  $R_{k+1} \leftarrow R_k \cup \{ra\}$.
        Step 5.3  $c_{k+1} = \mathrm{mincex}(M_{k+1}, M_*)$.
        Step 5.4  $\forall b \in \Sigma :\ C(ra,b) \leftarrow \{t \in R_{k+1} |\ t < rab\}$.
        Step 5.5  $\forall t \in R_k, \forall b \in \Sigma :$ if $ra < tb$ then insert $ra$ into $C(t,b)$.
        else $M_{k+1} \leftarrow M'_k, R_{k+1} \leftarrow R_k$, $c_{k+1} = c'_k$.

Procedure EXTEND$(M_k, ra)$:
Step E1  Insert the new state $q_{ra}$ into the state diagram of $M_{k+1}$.
Step E2  Replace $B$-edge $q_r \xrightarrow{a} q_s$ by $F$-edge $q_r \xrightarrow{a} q_{ra}$.
Step E3  For each $b \in \Sigma$, create the $B$-edge directed from $q_{ra}$ and labeled by $b$.
         Let it point to the same node as the edge directed from $q_s$ and labeled by $b$.
Step E4  if $c_k = ra$ then declare $q_{ra}$ as "accepting" iff $q_s$ is "nonaccepting"
                       else declare $q_{ra}$ as "accepting" iff $q_s$ is "accepting".
```

Let $\mathrm{DFA}_{\Sigma,n}$ denote the class of DFAs with at most $n$ states and input alphabet $\Sigma$. We are now prepared to describe the algorithm `REDIRECT`[6] which learns an unknown DFA $M_* \in \mathrm{DFA}_{\Sigma,n}$ from less than $n^2$ smallest counterexamples. We assume without loss of generality that $M_*$ has exactly $n$ states. (DFAs in $\mathrm{DFA}_{\Sigma,n}$ with $n$ states represent the worst case within the analysis of `REDIRECT`.)

`REDIRECT` proceeds in rounds. In one round, the following is achieved. Round $k$ is entered with a partially correct DFA $M_k$ for $L_*$ with respect to a prefix-closed subset $R_k \subseteq R(L_*)$ and its smallest counterexample $c_k$. During round $k$, `REDIRECT` either stops with a DFA equivalent to $M_*$ or enters the next round with a partially correct DFA $M_{k+1}$ and its smallest counterexample $c_{k+1}$. In the following informal description (which can be compared with the more formal description in Table 3.1), we focus on a round $k$ not leading to the stopping condition.

During each round $k$, `REDIRECT` keeps track of the sets $C(r,a) \subseteq \{s \in R_k |\ s < ra\}$ containing all candidates for the endpoint of the $B$-edge directed from $q_r$ and labeled by $a$. As `REDIRECT` proceeds, it detects wrong $B$-edges and eliminates the corresponding candidates from $C(r,a)$.

1. The DFA $M_k$ obtained from the preceding round is inspected. According to Lemma 3.1, a wrong $B$-edge, say $q_r \xrightarrow{a} q_s$, is found on the computation path of $M_k$ for counterexample $c_k = \mathrm{mincex}(M_k, M_*)$. This leads to the safe elimination of $s$ from $C(r,a)$.

[6]The redirection of $B$-edges is one of its central features.

2. `REDIRECT` considers TEST-DFAs. If $C(r,a) \neq \emptyset$, it picks a candidate $s'$ from $C(r,a)$ and inspects the corresponding TEST-DFA $M'_k = M_k(r,a,s')$. According to Lemma 3.2, this either leads to a counterexample $c'_k = \text{mincex}(M'_k, M_*) > c_k$ or to the safe elimination of $s'$ from $C(r,a)$. In the former case, `REDIRECT` enters round $k+1$ with $M_{k+1}$ equal to $M'_k$. In the latter case, it proceeds with the next candidate from $C(r,a)$.

If all elements of $C(r,a)$ are eliminated without receiving a counterexample $c'_k > c_k$, `REDIRECT` decides to extend $R_k$ to $R_{k+1} = R_k \cup \{ra\}$. In order to define $M_{k+1}$ appropriately, `REDIRECT` calls procedure `EXTEND`$(M_k, ra)$. The task of this procedure is to include the new state $q_{ra}$ in the state diagram of $M_k$. Afterwards the state diagram contains the prefix-tree of $R_k \cup \{ra\}$. The modifications are performed carefully in order to guarantee correct accepting behavior on $ra$ (which is necessary to achieve partial correctness), and to modify $L(M_k)$ as little as possible. These subtle details of procedure EXTEND will be discussed subsequently. Finally, round $k+1$ is entered.

There is still one missing detail. We have to explain how `REDIRECT` gets started with a first partially correct DFA $M_1$. Let $M_1^0$ and $M_1^1$ be the DFAs shown in Figure 3.2.

M$_1^0$
q$_\lambda$
0
1
M$_1^1$
q$_\lambda$
0
1

FIG. 3.2. *Initial DFAs $M_1^0$ and $M_1^1$ for* $\Sigma = \{0,1\}$.

Their state diagrams consist of a single state $q_\lambda$ and the $B$-edges $q_\lambda \xrightarrow{a} q_\lambda$ for all $a \in \Sigma$. State $q_\lambda$ is accepting in $M_1^1$ and nonaccepting in $M_1^0$. Thus, $M_1^0$ and $M_1^1$ are acceptors of $\emptyset$ and $\Sigma^\star$, respectively. Note that $M_1^0$ is partially correct if $M_*(\lambda) = 0$. Otherwise, $M_1^1$ is partially correct. Thus, $M_1$ can be set to $M_0^{M_*(\lambda)}$ at the expense of one counterexample used to determine $M_*(\lambda)$. Round 1 can be entered with $M_1$, $c_1 = \text{mincex}(M_1, M_*)$, $R_1 = \{\lambda\}$, and $C(\lambda, a) = \{\lambda\}$ for all $a \in \Sigma$.

We now turn to the analysis of `REDIRECT`, which builds on the following auxiliary result.

LEMMA 3.3. *All DFAs $M_k$, $k \geq 1$, are partially correct.*

The proof of this lemma is somewhat tedious. It is postponed to section 3.1, because we want to present the high-level ideas of the analysis more coherently.

Note that Lemma 3.3 implies that `REDIRECT` creates only the $n$ states $q_u$ for $u \in R(L_*)$. It is now easy to show the following theorem.

THEOREM 3.4. `REDIRECT` *polynomially SC-learns $DFA_{\Sigma,n}$ from at most* $1 + |\Sigma|n^2 - |\Sigma|n$ *smallest counterexamples.*

*Proof.* The number of counterexamples is bounded by 1 plus the number of eliminated candidates from sets of the form $C(r,a)$, because the first counterexample is used to define $M_1$ properly, and each new counterexample leads to the elimination of a candidate. The total size of all candidate sets equals the number of candidate $B$-edges of the form $q_r \xrightarrow{a} q_s$, where $a \in \Sigma$, $r, s \in R(L_*)$, and $s < ra$. This number is obviously bounded by $|\Sigma|n^2$. Since the $|\Sigma|n$ candidates for the correct $B$-edges are not eliminated, the total number of eliminated candidates is at most $|\Sigma|n^2 - |\Sigma|n$. This leads to the desired bound on the number of counterexamples.

Since the run time per round is certainly polynomially bounded, it follows that the total run time is polynomially bounded. ∎

We conclude the current section with the proof of Lemma 3.3, given in section 3.1, and a refined time analysis, given in section 3.2. The reader not interested in these details may immediately pass to section 4.

**3.1. Proof of Lemma 3.3.** In this subsection, we prove the partial correctness of the $M_k$. We start with the following easy observations that hold for all $k \geq 1$.

1. $M_1$ is partially correct for $L_*$ with respect to $R_1 = \{\lambda\}$.
2. The state diagram of $M_k$ contains exactly the nodes $q_u$ for $u \in R_k$, and decomposes into the prefix-tree of $R_k$ and additional $B$-edges.
3. For all $u \in R_k$: $M_{k+1}(u) = M_k(u)$.
4. If $M_{k+1} = M_k(r, a, s')$, then $R_{k+1} = R_k$ and $c_{k+1} > c_k$.
5. If $M_{k+1} = M_k(r, a, s')$ and $M_k$ is partially correct for $L_*$ with respect to $R_k$, then $M_{k+1}$ is partially correct for $L_*$ with respect to $R_{k+1}$.
6. If $M_{k+1} = \texttt{EXTEND}(M_k, ra)$, then $R_{k+1} = R_k \cup \{ra\}$.

These observations fall short of proving the partial correctness of all $M_k$ by induction: the inductive step is only performed for $M_{k+1} = M_k(r, a, s')$—see observation 5—but not yet for $M_{k+1} = \texttt{EXTEND}(M_k, ra)$. Note that the partial correctness of $M_k$ carries over to $M_{k+1} = \texttt{EXTEND}(M_k, ra)$ if we can show that $ra \in R(L_*)$ and $M_{k+1}(ra) = M_*(ra)$.

We start with the proof of the latter assertion. (For technical reasons, we prove also that $c_{k+1} \geq c_k$.)

LEMMA 3.5. *If $M_{k+1} = \texttt{EXTEND}(M_k, ra)$, then $M_{k+1}(ra) = M_*(ra)$ and $c_{k+1} \geq c_k$.*

*Proof.* Remember that $c_k$ decomposes into $c_k = raz$ for some $z \in \Sigma^\star$. Thus, $c_k \geq ra$. We prove the lemma separately for the cases $c_k > ra$ and $c_k = ra$.

If $c_k > ra$, then $\texttt{EXTEND}(M, ra)$ was defined in such a way that states $q_s$ and $q_{ra}$ are equivalent.[7] It follows that $L(M_k) = L(M_{k+1})$. $M_k$ is correct on all words smaller than $c_k = \text{mincex}(M_k, M_*)$, in particular on $ra$. Thus, the same must hold for $M_{k+1}$ (implying $c_{k+1} \geq c_k$).

If $c_k = ra$, then $\texttt{EXTEND}(M, ra)$ was defined in such a way that $M_{k+1}(ra) \neq M_k(ra)$. Since $M_k$ is wrong on $ra = c_k = \text{mincex}(M_k, M_*)$, $M_{k+1}$ is correct on $ra$. For any word $w < c_k = ra$, the path $P_{M_k}(w)$ does not reach the part of the diagram that has been modified by procedure EXTEND. Thus, $M_{k+1}(w) = M_k(w) = M_*(w)$, where the latter equality follows from $w < c_k = \text{mincex}(M_k, M_*)$. Hence $c_{k+1} > c_k$. ∎

All that remains to be done is showing $ra \in R(L_*)$. The proof for this requires the following monotonicity property of the sequence $c_k$, given by observation 4 and Lemma 3.5.

COROLLARY 3.6. *The sequence $c_k$, $k \geq 1$, is nondecreasing.*

Using this monotonicity, we are able to show the following lemma.

LEMMA 3.7. *If $M_{k+1} = \texttt{EXTEND}(M_k, ra)$, then $ra \in R(L_*)$.*

*Proof.* Let us assume for the sake of contradiction that $ra \notin R(L_*)$, i.e., there exists a word $t \in \Sigma^\star$ satisfying $t < ra$ and $t \equiv_{L_*} ra$.

Let $q_w$ denote the state visited by $t$ in $M_k$. According to the visitor lemma, $w \leq t < ra$. Because all candidate sets are properly updated when a new state is

[7]As usual, states $q, q'$ of a DFA $M$ are called equivalent if the following holds for each $w \in \Sigma^\star$: $M$ started in $q$ accepts $w$ iff $M$ started in $q'$ accepts $w$.

created (see Steps 5.4 and 5.5 of `REDIRECT`), $w$ must belong to $C(r,a)$ as soon as $q_r$ and $q_w$ are both included. Since $r,w \in R_k$, $q_r \xrightarrow{a} q_w$ was a candidate $B$-edge before $C(r,a)$ became empty in round $k$. Thus, this $B$-edge was eliminated during some round $j \leq k$. Let $M'$ denote the DFA in round $j$ leading to the elimination of $w$ in Step 2 or Step 4.4 due to counterexample $c'$. Hence, $M'$ is either $M_j$ (and $c' = c_j$) or $M'$ is the TEST-DFA $M_j'$ for $B$-edge $q_r \xrightarrow{a} q_w$ (and $c' = c_j'$). We obtain $c' \leq c_k$ since $c_j \leq c_k$ and $c_j' \leq c_j \leq c_k$ by monotonicity. According to Lemma 3.2, $c'$ can be written as $c' = raz'$ for some $z' \in \Sigma^\star$, and we get

$$wz' \leq tz' < raz' = c' = \text{mincex}(M', M_*) \leq c_k = \text{mincex}(M_k, M_*). \tag{3.1}$$

Hence, $M_k$ is correct on $wz'$ and $tz'$. Since $M_k$ cannot distinguish between $t$ and $w$, we obtain

$$M_*(tz') = M_k(tz') = M_k(wz') = M_*(wz'). \tag{3.2}$$

According to (3.1), $M'$ is correct on $wz'$ but wrong on $raz'$. Since $M'$ cannot distinguish between $ra$ and $w$ and $t \equiv_{L_*} ra$, we obtain

$$M_*(tz') = M_*(raz') \neq M'(raz') = M'(wz) = M_*(wz). \tag{3.3}$$

We arrived at a contradiction between (3.2) and (3.3). $\square$

Lemmas 3.5 and 3.7 imply that all hypotheses $M_k$, $k \geq 1$, are partially correct for $L_*$ with respect to $R_k$, which concludes the analysis of `REDIRECT`.

**3.2. Some implementation details.** The following considerations show that `REDIRECT` has time bound $O(|\Sigma|n^3)$.

The central data structure in the implementation of `REDIRECT` is the prefix-tree of the actual set $R_k$. It forms the skeleton of the actual hypothesis DFA $M_k$. The edges are labeled by letters from $\Sigma$. Each node in the tree represents a state $q_r$, where $r \in R_k$ is given implicitly by the path from the root to $q_r$. Each state $q_r$ is correctly labeled as either "accepting" or "nonaccepting." In addition, we use pointers associated with $q_r$ to store the $B$-edges directed from $q_r$ and the $|\Sigma|$ candidate lists $C(r,a)$, $a \in \Sigma$.

Let us first consider Steps 1 to 4 of `REDIRECT`. According to Theorem 3.4, each of these steps is executed less than $|\Sigma|n^2$ times during one run of `REDIRECT`. Furthermore, it is easy to see that each step can be performed in time $O(n)$ using the data structure described above. In particular, note that each word in $R(L_*)$ has length at most $n$ (the number of states of $M_*$). Each shortest counterexample, presented to a hypothesis DFA, has length smaller than $2n$ (the total number of states in $M_*$ and the current hypothesis DFA).[8] It follows that the total time spent for Steps 1 to 4 during one run of `REDIRECT` is bounded by $O(|\Sigma|n^3)$.

Step 5 is executed exactly $n-1$ times by `REDIRECT`, because the final hypothesis consists of $n$ states and each call of procedure `EXTEND` leads to the creation of a new state. The most expensive substeps are 5.4 and 5.5, which can be performed during $|\Sigma|$ appropriate walks through the prefix-tree. In Substep 5.4, `REDIRECT` directs the walk to states $q_t$ with $t < rab$. In Substep 5.5, the walk is directed to states $q_t$ with $ra < tb$. Since each walk takes time $O(n)$, the total time spent for Step 5 during one run of `REDIRECT` is bounded by $O(|\Sigma|n^2)$.

[8]This bound on the length of shortest counterexamples is well known, although we do not know its origin. A proof is contained in [10], for instance.

**4. The SC-learning algorithm for leveled DFAs.** A *leveled* DFA $M$ with $l$ levels is a DFA with the following special properties.

1. State set $Q$ is partitioned into $Q_0, \dots, Q_l$, where $Q_j$ denotes the set of states on level $j$.
2. $Q_0 = \{q_0\}$ contains the initial state.
3. If the DFA is in state $q \in Q_{j-1}$ and reads the $j$th input symbol $x_j = a \in \Sigma$, it passes to a unique state $\delta(q, a) \in Q_j$ for $j = 1, \dots, l$.
4. $Q_l$ is a nonempty subset of $\{q_+, q_-\}$, where $q_+$ is accepting and $q_-$ is nonaccepting.

The corresponding state diagrams are leveled acyclic graphs with nodes on levels $0, \dots, l$. A word $w \in \Sigma^l$ is *accepted* if the computation path $P_M(w)$ ends in $q_+$. All other words are rejected (in particular all words of a length differing from $l$). Thus, $L(M)$ is a subset of $\Sigma^l$.

We denote the class of leveled DFAs with at most $n$ states and input alphabet $\Sigma$ as LEV-DFA$_{\Sigma,n}$. All concepts and results considered so far for DFAs can be transferred to leveled DFAs in the obvious way. This transfer is used in what follows without explicit mentioning.

As outlined in the preceding section, general DFAs are efficiently learnable from less than $|\Sigma|n^2$ smallest counterexamples. Now we will show that $O(|\Sigma|n \log n)$ smallest counterexamples are sufficient for SC-learning leveled DFAs with at most $n$ states and input alphabet $\Sigma$ (even efficiently). In the next section, we show that $\Omega(|\Sigma|n \log n)$ smallest counterexamples are necessary (even with unlimited computational resources). To demonstrate sufficiency, we design an efficient algorithm called `LEVELED-REDIRECT`. It builds on the generic algorithm `REDIRECT` from section 4, but adds some subtle implementation details which prove extremely useful for leveled DFAs. The main modifications are as follows.

*Modification* 1. The hypotheses DFAs will be leveled. Consequently, only counterexamples of length $l$ are received.

*Modification* 2. The second "modification" is obtained automatically when the generic algorithm `REDIRECT` is applied to leveled DFAs. In general, it may happen that `REDIRECT` operates first on candidate set $C(r, a)$ in round $k$, on a different candidate set $C(r', a')$ in the same round (after receiving a counterexample to a TEST-DFA) or in round $k + 1$, and again on $C(r, a)$ in a later round. However, when applied to leveled DFAs, `REDIRECT` exhibits a nice property. Each round will be concerned with exactly one candidate set, say $C(r, a)$, and will find the final setting of it (after presenting a series of TEST-DFAs). $C(r, a)$ will shrink in round $k$ to either the empty set (in which case `EXTEND`$(M, ra)$ is called) or a singleton set $\{s'\}$ (in which case the $B$-edge directed from $q_r$ and labeled by $a$ will point to $q_{s'}$).

*Modification* 3. If the redirection of $B$-edges is cleverly done, then each counterexample will allow us to eliminate not only a single candidate from $C(r, a)$, but at least fifty percent of it.

Modifications 2 and 3 imply that each round performs a sort of binary search for the right candidate in one of the sets $C(r, a)$ and either finds it or adds $ra$ to the prefix-closed set.

The remainder of this section is organized as follows. We start with a detailed explanation of Modifications 2 and 3 in section 4.1. In section 4.2, we present a high-level description of `LEVELED-REDIRECT` (focusing on the main differences to `REDIRECT`). In order to derive a time bound, some implementation details are finally presented in section 4.3.

**4.1. Binary search for the right candidate.** The reason REDIRECT possibly operates on the same candidate set in different rounds is that $c' = r'a'z'$ can be smaller than $c = raz$ even if $ra < r'a'$, because the length of suffix $z'$ can be smaller than the length of suffix $z$. This may cause REDIRECT to stop its current work on $C(r, a)$ intermediately and to switch to $C(r', a')$. For leveled DFAs, however, this cannot happen since all counterexamples are of the same length. These observations lead to Lemma 4.1 which provides a theoretical explanation for Modification 2. The notations are as in Lemma 3.2, but here they refer to leveled DFAs.

LEMMA 4.1. *Let $c = \mathit{mincex}(M, M_*)$, $q_r \xrightarrow{a} q_s$ be the first B-edge on $P_M(c)$, $q_{s'}$ be a state of $M$ located at the same level as $q_s$, $M' = M(r, a, s')$ be the TEST-DFA for B-edge $q_r \xrightarrow{a} q_{s'}$, $c' = \mathit{mincex}(M', M_*)$, and $q_{r'} \xrightarrow{a'} q_t$ be the first B-edge on $P_{M'}(c')$. If $ra$ is not a prefix of $c'$ (i.e., if $ra \neq r'a'$), then $ra <_{\mathrm{LEX}} r'a'$ and $s' \equiv_{L_*} ra$.*

*Proof.* Lemma 3.2 implies that $c' > c$. (Otherwise, $ra$ must be a prefix of $c'$.) Since $c$ and $c'$ both have length $l$, $c < c'$ implies that $ra <_{\mathrm{LEX}} r'a'$. Assume for the sake of contradiction that $s' \not\equiv_{L_*} ra$. Then there exists a word $z \in \Sigma^\star$ such that $M_*(s'z) \neq M_*(raz)$. Since $M'$ cannot distinguish between $s'$ and $ra$, it is wrong on either $s'z$ or $raz$. This contradicts $s'z < raz < c' = \mathrm{mincex}(M', M_*)$. ∎

LEVELED-REDIRECT can use this lemma as follows. Whenever a TEST-DFA for B-edge $q_r \xrightarrow{a} q_{s'}$ leads to a counterexample referring to a candidate set $C(r', a')$ different from $C(r, a)$, B-edge $q_r \xrightarrow{a} q_{s'}$ is correct, and $C(r, a)$ can safely be set to singleton $\{s'\}$.

Assume that $M_k, R_k, c_k, r, a$ (as defined within REDIRECT) are given. Next, we want to explain how $C(r, a)$ can be halved per counterexample. REDIRECT used Lemmas 3.1 and 3.2 to perform a sort of "exhaustive search" for the correct candidate in $C(r, a)$. Each counterexample led to the elimination of one candidate only. It is a straightforward idea to use "binary search" instead. For this purpose, we define in what follows a carefully selected candidate $s(C(r, a))$ which is either correct or leads to the simultaneous removal of all elements from a set $E(C(r, a))$ containing at least half of the former candidates.

The basic idea behind the choice of $s' = s(C(r, a))$ is as follows. If all TEST-DFAs for B-edges directed from $r$ and labeled by $a$ were equivalent, they would certainly receive the same smallest counterexample $c'$. (In a somewhat more general definition below, we will call $C(r, a)$ "homogeneous" in this case.) If they are not all equivalent, there is a smallest word $d$ on which each output label is produced by at least one TEST-DFA. In this case, LEVELED-REDIRECT should "go with the majority," because, if the smallest counterexample happens to be $d$, then the majority of TEST-DFAs fails. This leaves two questions open. First, how do we implement "going with the majority"? Second, what have we gained when the smallest counterexample does not happen to be $d$? Fortunately, there is an easy answer to both questions. First, we can go with the majority by making the commitment to select $s'$ from the subset of candidates leading to the more frequent output label on $d$. Second, the case of counterexamples different from $d$ is handled by a recursive application of this commitment scheme. Details follow.

In what follows, we define operators for selection and elimination. Because the definitions will proceed recursively, they refer to an arbitrary set $C \subseteq C(r, a)$.

1. $D(C) := \{d \in \Sigma^\star \mid \exists s_1, s_2 \in C : M(r, a, s_1)(d) \neq M(r, a, s_2)(d)\}$.
2. $C$ is called *homogeneous* if $D(C) = \emptyset$, and *heterogeneous* otherwise.
3. For heterogeneous $C$, we define $d(C) := \min\{x \mid x \in D(C)\}$, i.e., $d(C)$ is the smallest word on which at least two of the TEST-DFAs disagree. For homogeneous

$C$, we define $d(C) := \infty$ by default (where $\infty$ represents an "infinite word" greater than any word from $\Sigma^\star$).

4. For heterogeneous $C$, word $d(C)$ leads to the following partition of $C$ into $C_0$ and $C_1$:

$$C_i := \{s' \in C |\ M(r,a,s')(d(C)) = i\},\ i = 0,1.$$

Denote by $C_{\mathrm{MAJ}}$ the larger of the two sets (say $C_0$ if there is a tie), and by $C_{\mathrm{MIN}}$ the other set.

5. The *selection operator* is given recursively by

$$s(C) := \begin{cases} \min\{x | x \in C\} & \text{if } C \text{ is homogeneous,} \\ s(C_{\mathrm{MAJ}}) & \text{otherwise.} \end{cases}$$

6. Let $M' = M(r,a,s(C))$ and $c' = \mathrm{mincex}(M', M_*)$. Assume that $c'$ has prefix $ra$. Then the *elimination operator* is given recursively by

$$E(C) := \begin{cases} C & \text{if } c' < d(C), \\ C_{\mathrm{MAJ}} & \text{if } c' = d(C), \\ C_{\mathrm{MIN}} \cup E(C_{\mathrm{MAJ}}) & \text{if } c' > d(C). \end{cases}$$

It follows inductively from the recursive definition that $|E(C)| \geq \frac{1}{2}|C|$. (In particular, $E(C) = C$ if $C$ is homogeneous.) The "binary search" paradigm is therefore established by the following result which (in combination with Lemma 4.1) implies that counterexample $c'$ either witnesses the correctness of the current $B$-edge directed from $r$ and labeled by $a$ or justifies to eliminate all candidates belonging to $E(C)$ simultaneously.

LEMMA 4.2. *For all $C \subseteq C(r,a)$, $M' = M(r,a,s(C))$, and $c' = \mathrm{mincex}(M', M_*)$, the following holds.*

(i) *If $C$ is heterogeneous, then $ra$ is a prefix of $d(C)$.*

(ii) *If $ra$ is prefix of $c'$, then $t \not\equiv_{L_*} ra$ for all $t \in E(C)$.*

*Proof.*

(i) Pick $s_1$ and $s_2$ from $C$ such that $d(C) = \mathrm{mincex}(M_1, M_2)$, where $M_1 = M(r,a,s_1)$ and $M_2 = M(r,a,s_2)$. Obviously, the computation path $P_{M_1}(d(C))$ contains $B$-edge $q_r \xrightarrow{a} q_{s_1}$. It remains to show that this $B$-edge is the first $B$-edge on the path. Since $M_1$ is partially correct for $L(M_2)$, we may apply Lemma 3.1, identifying $M$ by $M_1$ and $M_*$ by $M_2$. We get that the first $B$-edge on $P_{M_1}(d(C))$ is wrong, i.e., this $B$-edge is not contained in $M_2$. The only $B$-edge not contained in $M_2$ is $q_r \xrightarrow{a} q_{s_1}$, yielding the claim.

(ii) The proof proceeds by case analysis, but the following argument will apply in all cases. A collection of DFAs shares the same smallest counterexample $c$ if the smallest word on which two of them disagree is greater than $c$. This argument will be used without further explicit mentioning.

If $c' < d(C)$, then $c' \notin D(C)$. Thus $c'$ is the smallest counterexample for each $M(r,a,t)$ with $t \in C$. Since $ra$ is prefix of $c'$, Lemma 3.1 yields $t \not\equiv_{L_*} ra$ for all $t \in C$. If $c' = d(C)$, then $M'$ votes on $c'$ as the majority of all TEST-DFAs. Thus $c'$ is the smallest counterexample not only for $M'$, but for all $M(r,a,t)$ with $t \in C_{\mathrm{MAJ}}$. Again, Lemma 3.1 applies. This time, it yields $t \not\equiv_{L_*} ra$ for all $t \in C_{\mathrm{MAJ}}$.

If $c' > d(C)$, then $M'$ is correct on $d(C)$. Thus $d(C)$ is the smallest counterexample for each $M(r,a,t)$ with $t \in C_{\mathrm{MIN}}$. Since $d(C)$ has prefix $ra$, Lemma 3.1 applies again. Now, it yields $t \not\equiv_{L_*} ra$ for all $t \in C_{\mathrm{MIN}}$. Furthermore, $t \not\equiv_{L_*} ra$ for all $t \in E(C_{\mathrm{MAJ}})$ follows inductively. □

**4.2. High-level description of** `LEVELED-REDIRECT`**.** The initialization of `LEVELED-REDIRECT` is performed analogously to the starting phase of `REDIRECT`. This time we find a first partially correct leveled DFA $M_1$ using the leveled DFA $M_1^0$ and $M_1^1$ shown in Figure 4.1.

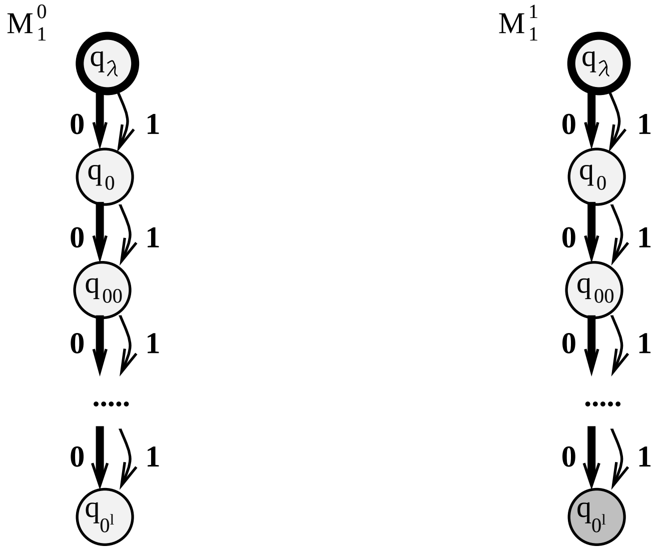


FIG. 4.1. *Initial leveled DFAs $M_1^0$ and $M_1^1$ for $\Sigma = \{0, 1\}$.*

An arbitrary round $k$ of algorithm `LEVELED-REDIRECT` is similarly structured as the corresponding round in `REDIRECT`. For this reason, we restrict ourselves to stress the main differences.

1. The former Step 4.1 is replaced by $s' \leftarrow s(C(r, a))$, i.e., the next candidate $B$-edge is determined by the selection operator.
2. The former Step 4.4 is replaced by

```
if $ra$ is prefix of $c'_k$ then eliminate all elements of $E(C(r,a))$ from $C(r,a)$
                              else SAMEROUND←FALSE.
```

3. The updates in Step 5 are adapted to leveled DFAs. In the **else** part (i.e., if $ra$ is not prefix of $c'$), we add the command $C(r, a) \leftarrow \{s'\}$. This reflects that $B$-edge $q_r \xrightarrow{a} q_{s'}$ is known to be correct in this case.

It becomes obvious from these modifications, that we treat $C(r, a)$ completely in one round, and halve it per counterexample.

We note that the selection and elimination operators allow, even in the case of arbitrary DFAs, to halve one of the candidate sets per smallest counterexample. The problem is that for arbitrary DFAs Lemma 4.1 does not hold. Thus, iterated halving of a set $C(r, a)$ and the insertion of up to $n$ elements in $C(r, a)$ are possibly interleaved. This may cause $\Omega(n)$ halvings. (For this reason, `REDIRECT` uses the simpler exhaustive search paradigm.) Because of Lemma 4.1, `REDIRECT` is not faced with this problem. When candidate set $C(r, a)$ is shrunk to the empty or a singleton set during one round, no later round will cause insertions into $C(r, a)$. From these considerations, we obtain the following theorem.

THEOREM 4.3. `LEVELED-REDIRECT` *polynomially SC-learns LEV-DFA$_{\Sigma,n}$ from at most $(1 + o(1))|\Sigma|n \log n$ smallest counterexamples.*

The *width* $w(M)$ of a leveled DFA $M$ is defined as

$$w(M) = \max_{0 \le i \le l} \{|Q_i|\},$$

i.e., $w(M)$ is the maximal number of nodes that appear on the same level in the state diagram of $M$. Let LEV-DFA$_{\Sigma,n,w}$ denote the class of leveled DFAs with at most $n$ states, input alphabet $\Sigma$, and width at most $w$. The following result is a straightforward generalization of Theorem 4.3.

COROLLARY 4.4. `LEVELED-REDIRECT` *polynomially SC-learns LEV-DFA*$_{\Sigma,n,w}$ *from at most* $(1 + o(1))|\Sigma|n \log w$ *smallest counterexamples.*

*Proof.* Remember that all updates performed by `LEVELED-REDIRECT` are adapted to leveled DFAs. In particular, all candidate sets $C(r, a)$ contain only words of length $|r| + 1$. Thus, the sizes of these sets are bounded by $w$. The number of halvings needed to shrink a candidate set from full size to size 1 or 0 is therefore logarithmically bounded in $w$. ◻

**4.3. Some implementation details.** We briefly describe how `LEVELED-REDIRECT` can be implemented such as to run in time proportional to $|\Sigma|^2 lnw^2$. The reader not interested in this low-level information can pass to section 5 without loss of continuity.

We use the same data structure as for `REDIRECT` to represent the actual hypothesis $M_k$. As in the analysis of `REDIRECT`, the total run time spent for Step 5 is bounded by $O(|\Sigma|n^2)$. (Since $n \le lw$, this does not contradict the claimed time bound.)

Let us now consider Steps 1 to 4 of `LEVELED-REDIRECT`. Obviously, it is sufficient to show that the total run time, spent for the recursive evaluations of the selection and elimination operators, is bounded by $O(|\Sigma|^2 lnw^2)$. Since there are $|\Sigma|n$ rounds, each round being concerned with a specific candidate set $C(r, a)$, the problem boils down to showing that the recursive evaluations during one round have time bound $O(|\Sigma|lw^2)$.

To this end, we consider the application of selection operator $s$ on $C \subseteq C(r, a)$. (The analysis of call $E(C)$ is similar.) $C$ is stored as a list of pointers (pointing to states $q_s$ for $s \in C$). Let $K$ denote the cardinality of $C$ and $N(K)$ the number of steps needed to compute $s(C)$. It is easy to show that, for each $s \in C$, the following holds:

$$d(C) = \min_{t \in C} \{\text{mincex}(M_k(r, a, s), M_k(r, a, t))\}.$$

Hence, we have to find the minimum of $K$ smallest counterexamples between TEST-DFAs in order to compute $d(C)$. If we adapt a procedure, presented in [10] for the corresponding problem on OBDDs,[9] it is possible to compute each counterexample in time proportional to $|\Sigma|l$. Hence after at most $O(|\Sigma|lK)$ steps, we either know that $C$ is homogeneous (which stops the recursion), or we can use $d(C)$ to compute $C_{\text{MAJ}}$ of cardinality at most $K - 1$ and enter the next level of recursion. Since $C_{\text{MAJ}}$ is easy to compute from $d(C)$, we get the recursion

$$N(K) \le \alpha_1 |\Sigma| lK + N(K - 1),\ N(1) = \alpha_2$$

for constants $\alpha_1, \alpha_2$. Obviously, $N(K) = O(|\Sigma|lK^2)$. We note without proof that the recursive evaluation of $E(C)$ has the same time bound (which can be derived by a similar analysis).

[9]As mentioned in the introduction, OBDDs stands for ordered binary decision diagrams, which is a data structure coinciding with what we call leveled DFAs up to minor differences.

During one round, candidate set $C(r,a)$ is iteratively halved. Before each halving, the operators $s$ and $E$ are applied to $C(r,a)$. In the beginning of a round, the cardinality of $C(r,a)$ is at most $w$. The recursive calls are therefore initiated on sets of sizes $w, \frac{w}{2}, \frac{w}{4}, \ldots$. It follows that the total time spent in one round is bounded by

$$O(|\Sigma|lw^2) \cdot \left(1^2 + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{4}\right)^2 + \cdots\right),$$

which is proportional to $|\Sigma|lw^2$ (as desired).

## 5. A general lower bound for SC-learning.

In this section, we show that any learning algorithm for LEV-DFA$_{\Sigma,n}$ (LEV-DFA$_{\Sigma,n,w}$, respectively) requires at least $\Omega(|\Sigma|n\log n)$ ($\Omega(|\Sigma|n\log w)$, respectively) smallest counterexamples. (Thus, `LEVELED-REDIRECT` is optimal in this respect.) On the way to this result, we prove a general lower bound in terms of a combinatorial dimension associated with the target class. Then we apply this result to the classes of arbitrary and leveled DFAs.

Let $\mathcal{F}$ be a concept class over a linearly ordered domain $X$. We say that a finite set $S \subseteq X$ is SC-shattered by $\mathcal{F}$ if $\mathcal{F}$ contains $2^{|S|}$ concepts that are pairwise different on $S$, but coincide on $X^{<S} := \{x \in X \setminus S \mid x < \max S\}$. Note that these concepts can realize each binary output pattern on $S$. The *SC-dimension*[10] of $\mathcal{F}$ is defined as follows:

$$\mathrm{SCdim}(\mathcal{F}) := \sup\{\, d \mid \exists S \subseteq X : \ |S| = d \text{ and } S \text{ is SC-shattered by } \mathcal{F}\}.$$

We show the following general lower bound.

LEMMA 5.1. *Each SC-learning algorithm for $\mathcal{F}$ requires at least $SC\mathrm{dim}(\mathcal{F})$ smallest counterexamples.*

*Proof.* Let $S$ be an SC-shattered set of size $d$, $\mathcal{F}_0 \subseteq \mathcal{F}$ be the set containing the $2^d$ SC-shattering concepts for $S$, and $v(x) \in \{0,1\}$ be their common value on an instance $x \in X^{<S}$. The lower bound is obvious from the following "adversary-argument."

Let $s_1 < s_2 < \cdots < s_d$ denote the ordered sequence of elements from $S$. The strategy of the adversary is to present $s_1, s_2, \ldots, s_d$ (in this order) as smallest counterexamples. When $s_i$ is presented, the value of the target concept on $s_i$ is revealed, but still all values on $s_{i+1}, \ldots, s_d$ are conceivable. The mistakes on $s_1, \ldots, s_d$ are therefore unavoidable. This strategy can be slightly disturbed by a hypothesis $h$ with an "additional" mistake on an $s_j$, $j \le i$, already presented, or on an element $s < s_{i+1}, s \notin S$. Now the adversary presents the smallest instance $s_{min}$ on which $h$ makes an additional mistake. In case of $s_{min} = s_j$, the SC-learner does not gain information. In case of $s_{min} \notin S$, the SC-learner only comes to know value $v(s_{min})$ being common for all concepts in $\mathcal{F}_0$. Thus, the additional mistakes do not restrict the variability of the adversary on $s_{i+1}, \ldots, s_d$. ∎

LEMMA 5.2. *The SC-dimensions of the classes* DFA$_{\Sigma,n}$ *and* LEV-DFA$_{\Sigma,n}$ *are bounded as follows:*

$$\left(\frac{1}{4} - o(1)\right)|\Sigma|n\log n \le SC\mathrm{dim}(\text{LEV-DFA}_{\Sigma,n}) \le SC\mathrm{dim}(\text{DFA}_{\Sigma,n}) \le (1+o(1))|\Sigma|n\log n.$$

[10]The reader familiar with the theory of Vapnik and Chervonenkis (see [22]) will notice the similarity between the SC-dimension and the VC-dimension. In general, $\mathrm{SCdim}(\mathcal{F}) \le \mathrm{VCdim}(\mathcal{F})$ because the constraint of coincidence on $X^{<S}$ does not occur in the definition of the VC-dimension. Intuitively, this additional constraint will enable an adversary to pick all nonredundant smallest counterexamples from $S$.

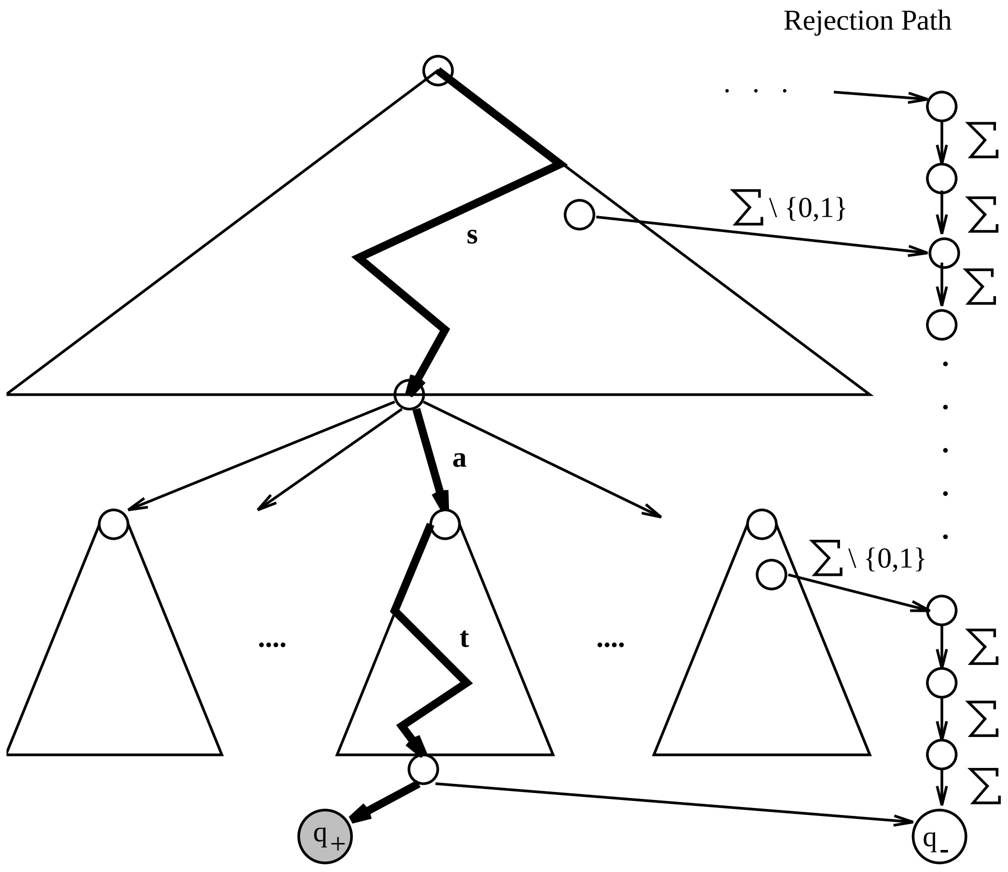


FIG. 5.1. *State diagram D for the proof of Lemma 5.2. The path for an input word $w = sat \in S$ through D is displayed accentuated.*

*Proof.* The rightmost inequality follows from the fact that $\mathrm{SCdim}(\mathcal{F}) \leq \log|\mathcal{F}|$ and that $|\mathrm{DFA}_{\Sigma,n}| \leq n2^n n^{n|\Sigma|}$. Since leveled DFAs form a subclass of arbitrary DFAs, we know

$$SC\mathrm{dim}(\text{LEV-DFA}_{\Sigma,n}) \leq SC\mathrm{dim}(\mathrm{DFA}_{\Sigma,n}).$$

The leftmost inequality follows from showing that the set

$$S = \{0,1\}^{\log n}\Sigma\{0,1\}^{\log m},$$

where $m = \log n - \log\log n$, is SC-shattered by leveled DFAs with at most $4n$ states and input alphabet $\Sigma$. A word $w$ from $S$ has the form $w = sat$, where $s$ is a binary word of length $\log n$, $a$ is an arbitrary letter, and $t$ is a binary word of length $\log m$. It is convenient to represent a binary output pattern on $S$ as a collection of $|\Sigma|$ binary matrices with $n$ rows and $m$ columns, respectively. Given $w = sat \in S$, we obtain the corresponding output bit from entry $(s, t)$ of the matrix $\mathcal{M}_a$ associated with letter $a$. We now show that an arbitrary collection $(\mathcal{M}_a)_{a\in\Sigma}$ of such matrices can be realized by a leveled DFA $M$ with less than $4n$ states. Figure 5.1 shows the structure of the transition diagram $D$.

We describe the resulting computation on a correctly formed input $w = sat \in S$: Diagram $D$ starts with a complete binary tree of depth $\log n$ with $2n-1$ nodes and $n$ leaves. The leaves are in one-to-one correspondence to the possible strings $s$ in the decomposition $w = sat$. In other words, when reaching a leaf, $s$ is stored in the finite control. Now $M$ reads letter $a$. At this moment, it knows perfectly the relevant matrix $\mathcal{M}_a$, and the relevant row $s$. The remainder of the computation only depends on the binary pattern in row $s$ of matrix $\mathcal{M}_a$. Instead of storing $(s, a)$ in the finite control, it is more economical to store the binary pattern $p$ of the row: if for $(s_1, a_1)$ and $(s_2, a_2)$, the rows $s_1$ and $s_2$ in $M_{a_1}$ and $M_{a_2}$, respectively, form the same binary pattern $p$, the corresponding edges in $D$ point to the same node. Consequently, the next level in $D$ contains $2^m = n/\log n$ nodes, which is precisely the number of binary patterns of length $m = \log n - \log\log n$. Each of these $n/\log n$ nodes is the root of a complete binary tree of depth $\log m$. When $M$ reaches a leaf in one of these trees, it has stored suffix $t$ (and still the relevant binary pattern $p \in \{0,1\}^m$) in its finite control, and can select and output the correct bit.

If the input word $w$ does not have the correct form, i.e., $w = sat \notin S$, either $s$ or $t$ contains a digit different from 0 and 1. In this case, diagram $D$ feeds the computation path into a "rejection path," which assigns default value 0 to each $w \notin S$. The number of nodes in diagram $D$ is easily seen to be less than $4n$. This shows that $S$ is SC-shattered by the class of leveled DFAs with less than $4n$ states. □

Note that the $2^{|S|}$ DFAs, used in this proof to SC-shatter $S$, coincide not only on $X^{<S}$, but even on $X \setminus S$. This observation will be used in section 7, which is devoted to SD-learning.

Similar bounds for the VC-dimension of class $\mathrm{DFA}_{\Sigma,n}$ were recently shown in [17]. Our lower bound is stronger for two reasons. First, it applies to leveled DFAs. Second, it applies to the SC-dimension which is smaller, in general, than the VC-dimension. Besides that, our proof is much simpler. Note that the DFAs considered in the proof have $4n$ states and width $n+1$. So, the stated lower bound doesn't contradict Corollary 4.4.

We are also able to bound the SC-dimension for leveled DFAs in terms of their width.

LEMMA 5.3. *The SC-dimension of the class* $\mathrm{LEV\text{-}DFA}_{\Sigma,n,w}$ *is bounded as follows:*

$$\left(\frac{1}{4} - o(1)\right)|\Sigma| n \log w \le SC\dim(\mathrm{LEV\text{-}DFA}_{\Sigma,n,w}) \le (1 + o(1))|\Sigma| n \log w.$$

*Proof.* The upper bound follows from Lemma 5.1 and Corollary 4.4.

The lower bound can be shown by a variant of the construction that was used in the proof of Lemma 5.2. There, we constructed for each $n$ a subclass $\mathcal{D}$ of leveled DFAs with less than $4n$ states that SC-shatters a set $S$ of size $(|\Sigma|n(\log n - \log\log n) = (1 - o(1))|\Sigma| n \log n$. Note that all DFAs in $\mathcal{D}$ have width $n+1$. This construction does not readily imply Lemma 5.3 because parameters $w$ and $n$ cannot be chosen independently from each other. However, Lemma 5.3 obviously follows from the existence of a family of subclasses $(\mathcal{D}_k)_{k \ge 1}$ that consists of leveled DFAs and has the following properties.

1. Each DFA in $\mathcal{D}_k$ has at most $N = 4kn$ states and width $w = n + 2$.
2. $\mathcal{D}_k$ SC-shatters a set $S_k$ of size $k|S| = (k|\Sigma|n(\log n - \log\log n) = (1/4 - o(1))|\Sigma| N \log w$.

Figure 5.2 shows how a DFA $D$ from $\mathcal{D}_k$ is obtained from $k$ DFAs $D_1, \ldots, D_k$ from $\mathcal{D}$. The computation path $P_{D_i}(0^l)$, leads to the root node of $D_{i+1}$. Each

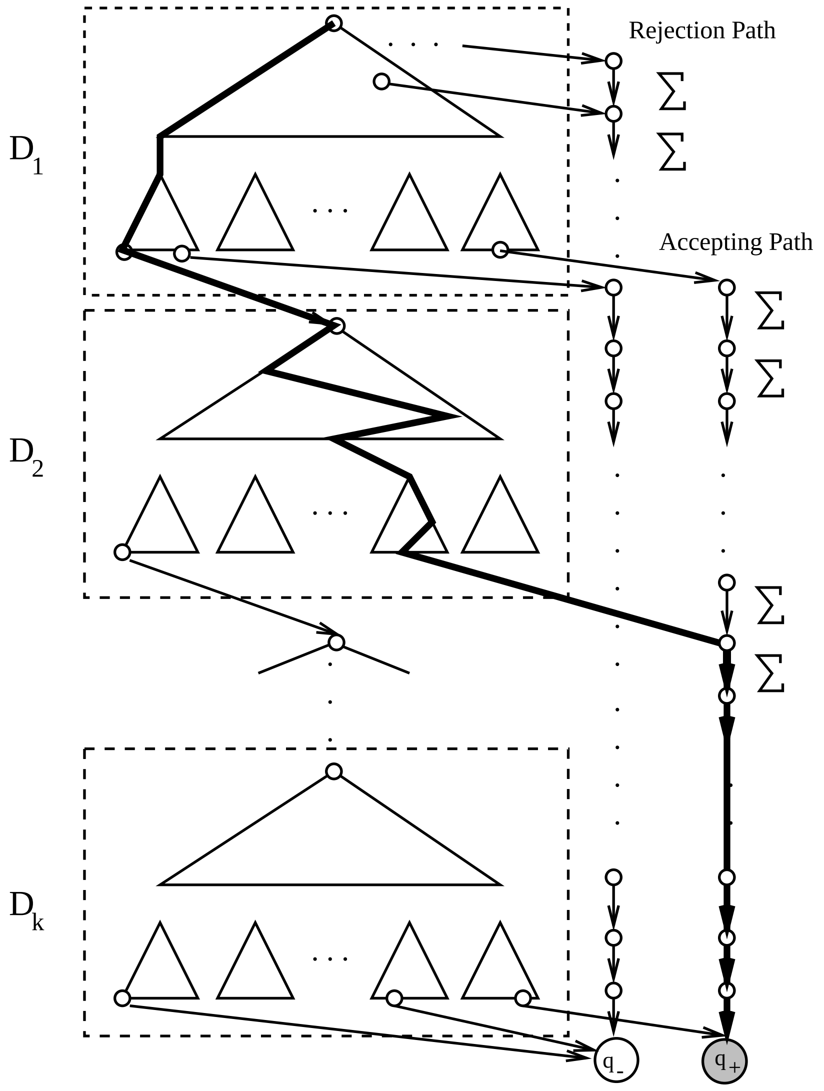


FIG. 5.2. *State diagram of a DFA $D$ from $\mathcal{D}_k$ obtained from $k$ DFAs $D_1, \ldots, D_k$ from $\mathcal{D}$. The computation path for an input word $0^l sat 0^{(k-2)l} \in S_k$ is displayed accentuated.*

other computation path $P_{D_i}(v)$ is fed either into the rejection path (if it ends at a nonaccepting state in $D_i$) or into the accepting path (if it ends at an accepting state in $D_i$).

It is not hard to see that the following holds. Given that $\mathcal{D}$ SC-shatters $S$, $\mathcal{D}_k$ SC-shatters

$$S_k = \{0^{il}\ S\ 0^{(k-i-1)l} \mid i = 0, \ldots, k-1\},$$

a set of the desired size $k|S|$. The width of a DFA $D$ from $\mathcal{D}_k$ is clearly $n+2$ (since a DFA from $\mathcal{D}$ has width $n+1$, and we added the accepting path). The number of nodes in $D$ is obviously bounded by $4kn$ plus the number of nodes within the accepting path (which would already suffice to accomplish the proof). A closer inspection of the sizes of the DFAs $D_i$ from $\mathcal{D}$ reveals that $D$ from $\mathcal{D}_k$ has at most $4kn$ states. This completes the proof of Lemma 5.3. $\square$

**6. A halving strategy for proper SC-learning.** Let $\mathcal{F}$ be a finite concept class over domain $X$. It is well known that $\mathcal{F}$ can be learned from $\lfloor \log |\mathcal{F}| \rfloor$ arbitrary counterexamples if the learner may issue arbitrary hypotheses of the form $h: X \to \{0,1\}$ (not necessarily belonging to $\mathcal{F}$). The underlying learning strategy is the so-called *majority vote* or *halving strategy.* The learner keeps track of the current *version space* $\mathcal{V}$ consisting of all concepts from $\mathcal{F}$ that are consistent with all counterexamples received so far. For each $x \in X$ and $i \in \{0,1\}$, let

$$\mathcal{V}_i(x) = \{f \in \mathcal{V} |\ f(x) = i\}.$$

An element $x$ is called *proper split element* for $\mathcal{V}$ if $\mathcal{V}_0(x)$ and $\mathcal{V}_1(x)$ are nonempty. A function $h: X \to \{0,1\}$ is called *majority vote with respect to* $\mathcal{V}$ if $|\mathcal{V}_0(x)| > |\mathcal{V}_1(x)|$ implies $h(x) = 0$ and $|\mathcal{V}_1(x)| > |\mathcal{V}_0(x)|$ implies $h(x) = 1$. If the learner issues an EQ with a majority vote as hypothesis, then the resulting counterexample reduces the size of the current version space by fifty percent or more. Clearly, after receiving $\lfloor \log |\mathcal{F}| \rfloor$ counterexamples, the initial version space $\mathcal{F}$ is shrunk to a singleton and the target concept is exactly identified (up to equivalence). The halving strategy for learning from counterexamples has two drawbacks. First, it cannot be implemented efficiently in general. Second, it uses hypotheses which do not belong to $\mathcal{F}$ in general.

We speak of *proper learning* when the learner uses only hypotheses from the concept class $\mathcal{F}$. Let us assume that domain $X$ is augmented with a linear ordering, and each nonempty subset of $X$ has a smallest element. We present the so-called *SC-halving strategy* which leads to proper SC-learning of $\mathcal{F}$ from $\lfloor \log |\mathcal{F}| \rfloor$ smallest counterexamples. The basic idea is quite simple. Let $\mathcal{V}$ denote the current version space. If $|\mathcal{V}| = 1$, then the target concept is identified. Otherwise, let $x = x(\mathcal{V}) \in X$ denote the smallest proper split element for $\mathcal{V}$, $\mathcal{V}_{\mathrm{MAJ}}$ the bigger one of the two sets $\mathcal{V}_0(x), \mathcal{V}_1(x)$ and $\mathcal{V}_{\mathrm{MIN}}$ the smaller one (breaking ties arbitrarily). Now, the SC-learner commits itself to pick its hypothesis from $\mathcal{V}_{\mathrm{MAJ}}$, i.e., to make a majority vote on $x$. Then it recursively applies its strategy to $\mathcal{V}_{\mathrm{MAJ}}$ until the resulting subspace of $\mathcal{V}$ is a singleton $\{f\}$. Finally, $f$ is issued as the hypothesis. Loosely speaking, the learner anticipates the iterated splittings of $\mathcal{V}$ into subspaces that could result from smallest counterexamples and goes always with the majority. The resulting hypothesis is not a majority vote with respect to $\mathcal{V}$ in the aforementioned sense. Instead, it is a majority vote on the smallest proper split element for $\mathcal{V}$ with respect to $\mathcal{V}$, and a majority vote on the smallest proper split element for $\mathcal{V}_{\mathrm{MAJ}}$ with respect to $\mathcal{V}_{\mathrm{MAJ}}$, and so on.

Let $f_*$ denote the target concept. A more formal description of the SC-halving strategy uses a selection operator $s(\mathcal{V})$ which represents the hypothesis chosen from $\mathcal{V}$, and an elimination operator $E(\mathcal{V})$ which represents the concepts from $\mathcal{V}$ which can be eliminated after receiving the smallest counterexample $c = \mathrm{mincex}(s(\mathcal{V}), f_*)$:

$$s(\mathcal{V}) := \begin{cases} f & \text{if } \mathcal{V} = \{f\}, \\ s(\mathcal{V}_{\mathrm{MAJ}}) & \text{if } |\mathcal{V}| \geq 2, \end{cases}$$

$$E(\mathcal{V}) := \begin{cases} \mathcal{V}_{\mathrm{MAJ}} & \text{if } c = x(\mathcal{V}), \\ \mathcal{V}_{\mathrm{MIN}} \cup E(\mathcal{V}_{\mathrm{MAJ}}) & \text{if } c > x(\mathcal{V}). \end{cases}$$

(Don't be confused by the fact that the operators deal with hypotheses this time, as opposed to states, as they did before.) Note that the case $c < x(\mathcal{V})$ cannot occur because the target concept $f_*$ belongs to $\mathcal{V}$. It is obvious that each smallest counterexample reduces the size of the current version space by at least fifty percent. Thus, we arrive at the following result.

THEOREM 6.1. *Let $\mathcal{F}$ be a finite concept class. The SC-halving strategy SC-learns $\mathcal{F}$ from at most $\lfloor \log |\mathcal{F}| \rfloor$ smallest counterexamples.*

COROLLARY 6.2. *The SC-halving strategy SC-learns DFA$_{\Sigma,n}$ from $(1 + o(1)) \cdot |\Sigma| n \cdot \log n$ smallest counterexamples.*

Note that the run time of the SC-halving strategy is polynomial in the size of $\mathcal{F}$. Since the number of essentially different DFAs with $n$ states is exponential in $n$, it is not a polynomial learning strategy for DFAs. It shows, however, that our lower bound $\Omega(|\Sigma| n \log n)$ on the number of smallest counterexamples needed to SC-learn DFA$_{\Sigma,n}$ is tight. We leave the question of whether this bound can be matched by a polynomial SC-learning algorithm as an object of future research.

## 7. SC-learning and SD-learning.

The model of *SD-learning* was introduced by Goldman and Sloan in [14]. This section presents an efficient conversion of SC-learners into SD-learners. For the particular concept classes DFA$_{\Sigma,n}$, LEV-DFA$_{\Sigma,n}$, and LEV-DFA$_{\Sigma,n,w}$, the conversion will lead to SD-learners whose number of mistakes matches a lower bound (modulo a small constant).

We briefly recall the SD-learning model. Let $\mathcal{F}$ denote a finite concept class over a finite domain $X$. SD-learning of a target concept $f_* \in \mathcal{F}$ proceeds in $|X|$ trials. In each trial, the SD-learning algorithm $A$ (sometimes simply called SD-learner) picks a "new" instance $x$ from $X$ (being different from all instances picked in previous trials), computes a prediction label $l \in \{0, 1\}$, supplies $(x, l)$ to a teacher, and receives the true label $f_*(x)$ as an answer. We say that $A$ made a *mistake* if $l \neq f_*(x)$. The total number of mistakes during the $|X|$ trials is denoted by $m_A(f_*)$. We say that *$A$ SD-learns $\mathcal{F}$ making at most $m_A := \max_{f_* \in \mathcal{F}} m_A(f_*)$ mistakes.* These definitions can be adapted in the obvious manner to the parameterized case, where $(\mathcal{F}_n, X_n)$ is a countable sequence of finite concept classes $\mathcal{F}_n$ over finite domains $X_n$, respectively. $m_A = m_A(n)$ is then written as function in $n$. $A$ is called *polynomial* if $m_A(n)$ and the maximum computation time per trial are both polynomially bounded in $n$.

The sequence $x_1, x_2, \ldots, x_{|X|} \in X$ of instances picked by an SD-learning algorithm in trials $1, 2, \ldots, |X|$ when learning $f_* \in \mathcal{F}$ is called the *query sequence* of $A$ for target concept $f_*$. Note that the query sequence is computed by $A$ in an on-line fashion (leading to different query sequences for different target concepts, in general). An SD-learner is called *oblivious* when it uses the same query sequence for all target concepts from $\mathcal{F}$. If $X$ is augmented with a linear ordering, the oblivious SD-learner always using the (correctly) ordered query sequence is called *strictly monotone.* It is not hard to construct artificial concept classes which demonstrate that the hierarchy, consisting of strictly monotone, oblivious, and general SD-learners, is strict.

Goldman and Sloan have shown in [14] that a partial equivalence query (a query type including ordinary equivalence or membership queries as special cases) can be simulated by an SD-learner at the expense of only one mistake. It follows that each query-learning algorithm using at most $m$ partial equivalence queries can be converted into an SD-learning algorithm making at most $m$ mistakes. We will present a similar conversion of SC-learners into (even strictly monotone) SD-learners. It is technically convenient to break this conversion into two steps by introducing strictly monotone SC-learning algorithms.

Let $A$ be an algorithm which SC-learns a concept class $\mathcal{F}$. The sequence of counterexamples received by $A$ when learning $f_* \in \mathcal{F}$ is called the *mincex-sequence* of

$A$ for target concept $f_*$. $A$ is called *strictly monotone* if, for all $f_* \in \mathcal{F}$, the mincex-sequence of $A$ for target concept $f_*$ is strictly increasing (with respect to the given linear ordering of $X$).

LEMMA 7.1. *Each algorithm $A$ which SC-learns $\mathcal{F}$ from at most $m$ smallest counterexamples can be converted into a strictly monotone algorithm $A'$ which SC-learns $\mathcal{F}$ from at most $m$ smallest counterexamples.*

*Proof.* Let $h_i$ denote the $i$th hypothesis of $A$ when learning target concept $f_*$ and $c_i = \mathrm{mincex}(h_i, f_*)$. An EQ with hypothesis $h_i$ is called *redundant* if there exists an index $j < i$ with $c_j \geq c_i$. Otherwise, it is called *relevant.* The subsequence of $c_i$ consisting of all counterexamples received after relevant queries is obviously strictly increasing. $A'$ proceeds as $A$ except that redundant queries are not delivered to the teacher. We have to describe how to check whether the EQ with hypothesis $h_{i+1}$ is redundant and what to do if it is. The crucial observation is that $h_i$ and $f_*$ coincide on all $x \in X$ with $x < c_i$ and differ on $c_i$. Given $h_i$ and $c_i$, the EQ with hypothesis $h_{i+1}$ is relevant iff $\mathrm{mincex}(h_{i+1}, h_i) = c_i$ (which happens iff $\mathrm{mincex}(h_{i+1}, f_*) > c_i$). Thus instead of supplying hypothesis $h_{i+1}$ to the teacher, $A'$ computes $c' := \mathrm{mincex}(h_{i+1}, h_i)$ and tests redundancy by comparing $c'$ with $c_i$. If $c' = c_i$ (the relevant case), then $A'$ may issue an EQ with hypothesis $h_{i+1}$. Otherwise, $c = \mathrm{mincex}(h_{i+1}, f_*)$ can be directly computed: $c$ is the smaller word among $c'$ and $c_i$. $\square$

The proof of Lemma 7.1 leads to the following observation. $A'$ achieves a strictly increasing mincex-sequence at the expense of at most $m$ mincex-computations and at most $m$ comparisons of two instances (with respect to the given linear ordering of $X$). Thus, if $A$ is polynomial and the computation of the smallest counterexample of two given hypotheses and the comparison of two given instances can be performed in polynomial time, then $A'$ is polynomial as well.

LEMMA 7.2. *Assume that concept class $\mathcal{F}$ and domain $X$ are finite. Then each strictly monotone algorithm $A$ which SC-learns $\mathcal{F}$ from $m$ smallest counterexamples can be converted into a strictly monotone algorithm $A'$ which SD-learns $\mathcal{F}$ making at most $m$ mistakes and vice versa.*

*Proof.* We first present the conversion of $A$ into $A'$. Let $|X| = N$ and let $x_1 < x_2 < \cdots < x_N$ be the ordered sequence of all instances from $X$, which coincides with the query sequence of $A'$ for all target concepts. Let $c_1, \ldots, c_k$ with $k \leq m$ be the mincex-sequence of $A$ for target concept $f_*$. Since $A$ is strictly monotone, $c_1 < c_2 < \cdots < c_k$ is a subsequence of $x_1 < x_2 < \cdots < x_N$. $A'$ simulates $A$ according to the following policy. Let $h_i$ denote the current hypothesis of $A$ and $x_j$ the next instance of the query-sequence. (Initially, $i = j = 1$.) The prediction label for $x_j$ is set to $h_i(x_j)$. If the label is correct, then $j$ is incremented. If the label is incorrect, then $x_j$ is delivered to $A$ as smallest counterexample to hypothesis $h_i$, $j$ and $i$ are both incremented, and the simulation can continue. The crucial observation is that the first mistake happens when $x_j = c_i$. Thus, $A'$ makes its mistakes exactly on the mincex-subsequence of the query sequence. This leads to $k \leq m$ mistakes.

Let's now assume that $A'$ is given and must be converted into $A$. $A$ uses a simulation of $A'$. When $x_i$ is the next instance in the query sequence, $A$ computes a hypothesis $h_i$ as follows. For all $x_j$ with $j < i$, $h_i(x_j)$ is set to the correct label $f_*(x_j)$, which is known at this stage. In order to set $h_i(x_j)$ appropriately for $j \geq i$, $A'$ is simulated to the end, except that it receives its own prediction labels instead of the correct ones. Loosely speaking, $A$ puts $A'$ into a virtual reality, where no mistakes are reported. For all $j \geq i$, $h_i(x_j)$ is now set to the prediction label that was proposed by $A'$. Then $A$ issues an EQ with hypothesis $h$ and receives $c_i = \mathrm{mincex}(h_i, f_*)$.

Obviously, $c_i = x_j$, where $j \geq i$ points to the earliest trial with a wrong prediction label. Then $A'$ is reset to trial $j$, receives the true label $f_*(x_j)$, and the simulation proceeds with trial $j+1$. Clearly, the total number of counterexamples delivered to $A$ equals the total number of mistakes made by $A'$. □

According to Lemma 7.2, strictly monotone SC-learners and strictly monotone SD-learners are (information-theoretically) equivalent. In addition, if the SC-learning algorithm $A$ can present its next hypothesis in polynomial time and if a given hypothesis can be evaluated on a given instance in polynomial time, then the corresponding strictly monotone SD-learning algorithm performs each prediction in polynomial time.

We want to apply the conversion of SC-learners into strictly monotone SD-learners to the class of (leveled or arbitrary) DFAs with at most $n$ states. Lemma 7.2 cannot be applied directly because domain $X = \Sigma^*$ is infinite. However, since a DFA with at most $n$ states is uniquely determined (up to equivalence) by its acceptance behavior on words of length at most $2n-1$, we may simply pass to subdomain $X' = \{w \in \Sigma^* : |w| \leq 2n-1\}$.[11] We denote the strictly monotone SD-learner resulting from a given SC-learner $A$ by $A_{SD}$ in what follows and immediately obtain the following corollary.

COROLLARY 7.3.

(i) $\texttt{REDIRECT}_{SD}$ *is a strictly monotone algorithm which SD-learns the class* $DFA_{\Sigma,n}$ *making at most* $1 + |\Sigma|n^2 - |\Sigma|n$ *mistakes. Each prediction is done in polynomial time.*

(ii) $\texttt{HALVING}_{SD}$ *is a strictly monotone algorithm which SD-learns* $DFA_{\Sigma,n}$ *making at most* $(1+o(1))|\Sigma|n\log n$ *mistakes. Each prediction is done in exponential time.*

(iii) $\texttt{LEVELED-REDIRECT}_{SD}$ *is a strictly monotone algorithm which SD-learns the class* $LEV\text{-}DFA_{\Sigma,n}$ *(*$LEV\text{-}DFA_{\Sigma,n,w}$*, respectively) making at most* $(1+o(1))|\Sigma|n\log n$ *(*$(1+o(1))|\Sigma|n\log w$*, respectively) mistakes. Each prediction is done in polynomial time.*

We want to show that any SD-learning algorithm for $\mathrm{DFA}_{\Sigma,n}$ or $\text{LEV-DFA}_{\Sigma,n}$ makes at least $\Omega(|\Sigma|n\log n)$ mistakes. Towards this end, we introduce the SD-dimension, which is closely related to the SC-dimension.

Assume that $\mathcal{F}$ is a finite concept class over a finite domain $X$. We say that $S \subseteq X$ is SD-shattered by $\mathcal{F}$ if $\mathcal{F}$ contains $2^{|S|}$ concepts that are pairwise different on $S$, but coincide on $X \setminus S$. The *SD-dimension* is defined as follows:

$$SD\mathrm{dim}(\mathcal{F}) := \max\{\, d \mid \exists S \subseteq X : \ |S| = d \text{ and } S \text{ is SD-shattered by } \mathcal{F}\}.$$

The definitions of the SC- and the SD-dimension differ only slightly. In the latter case, the $2^{|S|}$ concepts used to shatter $S$ have to coincide on $X \setminus S$. In the former case, they have to coincide on $X^{<S} \subseteq X \setminus S$. It follows that $SD\mathrm{dim}(\mathcal{F}) \leq SC\mathrm{dim}(\mathcal{F})$ for each concept class $\mathcal{F}$. We will see, however, in Lemma 7.5 that, for the specific classes $\mathrm{DFA}_{\Sigma,n}$ and $\text{LEV-DFA}_{\Sigma,n}$, the SC-dimension and the SD-dimension have the same order of magnitude.

The following lower bound follows from a straightforward adversary argument.

LEMMA 7.4. *Each SD-learning algorithm for* $\mathcal{F}$ *makes at least* $SD\mathrm{dim}(\mathcal{F})$ *mistakes.*

*Proof.* Let $S$ denote a maximum SD-shattered set for $\mathcal{F}$. By definition, there exist $2^{|S|}$ concepts in $\mathcal{F}$ which coincide outside $S$ and realize all $2^{|S|}$ label combinations on

[11]A similar argument applies whenever $\mathcal{F}$ is finite. We may then pass to subdomain $X' = \{x \in X \mid \exists f, g \in \mathcal{F} : x = \mathrm{mincex}(f,g)\}$, because the labels of the target concept $f_*$ on $X'$ uniquely specify $f_*$ and the instances outside $X'$ are never returned as smallest counterexamples anyway.

$S$. The lemma easily follows from the following observations. First, the SD-learner gains no information within trials concerned with instances outside $S$. Second, there exists a target concept such that each of its predictions on $S$ is wrong. $\square$

LEMMA 7.5. *The SD-dimensions of the classes* $\mathrm{DFA}_{\Sigma,n}$ *and* $\text{LEV-DFA}_{\Sigma,n}$ *are bounded as follows:*

$$\left(\frac{1}{4} - o(1)\right) |\Sigma| n \log n \leq SD\mathrm{dim}(\text{LEV-DFA}_{\Sigma,n}) \leq SD\mathrm{dim}(\mathrm{DFA}_{\Sigma,n})$$
$$\leq (1 + o(1))|\Sigma| n \log n.$$

*Proof.* The upper bound follows from $SD\mathrm{dim}(\mathrm{DFA}_{\Sigma,n}) \leq SC\mathrm{dim}(\mathrm{DFA}_{\Sigma,n})$ and Lemma 5.2. The lower bound follows from Lemma 5.2 and the subsequent remark concerning the proof of this lemma. (The DFAs used for SC-shattering a suitable set $S$ even satisfy the stronger condition for SD-shattering.) $\square$

In analogy to Lemma 5.3, we obtain the following lemma.

LEMMA 7.6. *The SD-dimension of the class* $\text{LEV-DFA}_{\Sigma,n,w}$ *is bounded as follows:*

$$\left(\frac{1}{4} - o(1)\right) |\Sigma| n \log w \ \leq \ SD\mathrm{dim}(\text{LEV-DFA}_{\Sigma,n,w}) \ \leq \ (1 + o(1))|\Sigma| n \log w.$$

We omit the proof, which is similar to the proof of Lemma 5.3.

As for the leveled or arbitrary DFAs, the situation for SD-learning is analogous to the situation for SC-learning.

1. `LEVELED-REDIRECT`$_{SD}$ is a polynomial SD-learning algorithm for the classes $\text{LEV-DFA}_{\Sigma,n}$ and $\text{LEV-DFA}_{\Sigma,n,w}$ making the smallest possible number of mistakes (modulo a small constant).
2. The SD-learning algorithm for $\mathrm{DFA}_{\Sigma,n}$ resulting from the halving strategy makes the smallest possible number of mistakes (modulo a small constant).
3. `REDIRECT`$_{SD}$ is a polynomial SD-learning algorithm for $\mathrm{DFA}_{\Sigma,n}$ whose number of mistakes differs from the smallest possible number by factor $\Theta(n/\log n)$.

We leave the problem to find a polynomial SD-learning algorithm for $\mathrm{DFA}_{\Sigma,n}$ making the smallest possible number of mistakes (say, modulo a small constant) as an object of future research. This problem might be easier to tackle than the corresponding problem for SC-learning because SD-learning is, in general, strictly more powerful than SC-learning.

## 8. A corollary concerning MAs.

We finally mention a consequence of the results from the previous section concerning MAs. Let $\Sigma$ denote an alphabet and $K$ a field. A *formal series* (with respect to $\Sigma, K$) is a function of the form $f : \Sigma^* \to K$. Matrix $(F_{x,y})_{x,y\in\Sigma^*}$ with $F_{x,y} = f(x \cdot y)$ is called the *Hankel matrix* of $f$. Series $f$ is called *recognizable* if its Hankel matrix $F$ has finite rank. We omit the formal definition of MAs and just note that a formal series is recognizable iff it can be computed by an MA. Moreover, the rank of the Hankel matrix of $f$ coincides with the size (number of states) of the smallest MA computing $f$. The reader interested in background information on MAs is referred to [12, 13, 8].

It was shown in [7] and [20] that MAs are efficiently learnable from multiplicity[12] and equivalence queries. In [5], these learning algorithms are considerably simplified and the amazing power of MAs is demonstrated by showing that various Boolean concept classes can be considered as classes of recognizable formal series of a "small"

[12]A *multiplicity query* is instantiated with an element $x \in \Sigma^*$ and is answered with $f(x)$.

rank. Since Boolean functions are defined on Boolean vectors of fixed length, the corresponding MAs are *leveled*. (The definition of leveled MAs is analogous to the definition of leveled DFAs.)

We denote the class of MAs with at most $n$ states and alphabet $\Sigma$ by $\text{MA}_{\Sigma,n}$. The corresponding subclass of leveled MAs is denoted by $\text{LEV-MA}_{\Sigma,n}$. Bergadano and Varricchio have shown in [6] that $\text{MA}_{\Sigma,n}$ is SC-learnable from $O(|\Sigma|n^5)$ smallest counterexamples, where $O(|\Sigma|n^2)$ smallest counterexamples are already sufficient for the important subclass $\text{LEV-MA}_{\Sigma,n}$. In [18], Forster presented an improved algorithm, called MA-REDIRECT[13] in what follows, which SC-learns $\text{MA}_{\Sigma,n}$ from $2+|\Sigma|n(n+1)$ smallest counterexamples. He showed furthermore that

$$\text{SCdim}(\text{MA}_{\Sigma,n}) \geq \text{SCdim}(\text{LEV-MA}_{\Sigma,n}) \geq |\Sigma|n^2/64. \tag{8.1}$$

Thus $O(|\Sigma|n^2)$ smallest counterexamples are sufficient to SC-learn $\text{MA}_{\Sigma,n}$ efficiently, and $\Omega(|\Sigma|n^2)$ smallest counterexamples are necessary (even for subclass $\text{LEV-MA}_{\Sigma,n}$ and even with unlimited amount of computation). An inspection of the proof of the above inequalities (8.1) reveals that the same inequalities are valid with the SD-dimension substituted for the SC-dimension. Applying the results of the previous section, we immediately obtain the following corollary.

COROLLARY 8.1.

(i) `MA-REDIRECT`$_{SD}$ *is a strictly monotone algorithm which SD-learns* $MA_{\Sigma,n}$ *making at most* $2+|\Sigma|n(n+1)$ *mistakes. Each prediction is done in polynomial time.*

(ii) *Each SD-learning algorithm for* $MA_{\Sigma,n}$ *(or even* $LEV\text{-}MA_{\Sigma,n}$*) makes at least* $|\Sigma|n^2/64$ *mistakes.*

**Acknowledgments.** The authors would like to thank two anonymous referees for many valuable suggestions.

## REFERENCES

[1] S. B. AKERS, *Binary decision diagrams*, IEEE Trans. Comput., 27 (1978), pp. 509–516.
[2] D. ANGLUIN, *Learning regular sets from queries and counterexamples*, Inform. and Comput., 75 (1987), pp. 87–106.
[3] D. ANGLUIN, *Queries and concept learning*, Machine Learning, 2 (1988), pp. 319–342.
[4] D. ANGLUIN, *Negative results for equivalence queries*, Machine Learning, 5 (1990), pp. 121–150.
[5] A. BEIMEL, F. BERGADANO, N. H. BSHOUTY, E. KUSHILEVITZ, AND S. VARRICCHIO, *On the applications of multiplicity automata in learning*, in Proceedings of the 37th IEEE Annual Symposium on the Foundations of Computer Science, Burlington, VT, 1996, pp. 349–358.
[6] F. BERGADANO AND S. VARRICCHIO, *Learning behaviours of automata from shortest counterexamples*, in Proceedings of the 2nd European Conference on Computational Learning Theory, Springer-Verlag, Berlin, 1995, pp. 380–391.
[7] F. BERGADANO AND S. VARRICCHIO, *Learning behaviors of automata from multiplicity and equivalence queries*, SIAM J. Comput., 25 (1996), pp. 1268–1280.
[8] J. BERSTEL AND C. REUTENAUER, *Rational Series and Their Languages*, Monogr. Theoret. Comput. Sci. EATCS Ser. 12, Springer-Verlag, Berlin, 1988.
[9] A. BIRKENDORF, A. BÖKER, AND H. U. SIMON, *Learning deterministic finite automata from smallest counterexamples*, in Proceedings of the 9th International ACM–SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, pp. 599–609.
[10] A. BIRKENDORF AND H. U. SIMON, *Using computational learning strategies as a tool for combinatorial optimization*, Ann. Math. Artificial Intelligence, 22 (1998), pp. 237–257.
[11] R. E. BRYANT, *Symbolic boolean manipulation with ordered binary-decision diagrams*, ACM Computing Surveys, 24 (1992), pp. 293–318.
[12] J. W. CARLYLE AND A. PAZ, *Realization by stochastic finite automaton*, J. Comput. System Sci., 5 (1971), pp. 26–40.

[13] A clever adaptation of algorithm `REDIRECT` as it was published in [9].

[13] M. Fliess, *Matrices de hankel*, J. Math. Pures Appl. (9), 53 (1974), pp. 197–222. Erratum in vol. 54.

[14] S. A. Goldman and R. H. Sloan, *The power of self-directed learning*, Machine Learning, 14 (1994), pp. 271–294.

[15] J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relation to Automata*, Addison Wesley, Reading, MA, 1969.

[16] O. H. Ibarra and T. Jiang, *Learning regular languages from counterexamples*, J. Comput. System Sci., 43 (1991), pp. 299–316.

[17] Y. Ishigami and S. Tani, *VC-dimensions of finite automata and commutative finite automata with k letters and n states*, Discrete Appl. Math., 74 (1997), pp. 123–134.

[18] J. Forster, *Learning multiplicity automata from smallest counterexamples*, in Proceedings of the 4th European Conference on Computational Learning Theory, Springer-Verlag, Berlin, 1999, pp. 79–90.

[19] C. Y. Lee, *Representation of switching circuits by binary-decision programs*, Bell Systems Technical Journal, 38 (1959), pp. 985–999.

[20] H. Ohnishi, H. Seki, and T. Kasami, *A polynomial time learning algorithm for recognizable series*, IEICE Transactions on Informations and Systems, E77-D(10) (1994), pp. 1077–1085.

[21] R. L. Rivest and R. E. Schapire, *Inference of finite automata using homing sequences*, Inform. and Comput., 103 (1993), pp. 299–347.

[22] V. N. Vapnik and A. Y. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory Probab. Appl., 16 (1971), pp. 264–280.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 4, pp. 492–504

# BOUNDS FOR CODES IDENTIFYING VERTICES IN THE HEXAGONAL GRID[*]

GÉRARD D. COHEN[†], IIRO HONKALA[‡], ANTOINE LOBSTEIN[†], AND GILLES ZÉMOR[†]

**Abstract.** In an undirected graph $G = (V, E)$, a subset $C \subseteq V$ is called an identifying code if the sets $B_1(v) \cap C$ consisting of all elements of $C$ within distance one from the vertex $v$ are nonempty and different. We take $G$ to be the infinite hexagonal grid and show that the density of any identifying code is at least 16/39 and that there is an identifying code of density 3/7.

**Key words.** graph, hexagonal grid, code, identifying code, density

**AMS subject classifications.** 05C70 (68R10, 94B65)

**PII.** S0895480199360990

## 1. Introduction.

Let $G = (V, E)$ be an undirected graph (which may be finite or infinite). For $x, y \in V$, we denote by $d(x, y)$ the graphic distance between them, i.e., the number of edges in any shortest path from $y$ to $x$. If $d(x, y) \le 1$, we say that $x$ *covers* $y$ (and vice versa).

For $x \in V$, we denote

$$B_i(x) = \{y \in V : d(y, x) \le i\}$$

and

$$S_i(x) = \{y \in V : d(y, x) = i\}.$$

A *code* $C$ is a nonempty subset of $V$. Its elements are called *codewords*. The code $C$ is called an *identifying code* if the sets $B_1(v) \cap C$, $v \in V$, are all nonempty and different.

Motivation for this last definition comes from Karpovsky, Chakrabarty, and Levitin [8] and Karpovsky, Chakrabarty, Levitin, and Avresky [9] and is concerned with fault diagnosis in arrays of processors. A multiprocessor system can be modeled as an undirected graph $G = (V, E)$, where $V$ is the set of processors and $E$ is the set of links in the system. Fault diagnosis consists of testing the system and locating faulty processors. For this purpose a set of processors will be selected, and they will be assigned the task of testing their neighbors for malfunctions. Whenever a selected processor detects a fault of any kind among its neighbors or malfunctions itself, an error message is issued that specifies only its origin. We see that the set of selected processors should make up an identifying code of $G$.

We are therefore interested in identifying codes which have as few vertices as possible. In [8], among other graphs, the grids that represent the three tilings of the plane by regular polygons are presented and the question is raised: what is the minimum density of an identifying code for the square, triangular, and hexagonal grid?

[*]Received by the editors September 8, 1999; accepted for publication (in revised form) May 3, 2000; published electronically October 6, 2000.
http://www.siam.org/journals/sidma/13-4/36099.html

[†]ENST and CNRS, 46 rue Barrault, 75013 Paris, France (cohen@inf.enst.fr, lobstein@inf.enst.fr, zemor@infres.enst.fr).

[‡]Department of Mathematics, University of Turku, 20014 Turku, Finland (honkala@utu.fi). The research of this author was supported by the Academy of Finland.

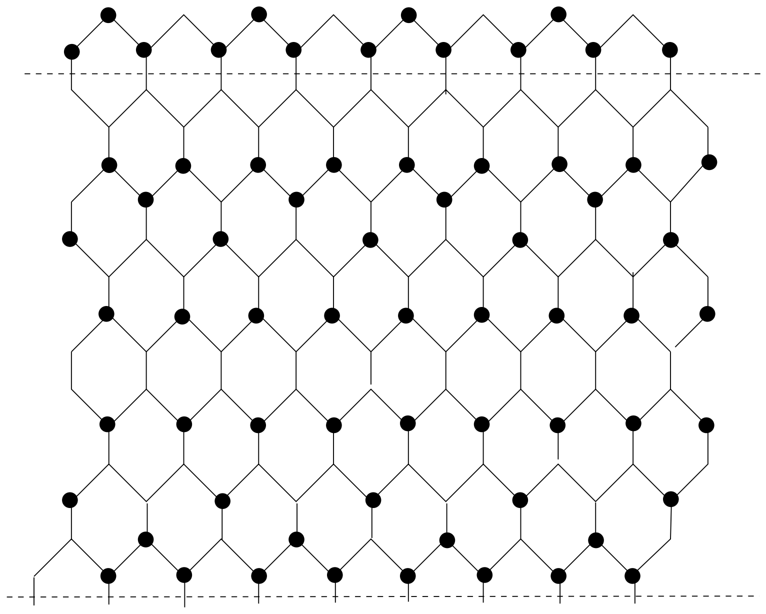


FIG. 1. *An identifying code of density* 3/7.

The triangular grid is the only one of the three for which the answer is known [8]. The minimum density of the square grid has been taken up in [4] and [3], where upper and lower bounds are derived: the gap between bounds is now smaller than 0.002.

In this paper we consider the remaining case when $G$ is the infinite hexagonal grid. In [8, Theorem 12] it is shown that the minimum density $D$ satisfies $2/5 \leq D \leq 1/2$. We shall improve this to $16/39 \leq D \leq 3/7$.

The papers [1] and [2] give further results in the Hamming spaces, and [5] and [6] give results for other graphs.

For a closely related problem in which $B_1(v) \cap C$ are required to be different only for noncodewords $v \in V \setminus C$, see Haynes, Hedetniemi, and Slater [7].

**2. Constructions.** There are at least two different periodic identifying codes in the infinite hexagonal grid, with density 3/7. They are represented on Figures 1 and 2, which should speak for themselves. (The formal definition of density is given only in section 3, but in Figures 1 and 2 the identifying codes are built using finite tiles, and the density can simply be obtained by calculating the proportion of codewords among the vertices in the tiles.)

THEOREM 2.1. *There is an identifying code of density* 3/7 *in the infinite hexagonal grid.*

**3. Preparation for the lower bound.** It will be useful for us to have a numbering of the vertices of the grid: the one we shall use is represented on Figure 3.

It will also be convenient to introduce a sequence $G_2, G_3, \ldots$, of finite subgraphs of $G$. The graphs $G_2$, $G_3$, and $G_4$ are shown in Figures 4, 5, and 6.

For even $m$, the points 0 and 9 are the two "middle" points of $G_m$; for odd $m$, the "middle" points are 1 and 10. The graph $G_m$ has roughly the shape of a rhombus, has $m$ hexagons on the bottom, and all in all consists of $m^2$ hexagons and

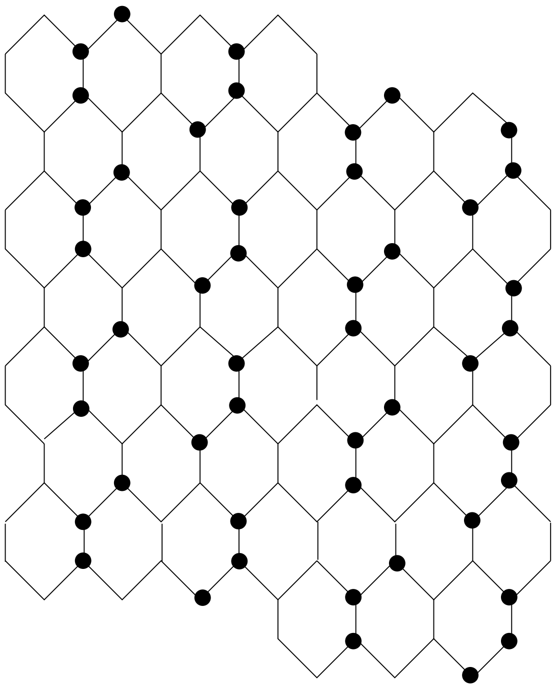


FIG. 2. *Another identifying code of density* 3/7.

$2m^2 + 4m$ vertices. The number of vertices in the perimeter of $G_m$ is $8m - 2$. Now $G_2 \subseteq G_3 \subseteq G_4 \ldots$, and the union of the graphs $G_m$, $m \geq 2$, is $G$.

We furthermore define a sequence $H_m$ of graphs by gluing together certain vertices in $G_m$. Namely, we glue together the points in the perimeter so that the points on the bottom are considered to be the same as the ones on the top, and the ones on the left are glued to the ones on the right. So, for instance, to obtain $H_2$ from $G_2$, we glue 3 to 7, 2 to 8, 1 to 29, and 12 to 30; and moreover, we glue 6 to 30, 5 to 31, 4 to 10, and 3 to 11. The idea is that we tile (in the natural way—there are others) the whole space using $G_m$ as a tile (each tile with the same fixed numbering), and whenever two points with different numbers meet, we consider them as one and the same point. In $G_2$, for instance, the "floor" consisting of 3, 2, 1, 12 coincides with the "ceiling" 7, 8, 29, 30; and, e.g., 11 is glued to 3 and 7. In this way we obtain a finite 3-regular graph with $2m^2$ vertices.

The density $D$ of an identifying code $C$ in the hexagonal grid $G$ is defined as

$$D = \limsup_{m\to\infty} \frac{|C \cap V(G_m)|}{|V(G_m)|},$$

where $V(G_m)$ is the vertex set of $G_m$. For finite graphs $(V, E)$ we shall also speak of the density of a code $C$, which is simply $D = |C|/|V|$.

In section 4 we show that for all large $m$, the density of an identifying code in $H_m$ is at least 16/39. Then the following argument shows that any identifying code in $G$ also has density at least 16/39.

Assume that we have shown that the density of any identifying code in $H_m$ has density at least $\alpha$, and that this is true for all $H_m$ such that $m \geq k$, where $k$ is a constant. This implies that the density of any identifying code $C$ in $G$ has density

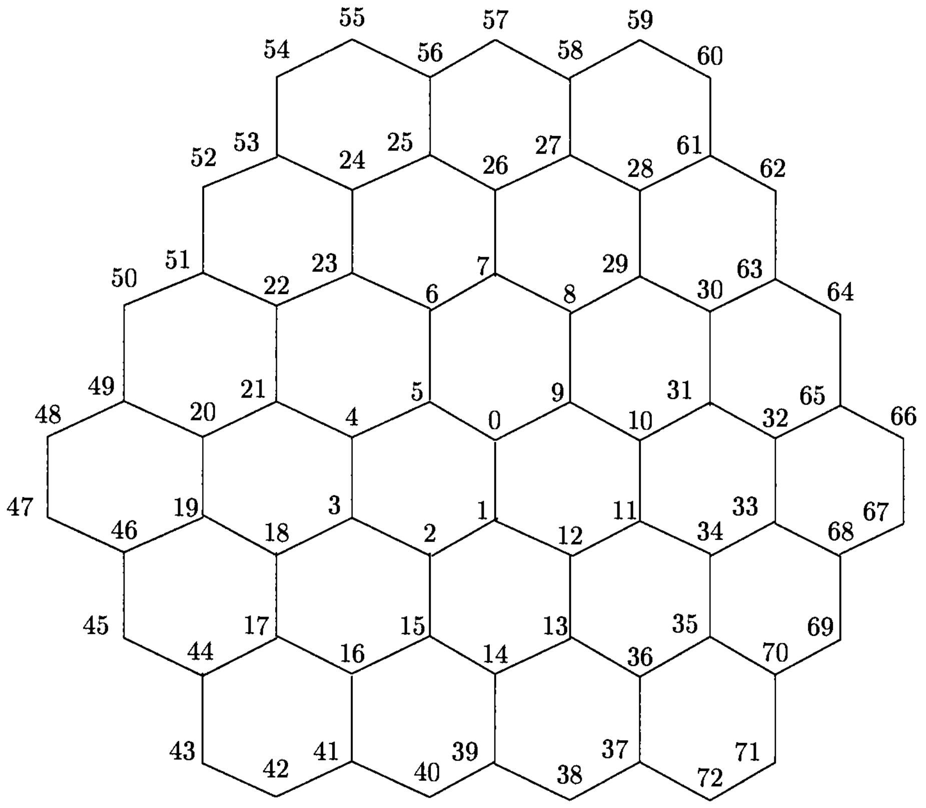


FIG. 3. *Numbering of the points of the hexagonal grid.*

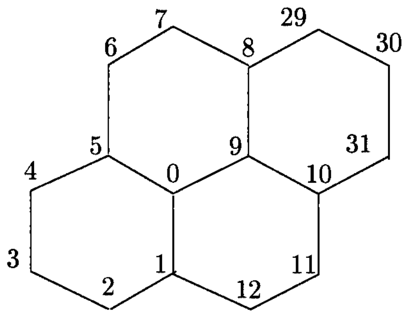


FIG. 4. *The graph $G_2$.*

at least $\alpha$. Indeed, let $T_m$ consist of all the vertices in the perimeter of $G_m$ together with the codewords in $C \cap V(G_m)$. It is easy to verify that these points—viewed now as points in $H_m$—form an identifying code in $H_m$. Therefore,

$$|C \cap V(G_m)| + 8m - 2 \geq |T_m| \geq \alpha \cdot 2m^2,$$

i.e.,

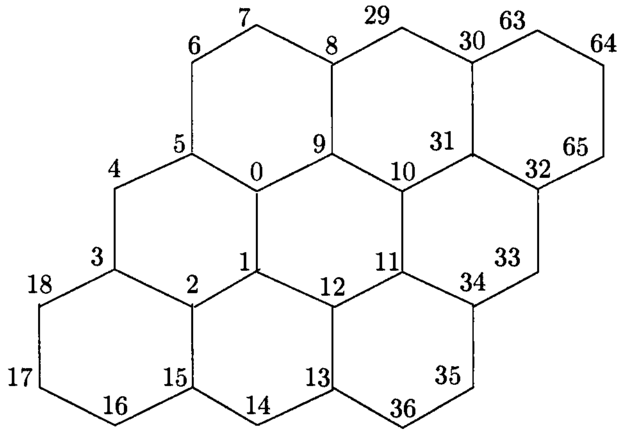


FIG. 5. *The graph $G_3$.*

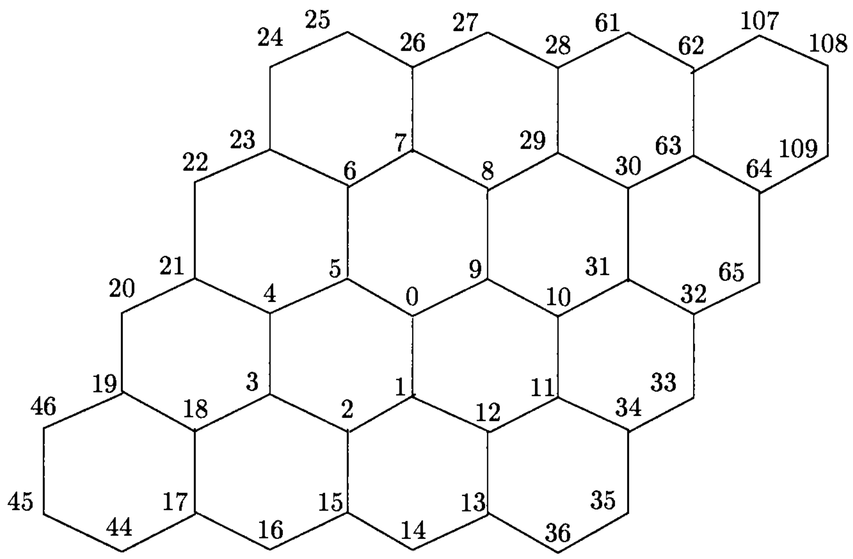


FIG. 6. *The graph $G_4$.*

$$\frac{|C \cap V(G_m)|}{|V(G_m)|} \geq \alpha \frac{2m^2}{2m^2+4m} - \frac{8m-2}{2m^2+4m},$$

from which the claim follows.

**4. Lower bound.** Assume from now on that our graph $G = (V, E)$ is $H_m$, where $m$ is suitably large, say at least 20.

Because every vertex has degree three, we know that

$$4|C| \geq |C| + 2(|V| - |C|).$$

We call the difference of the left-hand side and the right-hand side the *excess* (on $G$) and denote it by $e(C)$, i.e., $e(C) = 5|C| - 2|V|$.

Definition 4.1. *A point that is covered by more than two codewords is called an excess point.*

*A codeword $c \in C$ for which there is no vertex $x \in V$ such that $B_1(x) \cap C = \{c\}$ is called a deficient codeword.*

*A point is special if it is an excess point or a deficient codeword (or both).*

*The set of deficient codewords is denoted by $\mathcal{D}$, the set of excess points by $\mathcal{E}$, and the set of special points by $\mathcal{A}$.*

*If $c \in C$ and $B_1(c) \cap C = \{c\}$, we call $c$ an isolated codeword.*

We divide the codewords according to their distance to the nearest special point.

Definition 4.2. *Denote*

$$C_i = \{c \in C : d(c, \mathcal{A}) = i\}$$

*for $i = 0, 1, 2, \dots$.*

The following simple observations will be used repeatedly.

- *If $c \in C$, $c' \in C$, and $d(c, c') = 1$, then at least one of the words $c$ and $c'$ is an excess point.* Indeed, the sets $B_1(c) \cap C$ and $B_1(c') \cap C$ are different and contain both $c$ and $c'$, and therefore at least one of the sets must have more than two elements.
- *If $c \in C_i$, $i \geq 2$, then $c$ is an isolated codeword.* If not, then there is a codeword $c' \in C$ with $d(c, c') = 1$, and by the previous observation $c$ or $c'$ is an excess point, which is of course impossible when $c \in C_i$, $i \geq 2$.

Lemma 4.3. *For any codeword $c \in C$ we have $d(c, \mathcal{E}) \leq 4$. Moreover, if $d(c, \mathcal{E}) = 4$, then there are at least two excess points at distance four from $c$, or there exists a deficient codeword at distance four from $c$.*

In particular, the lemma implies that if $c \in C$ has distance four to the nearest excess point, then there are (1) two excess points at distance four, or (2) an excess point and a deficient codeword (which may coincide) at distance four.

*Proof.* Assume that $c \in C$ is the point 0 and that $d(c, \mathcal{E}) \geq 4$. Our first observation is that every codeword $b \in C$ within distance two from $c$ is an isolated codeword; if $b' \in B_1(b) \cap C \setminus \{b\}$, then $b$ or $b'$ is an excess point and still within distance three from $c$.

In particular, none of 1, 5, 9 is in $C$. The point 9 cannot be covered only by 0, and therefore either 8 or 10 is in $C$. By symmetry, let us assume that 8 is in $C$ and 10 is not. By our first observation, neither 7 nor 29 is in $C$. We now consider two cases.

*Case* 1: 6 *is not in* $C$. Then 6 must be covered by 23, and 23 cannot be an isolated codeword, so either 22 or 24 is in $C$, but not both; otherwise 23 is an excess point at distance three from 0. Anyway, 22 or 24 is an excess point and has distance four to 0.

We know that 5 must be covered by a codeword other than 0, and since 6 is not in $C$, 4 must be a codeword, and again by our first observation, neither 3 nor 21 is in $C$. If 22 is in $C$, we know that 51 is as well; but then 22 is a deficient codeword, and we are done. Assume therefore that 22 is not a codeword and 24 is, together with 25 or 53. We know that 7 must be covered by a codeword other than 8, and hence 26 is in $C$. If now 53 is in $C$, then 24 is a deficient codeword and we are done. We may therefore assume that 53 is not in $C$, but 25 is.

We know that 10 is not in $C$. If 31 is not in $C$, then 10 must be covered by 11, and, moreover, 12 or 34 is also in $C$. This implies that 34 is the required second excess point. Assume hence that 31 is in $C$. If 30 is a codeword, then 30 is the second excess point. Assume that 30 is not in $C$. Then 29 must be covered by 28, which makes 26 a deficient codeword, completing Case 1.

*Case* 2: 6 *is in* $C$. We can now proceed similarly. First, neither 4 nor 23 is in $C$. We know that 2 or 12 is in $C$ but not both. By symmetry, assume that 2 is a codeword and 12 is not. By our first observation, neither 3 nor 15 is in $C$. But then 4 must be covered by 21, and, moreover, either 20 or 22 is in $C$, but not both, which implies that 20 or 22 is an excess point and has distance four to 0.

If 22 is in $C$, so is 51, and we are done since 22 is a deficient codeword. Assume therefore that 22 is not in $C$, but 20 is. Then 19 or 49 is also in $C$. Because 3 must be covered by a codeword other than 2, we know that 18 is in $C$. Hence if 49 is a codeword, we are done, because 20 is a deficient codeword. Assume, therefore, that 19 is a codeword.

If 11 is not in $C$, then 12 must be covered by 13, and, moreover, 14 or 36 is a codeword. Then 14 or 36 is an excess point and we are done. We can therefore assume that 11 is in $C$. Moreover, we can assume that 11 is an isolated codeword: otherwise 34 is an excess point and we are done. Now 12 must be covered also by 13. If 13 is not an isolated codeword, we are done, so assume that 14 is not in $C$. Since 15 is not a codeword, 16 must be. But then 18 is a deficient codeword, concluding Case 2. $\square$

Denote

$$\mathcal{A}_2 = \mathcal{E} \cap \mathcal{D},$$
$$\mathcal{A}_1 = \mathcal{A} \setminus \mathcal{A}_2.$$

We further denote for $i = 0, 1, 2, \dots,$

$$C_i'' = \{c \in C_i : |S_i(c) \cap \mathcal{A}| \geq 2 \text{ or } S_i(c) \cap \mathcal{A}_2 \neq \emptyset\},$$
$$C_i' = C_i \setminus C_i'',$$

and

$$C' = \bigcup_{i=0}^{4} C_i',$$
$$C'' = \bigcup_{i=0}^{4} C_i''.$$

By Lemma 4.3 we know that $C_i = C_i' = C_i'' = \emptyset$ for all $i \geq 5$, and so $C = C' \cup C''$.

For a given $x \in \mathcal{A}$ we denote for $i = 0, 1, 2, 3, 4,$

$$f_i = |S_i(x) \cap C_i|,$$
$$g_i = |S_i(x) \cap C_i'|,$$
$$h_i = |S_i(x) \cap C_i''|,$$
$$g = \sum_{i=0}^{4} g_i,$$

and

$$h = \sum_{i=0}^{4} h_i.$$

By the previous lemma, $C_4 = C_4''$ and therefore $f_4 = h_4$. Trivially, $f_i = g_i + h_i$.

LEMMA 4.4. *Assume that $x \in \mathcal{A}$. Then $g + \frac{1}{2}h \leq 17/2$.*

*Proof.* In what follows we very often replace some of the terms $g_i + \frac{1}{2}h_i$ by $f_i$ and show that even such a stronger version of the required inequality is true. Moreover, we show that, apart from a single special case, we always have $g + \frac{1}{2}h \leq 8$.

*Case* 1: *$x = 0$ is both an excess point and a deficient codeword.* Then we may assume that 1 and 9 are in $C$. Clearly 2, 12, 10, and 8 are not in $C_2$. If 5 is also in $C$, then $f_2 = 0$ and $f_0 + f_1 + f_2 \leq 4$. Assume that 5 is not in $C$. Now at most one of the points 4 and 6 can be in $C_2$, and again $f_0 + f_1 + f_2 \leq 4$.

There are nine points at distance three from $x$ and 12 at distance four. Moreover, if 3 is in $C_3$, then (since all the codewords in $C_3$ are isolated) 18 cannot be in $C_4$. In general, in any of the nine pairs $(3, 18)$, $(21, 20)$, $(23, 24)$, $(7, 26)$, $(29, 28)$, $(31, 32)$, $(11, 34)$, $(13, 36)$, $(15, 16)$, if the first component belongs to $C_3$, the second cannot belong to $C_4$. Therefore $f_3 + f_4 \leq 12$ and $f_0 + f_1 + f_2 + f_3 + f_4 \leq 16$. By the assumption that $x$ is both an excess point and a deficient codeword we have $f_i = h_i$ and $g_i = 0$, and we get $g + \frac{1}{2}h \leq 8$.

*Case* 2: *$x = 0$ is an excess point and belongs to $C$ but is not a deficient codeword.* We may assume that 1 and 9 are in $C$, and none of 4, 5, and 6 are. Clearly $f_0 + f_1 + f_2 \leq 3$.

Assume first that 9 is a special point. Then 7, 29, 31, and 11 are not in $C_3$, and 26, 28, 30, 32, and 34 are not in $C_4$. The set $C_3$ can contain 23, 21 and at most one of 3, 15, and 13: if 13 together with at least one of 3 and 15 is in $C_3$, then 1 is a deficient codeword, contradicting the fact that 13 is in $C_3$; if 3 and 15 are in $C_3$, then 2 would be an excess point, contradicting the fact that 3 is in $C_3$. As in Case 1, we have the pairs $(23, 24)$, $(21, 20)$, $(3, 18)$, $(15, 16)$, $(13, 36)$, in which the first component can be in $C_3$ or the second in $C_4$ but not both. For all $j = 0, 1, 2, 3$ we therefore have $f_3 + \frac{1}{2}f_4 \leq j + \frac{1}{2}(7 - j) \leq 5$, and we are done. We can therefore assume that 9 is not special. By symmetry we can assume that 1 is not special either.

In particular, we can therefore assume that none of the points 8, 10, 12, and 2 is in $C$.

Now, 18 cannot be in $C_4$; otherwise it would be an isolated point, and 18 and 3 would both only be covered by 18. Similarly, neither 26 nor 34 is in $C_4$.

Moreover, if 11 is in $C$, then (recalling that neither 1 or 9 is special) we know that none of 7, 29, 15 and 3 is in $C$, and 31 and 13 cannot be in $C_3$. If 23 is in $C_3$, then it is an isolated codeword, and also 7 should be a codeword. Therefore 23 is not in $C_3$. By symmetry, 21 is not in $C_3$ either. Consequently, $f_3 \leq 1$. If 11 is not in $C_3$, then $f_3 = 0$ and $f_4 \leq 9$ imply that $f_0 + f_1 + f_2 + f_3 + \frac{1}{2}f_4 \leq 8$. Assume that 11 is in $C_3$. Then 34 is not in $C$, 33 or 35 is in $C$, and therefore 32 or 36 is not in $C_4$. Hence $f_4 \leq 8$ and $f_0 + f_1 + f_2 + f_3 + \frac{1}{2}f_4 \leq 8$. We may therefore assume that 11 is not in $C$.

Then of course 34 and at least one of 33 and 35 are in $C$, and neither 32 nor 36 is in $C_4$.

Assume then that 31 is in $C_3$. Trivially 30 is not in $C_4$. Then in the same way as before we see that none of 23, 7, and 29 is in $C_3$, and at most one of 3, 15, and 13 is in $C_3$. We know that $f_4 \leq 6$, and if $f_3 \leq 2$, we immediately get the inequality $g + \frac{1}{2}h \leq 8$. If $f_3 = 3$, then 21 is in $C_3$ and hence $f_4 \leq 4$ and we again get $g + \frac{1}{2}h \leq 8$. We can therefore assume that 31 is not in $C_3$ and by symmetry that 13 is not in $C_3$ either.

Assume that 29 is in $C_3$. Then 23 and 7 are not, and of course 28 and 30 are not in $C_4$. Because $f_4 \leq 5$, the inequality $g + \frac{1}{2}h \leq 8$ is immediate if $f_3 \leq 2$. Assume

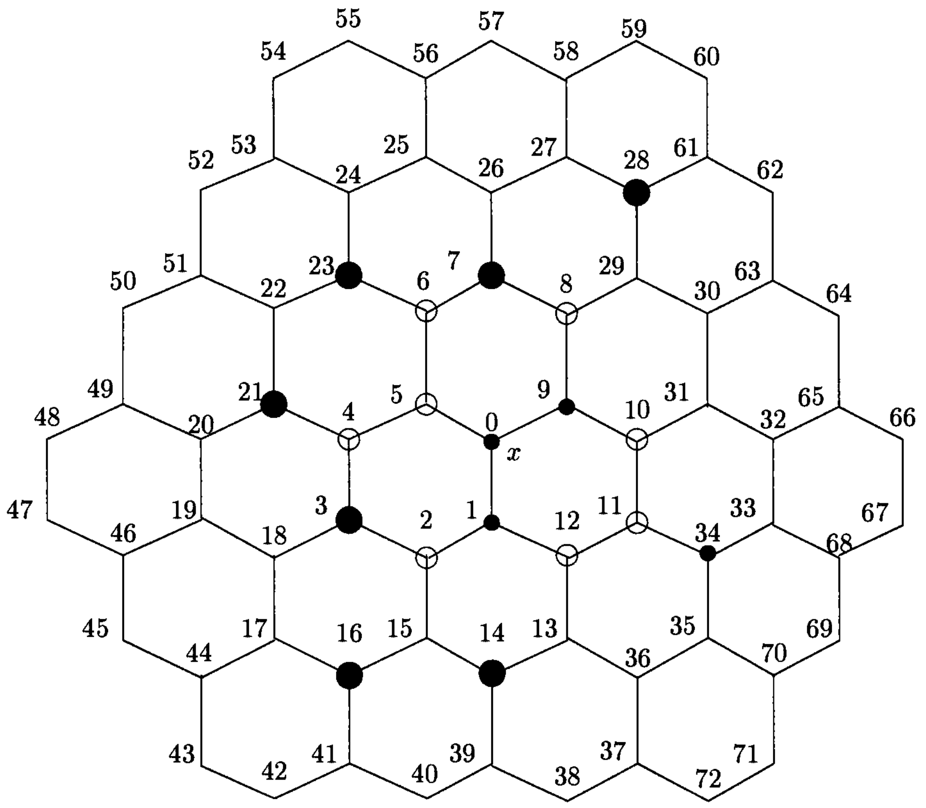


FIG. 7. *The constellation around an exceptional point $x$. Noncodewords have been marked with a circle, codewords with a filled circle, and the codewords in $C_3$ and $C_4$ with a larger filled circle.*

therefore that at least two of 21, 3, and 15 are in $C_3$. Then 21 is in $C_3$, and either 3 or 15 is in $C_3$ but not both. This implies $f_4 \leq 3$, and again $g + \frac{1}{2}h \leq 8$. So we can assume that 29 is not in $C_3$. By symmetry we can assume that 15 is not in $C_3$ either.

Now the only possible elements of $C_3$ are 3, 21, 23, and 7, and the only possible elements of $C_4$ are 14, 16, 20, 22, 24, 28, and 30. Both 14 and 30 cannot be in $C_4$. Indeed, if they are, they are isolated points, and 13 and 31 are not in $C$, which in turn implies that 32 and 36 must be in $C$. But we know that 33 or 35 is in $C$, and therefore 33 or 35 is an excess point. By symmetry, we can assume that 33 is. But then 30 cannot be in $C_4$. Hence $f_4 \leq 6$, and the inequality $g + \frac{1}{2}h \leq 8$ is clear if $f_3 \leq 2$. If $f_3 = 3$, then 21 or 23 is in $C_3$ and $f_4 \leq 4$; and again $g + \frac{1}{2}h \leq 8$. Assume therefore that $f_4 = 4$, i.e., 3, 21, 23, and 7 are all in $C_3$. The only points that can be in $C_4$ are now 14, 16, 28, and 30, and we have shown that both 14 and 30 cannot be in $C_4$. In this case $g + \frac{1}{2}h \leq 17/2$. Apart from this case we always get $g + \frac{1}{2}h \leq 8$. We call points $x \in \mathcal{A}$ for which $g + \frac{1}{2}h = 17/2$ *exceptional*. We will prove later that their effect can be averaged out. We have seen that, apart from obvious symmetries, the constellation around an exceptional point 0 is like that in Figure 7.

*Case* 3: *$x = 0$ is an excess point and does not belong to $C$.* Then 1, 5, and 9 are in $C$. Whether or not they are codewords, none of the points 2, 4, 6, 8, 10, and 12 is in $C_2$. Hence $f_0 = 0$, $f_1 \leq 3$, $f_2 = 0$.

Consider first what we can say of the six-element set consisting of the points 31, 11, 13, 32, 34, 36.

First of all, 34 cannot belong to $C_4$. Indeed, if it did, then it would be an isolated codeword, 11 would not be a codeword, and 10 or 12 would have to be in $C$, because 11 must be covered by a codeword other than 34. By symmetry, assume that 10 is in $C$. Then 9 or 10 is an excess point, and both of them have distance less than four to 34, contradicting the fact that 34 is in $C_4$.

If none of the points 11, 13, 31 is in $C_3$, and we already know that 34 is not in $C_4$, then the contribution of the six points 31, 11, 13, 32, 34, and 36 to the sum $f_0 + f_1 + f_2 + g_3 + \frac{1}{2}h_3 + \frac{1}{2}f_4$ is at most one.

Assume that 11 is not in $C_3$, but 13 is. The case where, instead of 13, we assume that 31 is in $C_3$ is of course symmetrical. Trivially, 36 is not then in $C_4$, and 11 cannot be a codeword either. If 32 is in $C_4$, then 10 is not in $C$. We already know that 11 and 12 are not. Then 33, 34, or 35 is an excess point contradicting the fact that 32 is in $C_4$. If 31 is not in $C_3$, then the six points together contribute at most one. Assume therefore that 31 is in $C_3$. Again, we see that 11 is covered by 34, and 33, 34, or 35 is an excess point. However, since both 31 and 13 are in $C_3$, neither 33 nor 35 can be excess points. Therefore, 34 is an excess point and 31 and 13 are both in $C_3''$. Again the six points together contribute at most one.

Assume then that 11 is in $C_3$. Then none of 31, 10, 12, and 13 is in $C$. In particular 31 and 13 cannot be in $C_3$. Moreover, 34 is not a codeword because 11 is an isolated codeword, but 34 must then be covered by 33 or 35. By symmetry, assume that 33 is in $C$. Then 32 is not in $C_4$. If, anyway, 32 is in $C$, then 32 or 33 is an excess point, which means that 30 cannot be in $C_4$. If 32 is not a codeword, then 31 must be covered by 30, and, moreover, 29 or 63 is a codeword. Again we conclude that 30 cannot be in $C_4$. If we, instead of 33, had assumed that 35 is in $C$, then we would have seen that 14 cannot be in $C_4$. In conclusion, if 11 is in $C_3$, then (i) the contribution of the six points 31, 11, 13, 32, 34, and 36 is at most one, or (ii) the contribution is 3/2 but at least one of the points 14 and 30 is not in $C_4$.

By symmetry, a similar conclusion holds for the sets 15, 3, 21, 16, 18, 20 and 23, 7, 29, 24, 26, 28.

If for all these three six-element sets the contribution is at most one, we are done, because the only other points left are 14, 22, and 30, and their contribution to the sum can be at most 3/2, thus giving us a sum total which is at most 15/2. Assume therefore that the contribution of the set 31, 11, 13, 32, 34, 36 is 3/2. Without loss of generality we can still assume that 30 is then not in $C_4$. The contribution of the set 23, 7, 29, 24, 26, 28 is at most 3/2. If the six points 15, 3, 21, 16, 18, 20 contribute at most one, we are again done, because the two remaining points 14 and 22 together contribute at most one. If 15, 3, 21, 16, 18, 20 together contribute 3/2, then we have shown that at least one of the points 14 and 22 is not in $C_4$, and the sum total is again at most eight.

*Case* 4: $x = 0$ *is a deficient codeword but not an excess point.* This is a relatively simple case. We can assume that 8 and 9 are in $C$, and 1 and 5 are not. Moreover, 4 or 6 is in $C$ (or both); and 2 or 12 is in $C$ (or both). Because 9 is an excess point, 26, 28, 30, 32, and 34 are not in $C_4$, and for the same reason, 7, 29, 31, and 11 are not in $C_3$. Trivially, none of 1, 5, and 9 is in $C_1$. If both 4 and 6 are in $C_2$, then 5 is an excess point and neither 4 nor 6 is in $C_2$. We conclude that at most one of 4 and 6, and similarly at most one of 2 and 12, is in $C_2$, and 8 and 10 are clearly not in $C_2$. Hence $f_0 = 1$, $f_1 = 0$, and $f_2 \leq 2$.

We know that 2 or 12 is in $C$. If 2 is a codeword, then 15 is not in $C_3$, because

2 or 15 is an excess point; and if 12 is a codeword, then 13 is not in $C_3$. Similarly, at most one of 21 and 23 can be in $C_3$. Consequently, $f_3 \leq 3$. We have already shown that $f_4 \leq 7$. If $f_3 = 0$, then the sum $f_0 + f_1 + f_2 + f_3 + \frac{1}{2} f_4$ is at most 13/2. Assume that $f_3 > 0$. If, for instance, 3 is in $C_3$, then of course 18 is not in $C_4$. We see that if $f_3 > 0$, then $f_4 \leq 6$. Therefore, if $f_3 = 1$ or 2, the sum is again at most eight. Finally, if $f_3 = 3$, then one of 21 and 23, one of 13 and 15, and 3 are in $C_3$, and $f_4 \leq 2$, and $g + \frac{1}{2} h \leq 7$. □

Denote for $i = 1, 2, 3, 4$

$$L_i = \{x \in V : |B_1(x) \cap C| = i\},$$

and

$$\ell_i = |L_i|.$$

Then

$$|\mathcal{E}| = \sum_{i \geq 3} \ell_i.$$

Moreover,

$$\ell_1 = |C| - |\mathcal{D}|.$$

Indeed, by definition, $\ell_1$ equals the number of pairs $(x, c)$ such that $x \in L_1$, $c \in C$, $d(x, c) \leq 1$; because $C$ is identifying, there is at most one $x$ for each $c$, and by definition such a vertex $x$ exists if and only if $c$ is not deficient. Consequently,

$$\begin{aligned} 4|C| &= \sum_{i=1}^{4} i\ell_i \\ &\geq \ell_1 + 2(|V| - \ell_1) + |\mathcal{E}| \\ &= 2|V| - \ell_1 + |\mathcal{E}| \\ &= 2|V| - |C| + |\mathcal{D}| + |\mathcal{E}| \\ &= 2|V| - |C| + |\mathcal{A}| + |\mathcal{D} \cap \mathcal{E}| \\ &= 2|V| - |C| + |\mathcal{A}_1| + 2|\mathcal{A}_2|, \end{aligned}$$

and therefore

$$e(C) \geq |\mathcal{A}_1| + 2|\mathcal{A}_2|. \tag{4.1}$$

LEMMA 4.5.

$$e(C) \geq \frac{2}{17} |C|.$$

*Proof.* We write a list of pairs $(c, x)$, $c \in C$, $x \in \mathcal{A}$ in the following way. For each $c \in C'$ we write the unique pair $(c, x)$, where $c \in C'_i$, $x \in \mathcal{A}_1$, and $d(c, x) = i$, to the list. For each $c \in C''$ we write all the pairs $(c, x)$, where $c \in C''_i$, $x \in \mathcal{A}$, and $d(c, x) = i$, *and, moreover, we write the pair a second time in the list if* $x \in \mathcal{A}_2$.

We now estimate in two ways the quantity $2a + b$, where $a$ denotes the number of pairs $(c, x)$ in the list where $c \in C'$ and $b$ denotes the number of pairs $(c, x)$ in the list where $c \in C''$.

By the construction, $2a+b$ is at least $2|C'|+2|C''|$.

On the other hand, given $x \in \mathcal{A}_1$ (resp., $\mathcal{A}_2$), two times the number of pairs $(c,x)$ where $c \in C'$ plus the number of pairs $(c,x)$ where $c \in C''$ is at most 17 (resp., 34) by the previous lemma. Together with (4.1) this implies that

$$17e(C) \geq 17(|\mathcal{A}_1| + 2|\mathcal{A}_2|) \geq 2a+b \geq 2|C'|+2|C''| = 2|C|,$$

proving the claim. ∎

This already implies that $D \geq \frac{34}{83} \approx 0.4096$. We now attack the exceptional points.

LEMMA 4.6.

$$e(C) \geq \frac{1}{8}|C|.$$

*Proof.* In the proof of Lemma 4.4 we saw that for all $x \in \mathcal{A}$ we have $g + \frac{1}{2}h \leq 8$, except for the *exceptional* points defined at the end of Case 2.

Recall the constellation around an exceptional point. Assume that $x$ is at 0. By symmetry we can assume that 0, 1, 9, and 34 are in $C$; 2, 4, 5, 6, 8, 10, 11, and 12 are not in $C$; 3, 7, 21, and 23 are in $C_3$; 14, 16, and 28 are in $C_4$. Then 29 is not in $C$, 30 is an isolated codeword (to cover 29), and 31 and 63 are not codewords. Then 32 is in $C$. Similarly, 36 is an isolated codeword, and 13, 35, and 37 are not codewords. Now 33 is a codeword and an excess point.

We see that for each exceptional $x$ we can define a unique *friend* $y \in \mathcal{E} \cap C$ as follows. If $x_1$ and $x_2$ are the codeword neighbors of $x$, there are two points $z_1$ and $z_2$ such that $B_1(z_1) \cap C = \{x_1\}$ and $B_1(z_2) \cap C = \{x_2\}$. The points $z_1$ and $z_2$ have a common neighbor $z$, which is covered by a single codeword $c$. This codeword $c$ has a unique codeword neighbor $y$ (point 33), which is an excess point. We define that this point $y$ is the friend of $x$.

Assume that we know that a given point $y$ is the friend of $x$, and we want to find $x$. By reversing the process above, we see that the number of possibilities for $x$ is at most the number of codeword neighbors of $y$. Indeed, for each codeword neighbor $c$ of $y$ there is at most one point $z$ such that $B_1(z) \cap C = \{c\}$, and once we know $c$ and $z$, there is a unique location for $x$.

Hence at most three exceptional points can share a friend $y$. If $y$ is shared by more than two exceptional points, then $y \in \mathcal{A}_2$. We say that the friend $y$ together with the at most three exceptional points that have $y$ as their friend form a *family.*

We now show that if $y$ is the friend of an exceptional point, then (where now $g$ and $h$ refer to $y$ instead of $x$)

(1) $g + \frac{1}{2}h \leq 7$ if $y \in \mathcal{A}_1$, and

(2) $g + \frac{1}{2}h \leq 6$ if $y \in \mathcal{A}_2$.

This will give the result of the lemma. Namely, in the proof of the previous lemma we estimated that $17(|\mathcal{A}_1| + 2|\mathcal{A}_2|) \geq 2a+b$. We can now sharpen this estimate by replacing 17 by 16. Indeed, we consider the families separately. If the friend $y$ in the family belongs to $\mathcal{A}_1$, then it is the friend of at most two exceptional points from $\mathcal{A}_1$, and on average $g + \frac{1}{2}h \leq 8$ for these points by (1). If $y \in \mathcal{A}_2$ and is the friend of $j \leq 3$ exceptional points, the claim follows from (2) because $17j + 24 \leq 16j + 32$ for all $j = 1, 2, 3$.

It therefore suffices to prove (1) and (2).

To view the situation in the by now familiar set-up, relabel the points so that $y$ becomes 0 and $x$ becomes 33. Then we know that the only points of $S_4(y)$ that can be in $C_4$ are 16, 18, 20, 22, 24, 26, and the only points of $S_3(y)$ that can be in $C_3$ are 13, 15, 3, 21, 23, 7, 29.

To prove (1), we assume that $y \in \mathcal{A}_1$ and therefore none of 4, 5, and 6 is in $C$. Then $f_0 + f_1 + f_2 \leq 3$. From the constellation around $y$ we see that 29 and 13 are in $C$. This implies that none of the points 7, 3, and 15 can be in $C_3$; if 3 or 15 were in $C_3$, then 1 would be a deficient codeword at distance two from 3 and 15. If 21 is in $C_3$, it is an isolated codeword and 3 must be in $C$, which implies that 13 is not in $C_3$. We also see that if 23 is in $C_3$, then 29 is not. Hence $f_3 \leq 2$. If 24 and 26 both were in $C_4$, they would be isolated codewords, and the point 6 would not be covered at all. The same argument shows that 18 and 20 cannot both be in $C_4$. Hence $f_4 \leq 4$, and $f_0 + f_1 + f_2 + f_3 + \frac{1}{2}f_4 \leq 7$, proving (1).

To prove (2), we recall that $y \in \mathcal{A}_2$ implies that $f_0 + f_1 + f_2 \leq 4$. Moreover, since in the pairs $(15, 16)$, $(3, 18)$, $(21, 20)$, $(23, 24)$, and $(7, 26)$ the first component may belong to $C_3$ and the second may belong to $C_4$ but not both, we immediately see that $f_3 + f_4 \leq 8$, proving (2), because for a point $y \in \mathcal{A}_2$, $f_i = h_i$ for all $i$. $\square$

Theorem 4.7.

$$D \geq \frac{16}{39} \approx 0.4102\ldots.$$

In the previous theorem $D$ refers to the graph $H_m$, but as we have seen in section 3, the same conclusion also holds for the infinite hexagonal grid, and we therefore obtain the following theorem.

Theorem 4.8. *The density of any identifying code in the infinite hexagonal grid is at least* 16/39.

## REFERENCES

[1] U. Blass, I. Honkala, and S. Litsyn, *Bounds on identifying codes*, Discrete Math., to appear.

[2] U. Blass, I. Honkala, and S. Litsyn, *On binary codes for identification*, J. Combin. Des., 8 (2000), pp. 151–156.

[3] G. Cohen, S. Gravier, I. Honkala, A. Lobstein, M. Mollard, C. Payan, and G. Zémor, *Improved identifying codes for the grid*, Electron. J. Combin., Comments to 6 (1999), R19, http://www.combinatorics.org/Volume_6/Html/v6i1r19.html.

[4] G. Cohen, I. Honkala, A. Lobstein, and G. Zémor, *New bounds for codes identifying vertices in graphs*, Electron. J. Combin., 6 (1999), R19, http://www.combinatorics.org/Volume_6/PDF/v6i1r19.pdf.

[5] G. Cohen, I. Honkala, A. Lobstein, and G. Zémor, *On codes identifying vertices in the two-dimensional square lattice with diagonals*, IEEE Trans. Comput., to appear.

[6] G. Cohen, I. Honkala, A. Lobstein, and G. Zémor, *On identifying codes*, submitted to the Proceedings of the DIMACS Workshop on Codes and Association Schemes, Rutgers University, Piscataway, NJ, 1999.

[7] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Fundamentals of Domination in Graphs*, Marcel Dekker, New York, 1998.

[8] M. G. Karpovsky, K. Chakrabarty, and L. B. Levitin, *On a new class of codes for identifying vertices in graphs*, IEEE Trans. Inform. Theory, 44 (1998), pp. 599–611.

[9] M. G. Karpovsky, K. Chakrabarty, L. B. Levitin, and D. R. Avresky, *On the covering of vertices for fault diagnosis in hypercubes*, Inform. Process. Lett., 69 (1999), pp. 99–103.

SIAM J. DISCRETE MATH.
Vol. 13, No. 4, pp. 505–520

# APPROXIMATING FRACTIONAL MULTICOMMODITY FLOW INDEPENDENT OF THE NUMBER OF COMMODITIES[*]

LISA K. FLEISCHER[†]

**Abstract.** We describe fully polynomial time approximation schemes for various multicommodity flow problems in graphs with $m$ edges and $n$ vertices. We present the first approximation scheme for maximum multicommodity flow that is independent of the number of commodities $k$, and our algorithm improves upon the run time of previous algorithms by this factor of $k$, running in $\mathcal{O}^*(\epsilon^{-2}m^2)$ time. For maximum concurrent flow and minimum cost concurrent flow, we present algorithms that are faster than the current known algorithms when the graph is sparse or the number of commodities $k$ is large, i.e., $k > m/n$. Our algorithms build on the framework proposed by Garg and Könemann [*Proceedings of the* 39*th Annual IEEE Symposium on Foundations of Computer Science*, IEEE, New York, 1998, pp. 300–309]. They are simple, deterministic, and for the versions without costs, they are strongly polynomial. The approximation guarantees are obtained by comparison with dual feasible solutions found by our algorithm.

Our maximum multicommodity flow algorithm extends to an approximation scheme for the maximum weighted multicommodity flow, which is faster than those implied by previous algorithms by a factor of $k/\log W$, where $W$ is the maximum weight of a commodity.

**Key words.** multicommodity flow, approximation algorithm, concurrent flow, VLSI routing

**AMS subject classifications.** 68Q25, 90C08, 90C27, 90C35, 90C59

**PII.** S0895480199355754

## 1. Introduction.

A multicommodity flow problem is defined on a directed network $G = (V, E)$ with capacities $u : E \to \mathbf{R}$ and $k$ source sink terminal pairs $(s_j, t_j)$, $1 \le j \le k$. The problem is to find flows $f_j$ from $s_j$ to $t_j$ that satisfy node conservation constraints and meet some objective function criteria so that the sum of flows on any edge does not exceed the capacity of the edge. Let $|f_j|$ denote the amount of flow sent from $s_j$ to $t_j$ in $f_j$. For the *maximum multicommodity flow* problem, the objective is to maximize the sum of the flows: $\max \sum_j |f_j|$. For the *maximum concurrent flow* problem, there are demands $d_j$ associated with each commodity $j$, and the objective is to satisfy the maximum possible proportion of all demands: $\max \lambda,\ |f_j| \ge \lambda d_j,\ \forall j$. If there are costs $c(e)$ associated with a unit of flow on edge $e$, the minimum cost concurrent flow problem is to find a maximum concurrent flow of minimum cost, where the cost of a flow is the cost of sending flow on each edge of the graph summed over all the edges carrying flow.

Our goal is to find an *$\epsilon$-approximate solution* for any error parameter $\epsilon > 0$. For the maximization problems, an $\epsilon$-approximate solution is a flow that has value at least $(1-\epsilon)$ times the maximum value. For versions with costs, an $\epsilon$-approximate solution is flow that has value at least $(1-\epsilon)$ times the maximum value and has cost at most the optimal cost. For each problem discussed in this paper, we describe a *fully*

[*]Received by the editors May 4, 1999; accepted for publication (in revised form) May 19, 2000; published electronically October 25, 2000. This research was partially supported by NSF grant EIA-9973858. This paper was written while the author was at the Department of Industrial Engineering and Operations Research at Columbia University and on leave at Center for Operations Research and Econometrics, Louvain-la-Neuve, Belgium. An extended abstract of this paper appeared in *Proceedings of the* 40*th Annual IEEE Symposium on Foundations of Computer Science*, IEEE, New York, 1999, pp. 24–31.
http://www.siam.org/journals/sidma/13-4/35575.html

[†]Graduate School of Industrial Administration, Posner 239, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213 (lkf@andrew.cmu.edu).

TABLE 1.1
*Comparison of multicommodity flow FPTAS. $\mathcal{O}^*()$ hides polylog$(m)$. $I := \log M$.*

| Problem | Previous best | This paper |
|---|---|---|
| max multiflow | $\mathcal{O}^*(\epsilon^{-2}km^2)$ [14, 9][1] | $\mathcal{O}^*(\epsilon^{-2}m^2)$ |
| max concurrent flow | $\mathcal{O}^*(\epsilon^{-2}m(m+k)+k$ max flows) [9][1] $\mathcal{O}^*(\epsilon^{-2}kmn)$ [18, 23] | $\mathcal{O}^*(\epsilon^{-2}m(m+k))$ |
| min cost concurrent flow | $\mathcal{O}^*((\epsilon^{-2}m(m+k)+kmn)I)$ [9][1] $\mathcal{O}^*(\epsilon^{-2}kmnI)$ [13] | $\mathcal{O}^*(\epsilon^{-2}m(m+k)I)$ |

*polynomial time approximation scheme* (FPTAS) to solve the problem. A FPTAS is a family of algorithms that finds an $\epsilon$-approximate solution in time polynomial in the size of the input and $1/\epsilon$. Here, the size of the input is specified by the number of nodes $n$, the number of arcs $m$, and the space needed in a binary representation of the largest integer $M$ used to specify any of the capacities, costs, and demands. To simplify the run times, we use $\mathcal{O}^*(f)$ to denote $f \log^{\mathcal{O}(1)} m$.

There has been a series of papers providing FPTASs for multicommodity flow problems and generalizations. We discuss these briefly below. These approximation schemes are all iterative algorithms modeled on Lagrangian relaxations and linear programming decomposition techniques. Preceding this work is a 1973 paper by Fratta, Gerla, and Kleinrock that uses very similar techniques for solving minimum cost multicommodity flow problems [7]. While they do not discuss approximation guarantees, it appears that the flow deviation method introduced in their paper yields an approximation guarantee for this problem with only minor adjustments [4]. Shahrokhi and Matula [25] give the first polynomial time combinatorial algorithm for approximating the maximum concurrent flow problem with uniform capacities, and introduce the use of an exponential length function to model the congestion of flow on an edge. Klein et al. [17] improve the complexity of this algorithm using randomization. Leighton et al. [18] extend [17] to handle graphs with arbitrary capacities, and give improved run times when capacities are uniform. None of these papers considers the versions with costs. Grigoriadis and Khachiyan [12] describe approximation schemes for block angular linear programs that generalize the uniform capacity maximum and minimum cost concurrent flow problems. Plotkin, Shmoys, and Tardos [22] formulate more general approximation schemes for fractional packing and covering problems, and describe an approximation scheme for the minimum cost concurrent flow with general capacities. They also obtain improved run times when all capacities are uniform. These last three papers all observe that it is not necessary to exactly solve the subproblems generated at each iteration; it suffices to obtain an approximate solution.

The most theoretically efficient algorithms discussed in the above references are randomized algorithms, although several also describe slower deterministic versions. Radzik [23] gives the first deterministic algorithm to match the run times of the fastest existing randomized algorithms for the maximum concurrent flow problem. His algorithm uses a “round-robin” approach to routing commodities. Karger and Plotkin [16] use this idea to obtain deterministic algorithms for minimum cost concurrent flow that reduce the dependence of deterministic algorithms on $\epsilon$. For fixed $\epsilon$, their algorithm also improves upon the fastest randomized algorithms. Grigoriadis and Khachiyan [13] reduce the dependence on $\epsilon$ further to the current best known $\epsilon^{-2}$ with a specialized version of their initial algorithm. To get this improvement, they

[1]The extended abstract [9] makes stronger claims, but the actual achievable run times are correctly stated here [8].

use the fact that it is sufficient to solve the subproblems approximately.

All of the above algorithms compute initial flows and then reroute flow from more congested paths to less congested paths. Young [27] describes a randomized algorithm that works by augmenting flow along shortest paths using the exponential length function, instead of augmenting by single commodity minimum cost flows. Shmoys [26] explains how the framework of [22] can be used to approximately solve the maximum multicommodity flow problem. The subproblem he uses is also a shortest path problem. Grigoriadis and Khachiyan [14] reduce the run time for approximately solving this problem by a factor of $1/\epsilon$ using a logarithmic instead of exponential potential function. Their algorithm can also provide approximation guarantees for the minimum cost version.

Recently, Garg and Könemann [9] give simple, deterministic algorithms to solve the maximum multicommodity flow problem, the concurrent flow problem, and the versions with costs. Like the work in [27], they augment flow on shortest paths. For the maximum multicommodity flow problem, their algorithm matches the complexity in [14]. They obtain a small improvement in run time for the concurrent flow problems. Their main contribution is to provide a very simple analysis for the correctness of their algorithms, and a simple framework for positive packing problems.

We present faster approximation schemes for maximum multicommodity flow, maximum concurrent flow, and their minimum cost versions. For the maximum multicommodity flow problem we give the first approximation scheme with run time that is independent of the number of commodities. It is faster than the best previous approximation schemes by the number of commodities, $k$. For the maximum concurrent flow problem, we describe an algorithm that is faster than the best previous approximation schemes when the graph is sparse or there are a large number of commodities. In particular, our algorithm is faster when $k > m/n$. We obtain similar improvements for the minimum cost versions. See Table 1.1 for comparison with previous work. Our algorithms are deterministic and build on the framework proposed in [9].

Our algorithms provide their approximation guarantees by simultaneously finding solutions to the dual linear programs. We show that our primal solutions are within $1-\epsilon$ of the dual solutions obtained, and hence both the primal solutions and the dual solutions are $\epsilon$-approximate. We discuss the interpretation of the dual problems at the very end of this section.

Fractional multicommodity flow problems can be solved in polynomial time by linear programming techniques. However, in many applications these problems are typically quite large and can take a long time to solve using these techniques. For this reason, it is useful to develop faster algorithms that deliver solutions that are provably close to optimal. Experimental results to date suggest that these techniques can lead to significantly faster solution times. For example, Bienstock has reported significant speedups in obtaining approximate and often exact optimal solutions to block decomposable linear programs by building on the $\epsilon$-approximation methods described in [22, 12]. There has also been earlier experimental work including [11, 13, 20]. In [11], they show that certain aspects of the approximation schemes need to be fine tuned to obtain good performance in practice. For example, one aspect is the choice of step size in each iteration. In the theoretical work [13, 22], each iteration involves computing a new flow for a commodity (an improving direction) and then moving to a new solution that is a convex combination of the old flow and the new flow. The step size is determined theoretically by parameters of the algorithm. Goldberg et al. [11] show that in practice it is better to compute the optimal step size at each iteration.

The focus of the current paper is not to explore directly the experimental efficiency but instead to provide theoretical improvements that are simple to implement and thus may improve experimental performance. Thus we ignore details such as step size and other issues here so that our presentation may be easier to follow. However, since the publication of an extended abstract of this paper [6], these ideas have been tested and shown to lead to demonstrable improvements in practice [1, 5, 24].

One area of application for obtaining quick approximate solutions to fractional multicommodity flow problems is the field of network design: in VLSI design [1, 5] or in design of telecommunication networks [3]. Given pairwise demands, it is desired to build a network with enough capacity to route all demand. The network design problems encountered in practice are typically NP-hard and difficult to solve. In addition, it is often desired to develop many solutions for many different scenarios. A fast multicommodity flow algorithm permits the designer to quickly test if the planned network is feasible and proceed accordingly. These algorithms can also be incorporated in a branch-and-cut framework to optimize network design problems. For example, see Bienstock [3].

Another area of application is interest in solving the dual problems. The integer versions of the dual problems to multicommodity flow problems are NP-hard. There are approximation algorithms for these problems that guarantee a solution within a logarithmic factor of the optimal solution. These algorithms all start with a solution to the linear program and round this solution. The guarantees are obtained by comparing the resulting integer solution with the LP bound. Since the approximation guarantees for these algorithms is much larger than $\epsilon$, the worst-case performance guarantee does not erode by starting from an $\epsilon$-approximate solution to the linear program instead of an exact solution. Since an $\epsilon$-approximate solution is much easier to obtain, this improves the run time of these algorithms. We briefly describe the integer dual problems below.

The integer version of the dual to the maximum multicommodity flow problem is the multicut problem. The *multicut problem* is, given $(s_j, t_j)$ pairs, to find a minimum capacity set of edges whose removal disconnects the graph so that $s_j$ is in a different component than $t_j$ for all pairs $j$. Garg, Vazirani, and Yannakakis [10] describe an algorithm for the multicut problem that returns a solution that is provably close to the optimal solution. Their algorithm rounds the fractional solution to the LP relaxation that is the dual of the maximum multicommodity flow problem. The integer version of the dual to the maximum concurrent flow problem is the sparsest cut problem. The *sparsest cut problem* is to find the set $S$ so that the ratio of the capacity of edges leaving $S$ divided by the sum of demands of demand pairs with one end in $S$ and the other outside $S$ is minimized. There have been several approximation results for this problem that again round the fractional solution to the LP relaxation of the sparsest cut problem; the most recent algorithms are given by London, Linial, and Rabinovich [21] and Aumann and Rabani [2]. The sparsest cut problem arises as a subroutine in an algorithm for finding an approximately optimal balanced cut [19]. The *balanced cut problem* is to partition the vertices of a graph into two sets of size at least $|V|/3$, so that the total capacity of the edges that have one endpoint in each set is minimized.

## 2. Maximum multicommodity flow.

Our work builds directly on the simple analysis given in [9]. We use the path-flow linear programming formulation of the maximum multicommodity flow problem. Let $\mathcal{P}_j$ denote the set of paths from $s_j$ to

$t_j$, and let $\mathcal{P} := \cup_j \mathcal{P}_j$. Variable $x(P)$ equals the amount of flow sent along path $P$.

The linear programming formulation is then

$$\text{(P)} \qquad \begin{array}{llll} \max & \sum_{P \in \mathcal{P}} x(P) & & \\ \forall e: & \sum\limits_{P:e \in P} x(P) & \leq & u(e) \\ \forall P: & x(P) & \geq & 0. \end{array}$$

The dual to this linear program corresponds to the problem of assigning lengths to the edges of the graph so that the length of the shortest path from $s_j$ to $t_j$ is at least 1 for all commodities $j$. The length of an edge represents the marginal cost of using an additional unit of capacity of the edge.

$$\text{(D)} \qquad \begin{array}{llll} \min & \sum_e u(e)l(e) & & \\ \forall P: & \sum\limits_{e \in P} l(e) & \geq & 1 \\ \forall e: & l(e) & \geq & 0. \end{array}$$

While neither of these linear programs are polynomially sized, they can both be solved in polynomial time. One way to see this is to realize that there are polynomially sized formulations of both problems: for example, the arc-flow formulation of the maximum multiflow problem has only $mk$ variables and $O((n+m)k)$ constraints. Another way to see this is to realize that the polytope described by the constraints in (D) has a polynomial time separation algorithm. This implies that (D) can be solved in polynomial time using the ellipsoid algorithm, via the equivalence of polynomial time separation and polynomial time optimization for convex polytopes, established by Grötschel, Lovász, and Schrijver [15]. This in turn implies that the primal can also be solved in polynomial time. Given a candidate vector $l$, the separation algorithm for (D) is to compute the shortest path for each commodity using $l$ as the length function. If there is a path with length less than one, this is a violated inequality. Otherwise $l$ satisfies all constraints.

For large problems it is often desirable to have faster algorithms. We now describe a fast algorithm for obtaining $\epsilon$-approximate solutions to (P) and (D). The Garg–Könemann algorithm starts with length function $l \equiv \delta$ for an appropriately small $\delta > 0$ depending on $m$ and $\epsilon$, and with a primal solution $x \equiv 0$. While there is a path in $\mathcal{P}$ of length less than 1, the algorithm selects such a path and increases both the primal and the dual variables associated with this path as follows. For the primal problem, the algorithm augments flow along this path. The amount of flow sent along path $P$ is determined by the bottleneck capacity of the path, using the *original* capacities. The *bottleneck* capacity of a path is the capacity of the minimum capacity edge in the path. Denote this capacity by $u$. The primal solution is updated by setting $x(P) = x(P) + u$. Note that this solution may be infeasible since it will likely violate capacity constraints. However, it satisfies nonnegativity constraints. Once we determine the most violated capacity constraint, we can scale the primal solution so that it is feasible by dividing all variables by the appropriate scalar. Thus, at any point in the algorithm we can associate our current solution with a feasible solution. Since we will show that our algorithm has only a polynomial number of iterations, and each iteration increases $x(P)$ for just one $P$, it is also easy to compute the appropriate scalar since there are always only a polynomial number of nonzero variables. (Alternatively, we may keep track of flows by keeping track of the flow on each edge instead of the flow on each path. This is a natural approach to implementing our algorithm, and makes computing the scale factor for feasibility a very simple task.

```
Input: network $G$, capacities $u(e)$, commodity pairs $(s_j, t_j)$, $1 \le j \le k$,
       accuracy $\epsilon$
Output: primal (infeasible) and dual solutions $x$ and $l$

Initialize $l(e) = \delta\ \forall e,\ x \equiv 0$.
while there is a $P \in \mathcal{P}$ with $l(P) < 1$
    Select a $P \in \mathcal{P}$ with $l(P) < 1$
    $u \leftarrow \min_{e \in P} u(e)$
    $x(P) \leftarrow x(P) + u$
    $\forall e \in P,\ l(e) \leftarrow l(e)(1 + \frac{\epsilon u}{u(e)})$
end while
Return $(x, l)$.
```

FIG. 2.1. *Generic algorithm for maximum multicommodity flow.*

However, for simplicity of presentation, the algorithm described below will keep track of the values $x(P)$ only.)

After updating $x(P)$, the dual variables are updated so that the length of an edge is exponential in the congestion of the edge. For edge $e$ on $P$, the update is $l(e) = l(e)(1 + \frac{\epsilon u}{u(e)})$. The lengths of edges not on $P$ remain unchanged. This update ensures that the length of the bottleneck edge on $P$ increases by a factor of $(1+\epsilon)$. We refer to this algorithm as the *generic algorithm*. It is summarized in Figure 2.1.

At any point in the algorithm we can also find the most violated dual constraint (via shortest path computations using lengths $l(e)$) and scale the dual solution so that it is feasible for (D) by multiplying all variables by the proportion of violation. Thus, at the end of every iteration we have implicit primal and dual feasible solutions. While we use dual feasibility as a termination criterion, this is done only for simplicity of our arguments, and is not required for correctness of the algorithm. In practice, it would make sense to keep track of the best dual solution encountered in the algorithm and terminate the algorithm when the ratio between this and the best primal solution is at most $1+\epsilon$. Our analysis will show that this always happens by the time the length of every path in $\mathcal{P}$ is at least one. (See the proof of Theorem 2.4 and the ensuing discussion for further details.) The following two lemmas imply that the algorithm does not require too many iterations, and that the final primal solution is not too far from feasible.

LEMMA 2.1. *After $\mathcal{O}(m \log_{1+\epsilon} \frac{1+\epsilon}{\delta})$ augmentations, the generic algorithm terminates.*

*Proof.* At start, $l(e) = \delta$ for all edges $e$. The last time the length of an edge is updated, it is on a path of length less than one, and it is increased by at most a factor of $1+\epsilon$. Thus the final length of any edge is at most $1+\epsilon$. Since every augmentation increases the length of some edge by a factor of at least $1+\epsilon$, the number of possible augmentations is at most $m \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$. ☐

LEMMA 2.2. *The flow obtained by scaling the final flow obtained in the generic algorithm by $\log_{1+\epsilon} \frac{1+\epsilon}{\delta}$ is feasible.*

*Proof.* Every time the total flow on an edge increases by a fraction $0 < a_i \le 1$ of its capacity, its length is multiplied by $1 + a_i\epsilon$. Since $1 + a\epsilon \ge (1+\epsilon)^a\ \forall\ 0 \le a \le 1$, we have $\Pi_i(1 + a_i\epsilon) \ge (1+\epsilon)^{\sum_i a_i}$ when $0 \le a_i \le 1\ \forall\ i$. Thus, every time the flow on an edge increases by its capacity, the length of the edge increases by a factor of at least $1+\epsilon$. Initially $l(e)=\delta$ and at the end $l(e)<1+\epsilon$, so the total flow on edge $e$ cannot exceed $u(e)\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$. ∎

The key to the efficiency of our algorithm lies in our selection of $P$. Garg and Könemann [9] show that if $P$ is selected so that $P=\operatorname{argmin}_{P\in\mathcal{P}} l(P)$, where $l(P)=\sum_{e\in P} l(e)$, then the following theorem holds. We omit the proof here since our proof of Theorem 2.4 provided in the subsequent section is a straightforward modification of the proof of this theorem.

THEOREM 2.3. *If $P$ is selected in each iteration to be the shortest $(s_i,t_i)$ path among all commodities, then for a final flow value $g_t$ we have that $\frac{g_t}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}\ge(1-2\epsilon)OPT$.*

To determine the minimum length path in $\mathcal{P}$, it is necessary to compute a shortest path for each commodity. This takes a total of $\mathcal{O}^*(km)$ time. Choosing $\delta=(1+\epsilon)/((1+\epsilon)n)^{1/\epsilon}$ then implies an $\mathcal{O}^*(\epsilon^{-2}km^2)$ time approximation scheme for maximum multicommodity flow.

**2.1. Our improvement.** We improve upon this algorithm by selecting $P$ in a less costly manner. Instead of finding the shortest path in $\mathcal{P}$, we settle for some path with length at most $(1+\epsilon)$ times the length of the shortest path and show that we can obtain a similar approximation guarantee.

Given a length function $l$, define $\alpha(l):=\min_{P\in\mathcal{P}} l(P)$. Denote the length function at the end of iteration $i$ by $l_i$, and for ease of notation let $\alpha(i)=\alpha(l_i)$. We show below in Theorem 2.4 that by augmenting in iteration $i$ along a path $P$ such that $l(P)\le(1+\epsilon)\alpha(i)$, the number of iterations is unchanged, and the final scaled flow has value at least $(1-4\epsilon)$OPT.

This enables us to modify the algorithm by reducing the number of shortest path computations. Instead of looking at all commodities to see which source sink pair is the closest according to the current length function, we cycle through the commodities, sticking with one commodity until the shortest source-to-sink path for that commodity is above a $1+\epsilon$ factor times a lower bound estimate of the overall shortest path. Let $\hat{\alpha}(i)$ be a lower bound on $\alpha(i)$. To start, we set $\hat{\alpha}(0)=\delta$. As long as there is some $P\in\mathcal{P}$ with $l(P)<\min\{1,(1+\epsilon)\hat{\alpha}(i)\}$, we augment flow along $P$, and set $\hat{\alpha}(i+1)=\hat{\alpha}(i)$. When this no longer holds, we know that the length of the shortest path is at least $(1+\epsilon)\hat{\alpha}(i)$, and so we set $\hat{\alpha}(i+1)=(1+\epsilon)\hat{\alpha}(i)$. Thus, throughout the course of the algorithm, $\hat{\alpha}$ takes on values in the set $\{\delta(1+\epsilon)^r\}_{r\in\mathcal{N}}$. Let $t$ be the final iteration. Since $\alpha(0)\ge\delta$ and $\alpha(t-1)<1$, we have $\alpha(t)<1+\epsilon$. Thus, when we stop, $\hat{\alpha}(t)$ is between 1 and $1+\epsilon$. Since each time we increase $\hat{\alpha}$, we increase it by a $1+\epsilon$ factor, the number of times that we increase $\hat{\alpha}$ is $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ (which implies that the final value of $r$ is $\lfloor\log_{1+\epsilon}\frac{1+\epsilon}{\delta}\rfloor$, where $\lfloor z\rfloor$ denotes the largest integer $\le z$.).

Between updates to $\hat{\alpha}$, the algorithm proceeds by considering each commodity one by one. As long as the shortest path for commodity $j$ has length less than the minimum of $1+\epsilon$ times the current value of $\hat{\alpha}$ and 1, flow is augmented along such a shortest path. When $\min_{P\in\mathcal{P}_j} l(P)$ is at least $(1+\epsilon)\hat{\alpha}$, commodity $j+1$ is considered. After all $k$ commodities are considered, we know that $\alpha(i)\ge(1+\epsilon)\hat{\alpha}(i)$ so we update $\hat{\alpha}$ by setting $\hat{\alpha}(i+1)=(1+\epsilon)\hat{\alpha}(i)$. We implement this idea by defining phases determined by the values of $\hat{\alpha}$, starting with $\hat{\alpha}=\delta$ and ending with $\hat{\alpha}=\delta(1+\epsilon)^r$ for $r$ such that $1\le\delta(1+\epsilon)^r<1+\epsilon$. This algorithm is presented in Figure 2.2.

Between each update to $\hat{\alpha}$, there is at most one shortest path computation per commodity that does not lead to an augmentation (For commodity $j$ this is the computation that reveals that $\min_{P\in\mathcal{P}_j} l(P)\ge(1+\epsilon)\hat{\alpha}$.) We charge these computations

```
Input: network $G$, capacities $u(e)$, commodity pairs $(s_j, t_j)$, $1 \le j \le k$,
       accuracy $\epsilon$
Output: primal (infeasible) and dual solutions $x$ and $l$

    Initialize $l(e) = \delta\ \forall e,\ x \equiv 0$.
    for $r = 1$ to $\lfloor \log_{1+\epsilon} \frac{1+\epsilon}{\delta} \rfloor$
        for $j = 1$ to $k$ do
            $P \leftarrow$ shortest path in $\mathcal{P}_j$ using $l$.
            while $l(P) < \min\{1, \delta(1+\epsilon)^r\}$
                $u \leftarrow \min_{e \in P} u(e)$
                $x(P) \leftarrow x(P) + u$
                $\forall e \in P,\ l(e) \leftarrow l(e)(1 + \frac{\epsilon u}{u(e)})$
                $P \leftarrow$ shortest path in $\mathcal{P}_j$ using $l$.
            end while
    Return $(x, l)$.
```

FIG. 2.2. *FPTAS for maximum multicommodity flow. Here $\hat{\alpha}$ is represented implicitly as $\delta(1+\epsilon)^r$.*

to the increase of $\hat{\alpha}$. Thus a total of at most $k \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$ shortest path computations are used to update $\hat{\alpha}$ over the course of the algorithm. The remaining shortest path computations all lead to augmentations. Lemma 2.1 enables us to bound the number of these computations by $\mathcal{O}^*(m \log_{1+\epsilon} \frac{1+\epsilon}{\delta})$. Using a Dijkstra shortest path algorithm, this implies a run time of $\mathcal{O}^*(\epsilon^{-2}(m^2 + km))$ for this modified algorithm. This can be reduced to $\mathcal{O}^*(\epsilon^{-2}m^2)$ by observing that we can group commodities by a common source node, and compute shortest paths from a source node to all other nodes in the same asymptotic time as needed to compute a shortest path from a source to one specified node.

Below, we show that this algorithm actually finds a flow that is close to optimal by comparing the scaled primal solution to the dual solution that is a byproduct of the algorithm. We choose $\delta$ so that this ratio is at least $(1 - O(\epsilon))$.

THEOREM 2.4. *An $\epsilon$-approximate maximum multicommodity flow can be computed in $\mathcal{O}(\frac{1}{\epsilon^2} m(m + n\log m)\log n)$ time.*

*Proof.* We show that the scaled flow in Lemma 2.2 has value within $1/(1+4\epsilon)$ of the dual optimal value, and hence the optimal value for the primal. By choosing an initial value $\epsilon' = \epsilon/4$, this then implies the theorem. At the end of iteration $i$, $\alpha(i)$ is the exact shortest path over all commodities using length function $l_i$. Given length function $l$, define $D(l) := \sum_e l(e)u(e)$, and let $D(i) := D(l_i)$. $D(i)$ is the dual objective function value of $l_i$ and $\beta := \min_l D(l)/\alpha(l)$ is the optimal dual objective value. Let $g_i$ be the primal objective function value at the end of iteration $i$.

For each iteration $i \ge 1$,

$$D(i) = \sum_e l_i(e)u(e) = \sum l_{i-1}(e)u(e) + \epsilon \sum_{e \in P} l_{i-1}(e)u \\ \le D(i-1) + \epsilon(g_i - g_{i-1})(1+\epsilon)\alpha(i-1), \tag{2.1}$$

which implies that

$$D(i) \le D(0) + \epsilon(1+\epsilon)\sum_{j=1}^{i}(g_j - g_{j-1})\alpha(j-1). \tag{2.2}$$

Consider the length function $l_i - l_0$. Note that $D(l_i - l_0) = D(i) - D(0)$. For any path used by the algorithm, the length of the path using $l_i$ versus $l_i - l_0$ differs by at most $\delta n$. Since this holds for the shortest path using length function $l_i - l_0$, we have that $\alpha(l_i - l_0) \ge \alpha(i) - \delta n$. Hence

$$\beta \le \frac{D(l_i - l_0)}{\alpha(l_i - l_0)} \le \frac{D(i) - D(0)}{\alpha(i) - \delta n}.$$

Substituting this bound on $D(i) - D(0)$ in (2.2) gives

$$\alpha(i) \le \delta n + \frac{\epsilon(1+\epsilon)}{\beta}\sum_{j=1}^{i}(g_j - g_{j-1})\alpha(j-1).$$

Observe that, for fixed $i$, this right-hand side is maximized by setting $\alpha(j)$ to its maximum possible value $\forall\ 0 \le j < i$. Call this maximum value $\alpha'(i)$. Hence

$$\begin{aligned}\alpha(i) \le \alpha'(i) &= \alpha'(i-1)(1 + \epsilon(1+\epsilon)(g_i - g_{i-1})/\beta)\\ &\le \alpha'(i-1)e^{\epsilon(1+\epsilon)(g_i - g_{i-1})/\beta},\end{aligned}$$

where this last inequality uses the fact that $1 + a \le e^a$ for $a \ge 0$. Since $\alpha'(0) = \delta n$, this implies that

$$\alpha(i) \le \delta n e^{\epsilon(1+\epsilon)g_i/\beta}. \tag{2.3}$$

By the stopping condition, in the final iteration $t$, we have

$$1 \le \alpha(t) \le \delta n e^{\epsilon(1+\epsilon)g_t/\beta},$$

and hence

$$\frac{\beta}{g_t} \le \frac{\epsilon(1+\epsilon)}{\ln(\delta n)^{-1}}. \tag{2.4}$$

Let $\gamma$ be the ratio of the dual and primal solutions. $\gamma := \frac{\beta}{g_t}\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$. By substituting the bound on $\beta/g_t$ from (2.4), we obtain

$$\gamma \le \frac{\epsilon(1+\epsilon)\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}{\ln(n\delta)^{-1}} = \frac{\epsilon(1+\epsilon)}{\ln(1+\epsilon)}\frac{\ln\frac{1+\epsilon}{\delta}}{\ln(n\delta)^{-1}}. \tag{2.5}$$

Letting $\delta = (1+\epsilon)((1+\epsilon)n)^{-1/\epsilon}$, we have

$$\gamma \le \frac{\epsilon(1+\epsilon)}{(1-\epsilon)\ln(1+\epsilon)} \le \frac{\epsilon(1+\epsilon)}{(1-\epsilon)(\epsilon - \epsilon^2/2)} \le \frac{(1+\epsilon)}{(1-\epsilon)^2}.$$

This is at most $(1+4\epsilon)$ for $\epsilon < .15$. The choice of $\delta$ together with Lemma 2.1 imply the run time. $\square$

We have shown above that the framework of Garg and Könemann [9] can be modified to obtain a more efficient algorithm, by showing that on average, it is sufficient to solve the shortest path subproblem for a single commodity. We note that the same is not easily said of the algorithm described in [14]. This algorithm requires that the shortest path subproblem for each commodity be solved (albeit approximately) at each iteration.

This algorithm also finds $\epsilon$-approximate dual solutions. Above, we have replaced $D/\alpha$ in (2.2) with the exact optimal dual solution $\beta$, and we have shown that (2.4) holds for this value. However, we can replace $\beta$ throughout the proof with $\beta'$, the best (lowest) value of $D(i)/\alpha(i)$ encountered in the algorithm, and we get the same bound for $\frac{\beta'}{g_t}$ as we have shown for $\frac{\beta}{g_t}$. This value $\beta'$ corresponds to the dual solution $l_i/\alpha(i)$, which is feasible by definition of $\alpha$. Thus our proof of $\epsilon$-optimality for the primal solution really is a proof that the final scaled primal solution is within $1/(1+4\epsilon)$ of the best dual solution obtained during the algorithm. As noted earlier, instead of terminating the algorithm when $\min_{P\in\mathcal{P}_j} l(P) \geq 1$, we can terminate the algorithm once we have this proof.

Although it is simple to extend this algorithm to approximately solve the minimum cost maximum multicommodity flow problem, the technique for doing so does not differ from the technique of extending the maximum concurrent flow problem to the minimum cost concurrent flow problem. For this reason, we omit this discussion and refer the reader to the section on minimum cost concurrent flows.

**2.2. Some practical modifications.** We discuss briefly two aspects of the above analysis that could impact practical performance.

1. For large graphs and small $\epsilon$, our expression for $\delta$ may be too small to implement easily. This is avoidable by choosing a larger value of $\delta$ and modifying the termination criterion appropriately. Instead of showing that the algorithm will terminate by the time the length of the shortest path is at least 1, for any $\delta$, it follows from a simple modification to (2.5) that the algorithm will terminate by the time the length of the shortest path is at least $q(\delta) := \frac{\delta}{1+\epsilon}[n(1+\epsilon)]^{1/\epsilon}$. The number of augmentations remains $O(m\frac{1}{\epsilon^2}\log n)$. This also implies that if the lengths of edges become too large during the course of the algorithm, they can be scaled down without affecting the run time or accuracy of the algorithm.

2. Above we have described an algorithm that maintains a variable $l(e)$ for each edge and modifies $l(e)$ whenever the flow on edge $e$ is increased by $u \leq u(e)$ by multiplying it by $(1+\frac{u}{u(e)}\epsilon)$. This is not the only update that will result in the approximation guarantees described here. Another example that will work is to update $l(e)$ by multiplying by $e^{\epsilon u/u(e)}$. With this update, $l(e) = \delta e^{\epsilon x(e)/u(e)}$ at any point during the algorithm. Thus, scaling by $\log_{e^\epsilon}\frac{\max|l(e)|}{\delta}$ yields a feasible primal solution at any point in the algorithm, and the total number of augmentations is at most $O(m\log_{e^\epsilon}(q(e)e^\epsilon/\delta))$. To get an approximation guarantee, note that $e^a \leq 1+a+a^2$ for $0 \leq a \leq 1$. Then, the increase in $u(e)l(e)$ in one iteration is bounded from above by $\epsilon u + \epsilon^2 u\frac{u}{u(e)} \leq \epsilon(1+\epsilon)u$ as long as $u \leq u(e)$. Substituting this in (2.1) turns it into an inequality, and results in a modification of inequality (2.3) to $\alpha(i) \leq \delta n e^{\epsilon(1+\epsilon)^2 g_i/\beta}$. This then implies that for an appropriate termination point $\gamma \leq (\frac{1+\epsilon}{1-\epsilon})^2$, which is at most $1+16\epsilon$ for small enough $\epsilon$.

In fact, comparing this analysis with the second order Taylor series expansion reveals that we may use as the update factor $\phi(u)$ any convex, increasing function on $[0, u(e)]$ that satisfies $\phi(0) = 1$, $\phi'(0) = \epsilon/u(e)$, and $\phi''(0) \geq 0$. The theory shows that the number of iterations may depend on $\phi''(0)$. Experiments comparing $1+\epsilon\frac{u}{u(e)}$ with $e^{\epsilon\frac{u}{u(e)}}$ indicate that the latter update performs better [1].

**2.3. Maximum weighted multicommodity flow.** The maximum multicommodity flow approximation scheme can be extended to handle weights in the objective function. Let $w_j$ be the weight of commodity $j$. We wish to find flows $f_j$ maximizing $\sum_j w_j|f_j|$. By scaling, we assume $\min_j w_j = 1$, and then $W := \max_j w_j$ is the maximum ratio of weights of any two commodities. The resulting change in the dual problem is that the path constraints are now of the form $\sum_{e\in P\subset\mathcal{P}_j} l(e) \geq w_j \; \forall P \in \cup_j \mathcal{P}_j$.

COROLLARY 2.5. *A solution of value within* $(1-\epsilon)$ *of the optimal solution for the weighted maximum multicommodity flow problem can be computed in* $\mathcal{O}^*(\epsilon^{-2}m^2\min\{\log W, k\})$ *time.*

*Proof.* We alter the FPTAS for maximum multicommodity flow by defining $\overline{\alpha}(i) := \min_{1\leq j\leq k}\min_{P\in\mathcal{P}_j}\frac{l(P)}{w_j}$ and substituting this for $\alpha$. As before, the algorithm terminates when $\overline{\alpha}\geq 1$. For this modified algorithm, we choose $\delta = (1+\epsilon)W/((1+\epsilon)nW)^{1/\epsilon}$. The rest of the algorithm remains unchanged. The analysis of the algorithm changes slightly, now that we are using $\overline{\alpha}$. The initial value of $\overline{\alpha}$ can be as low as $\delta/W$, and the final value of $\overline{\alpha}$ is at most $(1+\epsilon)$. Hence, the number of times $\hat{\alpha}$ can increase by a factor of $(1+\epsilon)$ is $\log_{1+\epsilon}\frac{(1+\epsilon)W}{\delta}$. The length of an edge starts at $\delta$ and can be as large as $(1+\epsilon)W$, so the number of iterations is at most $m\log_{1+\epsilon}\frac{(1+\epsilon)W}{\delta}$, and the scale factor to make the final solution primal feasible is at most $\log_{1+\epsilon}\frac{(1+\epsilon)W}{\delta}$. Choosing $\delta = (1+\epsilon)W/((1+\epsilon)nW)^{1/\epsilon}$ implies a run time of $\mathcal{O}^*(\epsilon^{-2}m^2\log W)$.

To establish that this modified algorithm finds an $\epsilon$-approximate solution, we follow the analysis in the proof of Theorem 2.4 with the following modifications: let $h_i$ be the value of the primal objective function at the end of iteration $i$, and let $D(i)$ be the dual objective function value, as before. For iteration $i\geq 1$ we have that

$$\begin{aligned} D(i) = \sum_e l_i(e)u(e) &= \sum l_{i-1}(e)u(e) + \epsilon\sum_{e\in P} l_{i-1}(e)u \\ &\leq D(i-1) + \epsilon(1+\epsilon)w_j\overline{\alpha}(i-1)\frac{h_i - h_{i-1}}{w_j} \\ &\leq D(i-1) + \epsilon(1+\epsilon)\overline{\alpha}(i-1)(h_i - h_{i-1}). \end{aligned}$$

Symmetric to the case without weights, a dual solution $l_i$ is made dual feasible by dividing by $\overline{\alpha}(l_i)$. Since the bounds on $D(l_i - l_0)$ and $\overline{\alpha}(l_i - l_0)$ remain valid, the same analysis goes through to obtain the following bound on $\gamma$, the ratio of primal and dual solutions obtained by the algorithm

$$\gamma \leq \frac{\epsilon(1+\epsilon)}{\ln(1+\epsilon)}\frac{\ln\frac{(1+\epsilon)W}{\delta}}{\ln(n\delta)^{-1}}.$$

With the above choice of $\delta$, this is at most $(1+4\epsilon)$ for $\epsilon < .15$.

The strongly polynomial run time follows directly from the analysis of the packing LP algorithm in [9]. □

**3. Maximum concurrent flow.** Recall the maximum concurrent flow problem has specified demands $d_j$ for each commodity $1\leq j\leq k$ and the problem is to find a flow that maximizes the ratio of met demands.

As with the maximum multicommodity flow, we let $x(P)$ denote the flow quantity on path $P$. The maximum concurrent flow problem can be formulated as the following linear program.

$$\begin{array}{lllll} \max & \lambda & & \\ \forall e: & \sum\limits_{P: e \in P} x(P) & \leq & u(e) \\ \forall j: & \sum\limits_{P \in \mathcal{P}_j} x(P) & \geq & \lambda d_j \\ \forall P: & x(P) & \geq & 0. \end{array}$$

The dual LP problem is to assign lengths to the edges of the graph and weights $z_j$ to the commodities so that the length of the shortest path from $s_j$ to $t_j$ is at least $z_j$ for all commodities $j$ and the sum of the product of commodity weights and demands is at least 1. The length of an edge represents the marginal cost of using an additional unit of capacity of the edge, and the weight of a commodity represents the marginal cost of not satisfying another unit of demand of the commodity.

$$\begin{array}{llll} \min & \sum_e u(e) l(e) & & \\ \forall j,\ \forall P \in \mathcal{P}_j: & \sum\limits_{e \in P} l(e) & \geq & z_j \\ & \sum\limits_{1 \leq j \leq k} d_j z_j & \geq & 1 \\ \forall e: & l(e) & \geq & 0 \\ \forall j: & z_j & \geq & 0. \end{array}$$

When the objective function value is $\geq 1$, Garg and Könemann [9] describe an $\mathcal{O}^*(\epsilon^{-2} m(m+k))$ time approximation scheme for the maximum concurrent flow problem. If the objective is less than one, then they describe a procedure to scale the problem so that the objective is at least one. This procedure requires computing $k$ maximum flows, which increases the run time of their algorithm so that it matches previously known algorithms. We describe a different procedure that requires $k$ maximum bottleneck path computations which can each be performed in $\mathcal{O}(m \log m)$ time. Since the total time spent on this new procedure no longer dominates the time to solve the scaled problem with objective function value $\geq 1$, the resulting algorithm solves the maximum concurrent flow problem in $\mathcal{O}^*(\epsilon^{-2} m(m+k))$ time.

For the cost bounded problem, Garg and Könemann [9] use a minimum cost flow subroutine. We use a cost bounded maximum bottleneck path subroutine, which can be solved in $\mathcal{O}(m \log m)$ time. Thus our procedure improves the run time for the minimum cost concurrent flow problem as well.

We first describe the approximation algorithm in [9]. Initially, $l(e) = \delta / u(e)$, $z_j = \min_{P \in \mathcal{P}_j} l(P)$, $x \equiv 0$. The algorithm proceeds in phases. In each phase, there are $k$ iterations. In iteration $j$, the objective is to route $d_j$ units of flow from $s_j$ to $t_j$. This is done in steps. In one step, a shortest path $P$ from $s_j$ to $t_j$ is computed using the current length function. Let $u$ be the bottleneck capacity of this path. Then the minimum of $u$ and the remaining demand is sent along this path. The dual variables $l$ are updated as before, and $z_j$ is set equal to the length of the new minimum length path from $s_j$ to $t_j$. The entire procedure stops when the dual objective function value is at least one: $D(l) := \sum_e u(e) l(e) \geq 1$. See Figure 3.1 for a summary of the algorithm. Garg and Könemann [9] prove the following sequence of lemmas for $\delta = (\frac{m}{1-\epsilon})^{-1/\epsilon}$. Here, $\beta$ is the optimal objective function value.

LEMMA 3.1. *If $\beta \geq 1$, the algorithm terminates after at most $t := 1 + \frac{\beta}{\epsilon} \log_{1+\epsilon} \frac{m}{1+\epsilon}$ phases.*

```
Input: network $G$, capacities $u(e)$, vertex pairs $(s_i, t_i)$
        with demands $d_i$, $1 \le i \le k$, accuracy $\epsilon$
Output: primal (infeasible) and dual solutions $x$ and $l$

    Initialize $l(e) = \delta/u(e)\ \forall e,\ x \equiv 0$.
    while $D(l) < 1$
        for $j = 1$ to $k$ do
            $d'_j \leftarrow d_j$
            while $D(l) < 1$ and $d'_j > 0$
                $P \leftarrow$ shortest path in $\mathcal{P}_j$ using $l$
                $u \leftarrow \min\{d'_j, \min_{e \in P} u(e)\}$
                $d'_j \leftarrow d'_j - u$
                $x(P) \leftarrow x(P) + u$
                $\forall e \in P,\ l(e) \leftarrow l(e)(1 + \frac{\epsilon u}{u(e)})$
            end while
    end while
    Return $(x, l)$.
```

FIG. 3.1. *FPTAS for max concurrent flow.*

LEMMA 3.2. *After $t-1$ phases, $(t-1)d_j$ units of each commodity $j$ have been routed. Scaling the final flow by $\log_{1+\epsilon} 1/\delta$ yields a feasible primal solution of value $\lambda = \frac{t-1}{\log_{1+\epsilon} 1/\delta}$.*

LEMMA 3.3. *If $\beta \ge 1$, then the final flow scaled by $\log_{1+\epsilon} 1/\delta$ has a value at least $(1-3\epsilon)$ OPT.*

The veracity of these lemmas relies on $\beta \ge 1$. Notice also that the run time depends on $\beta$. Thus we need to insure that $\beta$ is at least one and not too large.

Let $\zeta_j$ denote the maximum flow value of commodity $j$ in the graph when all other commodities have zero flow. Let $\zeta = \min_j \zeta_j/d_j$. Since at best all single commodity maximum flows can be routed simultaneously, $\zeta$ is an upper bound on the value of the optimal solution. The feasible solution that routes $1/k$ fraction of each commodity flow of value $\zeta_j$ demonstrates that $\zeta/k$ is a lower bound on the optimal solution. Once we have these bounds, we can scale the original demands so that this lower bound is at least one. However, now $\beta$ can be as large as $k$.

To reduce the dependence of the number of phases on $\beta$, we use a popular technique developed in [22] that is also used in [9]. We run the algorithm, and if it does not stop after $T := 2\frac{1}{\epsilon}\log_{1+\epsilon}\frac{m}{1-\epsilon}$ phases, then $\beta > 2$. We then multiply demands by 2, so that $\beta$ is halved and still at least 1. We continue the algorithm, and again double demands if it does not stop after $T$ phases. After repeating this at most $\log k$ times, the algorithm stops. The total number of phases is $T \log k$. The number of phases can be further reduced by first computing a solution of cost at most twice the optimal using this scheme. This takes $\mathcal{O}(\log k \log m)$ phases, and returns a value $\hat{\beta}$ such that $\beta \le \hat{\beta} \le 2\beta$. Thus with at most $T$ additional phases, an $\epsilon$-approximation is obtained. The following lemma follows from the fact that there are at most $k$ iterations per phase.

LEMMA 3.4. *The total number of iterations required by the algorithm is at most $2k \log m(\log k + \frac{1}{\epsilon^2})$.*

It remains to bound the number of steps. For each step except the last step in an iteration, the algorithm increases the length of some edge (the bottleneck edge on $P$) by $1+\epsilon$. Since each variable $l(e)$ has initial value $\delta/u(e)$ and value at most $\frac{1}{u(e)}$ before the final step of the algorithm (since $D(t-1) < 1$), the number of steps in the entire algorithm exceeds the number of iterations by at most $m \log_{1+\epsilon} \frac{1}{\delta} = m \log_{1+\epsilon} \frac{m}{1-\epsilon}$.

THEOREM 3.5. *Given $\zeta_j$, an $\epsilon$-approximate solution to the maximum concurrent flow problem can be obtained in $\mathcal{O}^*(\epsilon^{-2}m(k+m))$ time.*

Our contribution is to reduce the dependence of the run time of the algorithm on the computation of $\zeta_j$, $1 \leq j \leq k$, by observing that it is not necessary to obtain the exact values of $\zeta_j$ since they are just used to get an estimate on $\beta$. Computing an estimate of $\zeta_j$ that is at most a factor $m$ away from its true value increases the run time by only a $\log m$ factor. That is, if our estimate $\hat{\zeta_j}$ is $\geq \frac{1}{m}\zeta_j$, then we have upper and lower bounds on $\beta$ that differ by a factor of at most $mk$.

We compute estimates $\hat{\zeta_j} \geq \frac{1}{m}\zeta_j$ as follows. Any flow can be decomposed into at most $m$ path flows. Hence, flow sent along a maximum capacity path is an $m$-approximation to a maximum flow. Such a path can be computed easily in $\mathcal{O}(m \log m)$ time by binary searching for the capacity of this path. In fact, by grouping commodities by their common source node, we can compute all such paths in $\mathcal{O}(\min\{k, n\} m \log m)$ time.

THEOREM 3.6. *An $\epsilon$-approximate solution to the maximum concurrent flow problem can be obtained in $\mathcal{O}^*(\epsilon^{-2}m(k+m))$ time.*

## 4. Minimum cost concurrent flow.

By adding a budget constraint to the multiple commodity problem, it is possible to find a concurrent flow within $(1-\epsilon)$ of the maximum concurrent flow within the given budget. Since there is a different variable for each commodity-path pair, this budget constraint can easily incorporate different costs for different commodities.

In the dual problem, let $\phi$ be the dual variable corresponding to the budget constraint. In the algorithm, the initial value of $\phi$ is $\delta/B$. The subroutine to find a most violated primal constraint is a shortest path problem using length function $l + \phi c$, where $c$ is the vector of cost coefficients and $\phi$ is the dual variable for the budget constraint. (This can be easily adapted to multiple budget constraints with corresponding dual variables $\phi_i$ and cost vectors $c_i$ using length function $l + \sum_i \phi_i c_i$.) The amount of flow sent on this path is now determined by the minimum of the capacity of this path, the remaining demand to be sent of this commodity in this iteration, and $B/c(P)$, where $c(P) = \sum_{e \in P} c(e)$ is the cost of sending one unit of flow along this path. Call this quantity $u$. The length function is updated as before, and $\phi$ is updated by $\phi = \phi(1 + \epsilon \frac{uc(P)}{B})$. The correctness of this algorithm is demonstrated in [9] and follows a similar analysis as for the maximum concurrent flow algorithm.

Again, we improve on the run time in [9] by efficient approximation of the values $\zeta_j$. To get estimates on $\zeta_j$, as needed to delimit $\beta$, we compute $m$-approximate solutions to $\min\{n^2, k\}$ maximum-flow-with-budget problems. To do this, it is sufficient to compute a maximum bottleneck path of cost at most the budget. This can be done by binary searching for the capacity of this path among the $m$ candidate capacities, and performing a shortest path computation at each search step, for a run time of $\mathcal{O}(m \log m)$ per commodity.

To find a $(1-\epsilon)$ maximum concurrent flow of cost no more than the minimum cost maximum concurrent flow, we can then binary search for the correct budget.

THEOREM 4.1. *There exists an FPTAS for the minimum cost concurrent flow problem that requires $\mathcal{O}^*(\epsilon^{-2}m(m+k)\log M)$ time.*

**Acknowledgments.** We thank Cynthia Phillips for suggesting a simpler presentation of the maximum multicommodity flow algorithm and Kevin Wayne, Jeffrey Oldham, Aravind Srinivasan, and an anonymous referee for helpful comments.

## REFERENCES

[1] C. Albrecht, *Provably good global routing by a new approximation algorithm for multicommodity flow*, in Proceedings of the International Conference on Physical Design (ISPD), San Diego, CA, ACM, New York, 2000, pp. 19–25.

[2] Y. Aumann and Y. Rabani, *An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm*, SIAM J. Comput., 27 (1998), pp. 291–301.

[3] D. Bienstock, *Experiments with a Network Design Algorithm Using $\epsilon$-Approximate Linear Programs*, CORC Report 1999-4, Department of Industrial Engineering and Operations Research, Columbia University.

[4] D. Bienstock, *Approximately Solving Large-Scale Linear Programs* I: *Strengthening Lower Bounds and Accelerating Convergence*, CORC Report 1999-1, Department of Industrial Engineering and Operations Research, Columbia University, CORE Lecture Series 2000, to appear.

[5] F. Dragan, A. B. Kahng, S. Muddu, I. Mandoiu, and A. Zelikovsky, *Global routing with given buffer blocks*, in Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, 2000, to appear.

[6] L. K. Fleischer, *Approximating fractional multicommodity flows independent of the number of commodities*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, IEEE, New York, 1999, pp. 24–31.

[7] L. Fratta, M. Gerla, and L. Kleinrock, *The flow deviation method: an approach to store-and-forward communication network design*, Networks, 3 (1973), pp. 97–133.

[8] N. Garg, *personal communication*, 1999.

[9] N. Garg and J. Könemann, *Faster and simpler algorithms for multicommodity flow and other fractional packing problems*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, IEEE, New York, 1998, pp. 300–309.

[10] N. Garg, V. V. Vazirani, and M. Yannakakis, *Approximate max-flow min-(multi)cut theorems and their applications*, SIAM J. Comput., 25 (1996), pp. 235–251.

[11] A. V. Goldberg, J. D. Oldham, S. Plotkin, and C. Stein, *An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow*, in Integer Programming and Combinatorial Optimization, R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, eds., Lecture Notes in Comput. Sci. 1412, Springer, Berlin, pp. 338–352.

[12] M. D. Grigoriadis and L. G. Khachiyan, *Fast approximation schemes for convex programs with many blocks and coupling constraints*, SIAM J. Optim, 4 (1994), pp. 86–107.

[13] M. D. Grigoriadis and L. G. Khachiyan, *Approximate minimum-cost multicommodity flows*, Mathematical Programming, 75 (1996), pp. 477–482.

[14] M. D. Grigoriadis and L. G. Khachiyan, *Coordination complexity of parallel price-directive decomposition*, Math. Oper. Res., 21 (1996), pp. 321–340.

[15] M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica, 1 (1981), pp. 169–197.

[16] D. Karger and S. Plotkin, *Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows*, in Proceedings of the 27th Annual ACM Symposium on Theory of Computing, ACM, New York, 1995, pp. 18–25.

[17] P. Klein, S. Plotkin, C. Stein, and É. Tardos, *Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts*, SIAM J. Comput., 23 (1994), pp. 466–487.

[18] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas, *Fast approximation algorithms for multicommodity flow problems*, J. Comput. System Sci., 50 (1995), pp. 228–243.

[19] T. Leighton and S. Rao, *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*, J. ACM, 46 (1999), pp. 787–832. Extended abstract appeared in *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, New York, 1988, pp. 422–431.

[20] T. Leong, P. Shor, and C. Stein, *Implementation of a combinatorial multicommodity flow algorithm*, in The First DIMACS Implementation Challenge: Network Flows and Matchings, in DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 12, D. S. Johnson and C. McGoech, eds., AMS, Providence, RI, 1993, pp. 387–405.

[21] N. Linial, E. London, and Y. Rabinovich, *The geometry of graphs and some of its algorithmic applications*, Combinatorica, 15 (1995), pp. 215–246.

[22] S. A. Plotkin, D. Shmoys, and É. Tardos, *Fast approximation algorithms for fractional packing and covering problems*, Math. Oper. Res., 20 (1995), pp. 257–301.

[23] T. Radzik, *Fast deterministic approximation for the multicommodity flow problem*, in Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, 1995, ACM, New York, 1995, pp. 486–492.

[24] M. Sato, *Efficient Implementation of an Approximation Algorithm for Multicommodity Flows*, Master's thesis, Division of Systems Science, Graduate School of Engineering Science, Osaka University, Osaka, Japan, 2000.

[25] F. Shahrokhi and D. W. Matula, *The maximum concurrent flow problem*, J. ACM, 37 (1990), pp. 318–334.

[26] D. B. Shmoys, *Cut problems and their application to divide-and-conquer*, in Approximation Algorithms for NP-Hard Problems, D. S. Hochbaum, ed., PWS Publishing Company, Boston, 1997, chapter 5.

[27] N. Young, *Randomized rounding without solving the linear program*, in Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, 1995, ACM, New York, 1995, pp. 170–178.

SIAM J. DISCRETE MATH. © 2000 Society for Industrial and Applied Mathematics
Vol. 13, No. 4, pp. 521–534

# WELL-SPACED LABELINGS OF POINTS IN RECTANGULAR GRIDS*

JEFFREY C. LAGARIAS†

**Abstract.** We describe methods to label the $M_1 \times M_2$ grid with the integers 1 to $M_1M_2$ so that any $K$ consecutively labeled cells are relatively far apart in the grid in the Manhattan metric. Constructions of such labelings are given which are nearly optimal in a range of conditions. Such labelings can be used in addressing schemes for storing data on two-dimensional arrays that include randomly located "blobs" of defective cells. The data can be precoded using block error-correcting codes before storage, and the usefulness of well-spaced points is to decrease the probability of "burst" errors which cannot be corrected. Possible applications include the storage of speech or music on low-quality memory chips and in "holographic memories" to store bit-mapped data.

More generally, we present a general family of mappings of the integers 1 to $M_1M_2\cdots M_d$ onto the $d$-dimensional grid of size $M_1 \times M_2 \times \cdots \times M_d$, called mixed radix vector mappings. These mappings give labelings whenever they are one to one. We give a sufficient condition for these mappings to be one to one, which is easy to verify in many cases.

**Key words.** radix expansions, error correction, packings

**AMS subject classifications.** Primary, 90C27; Secondary, 05B40, 11A63, 11H71, 52C20, 68R05, 94B60

**PII.** S0895480199362241

## 1. Introduction.

Let $[N_1, N_2]$ denote the set of consecutive integers $\{N_1, N_1 + 1, \ldots, N_2\}$. The *d-dimensional rectangular grid* $G(M_1, M_2, \ldots, M_d)$ is the set of lattice points

$$\begin{aligned} G(M_1, \ldots, M_d) &= [0, M_1 - 1] \times [0, M_2 - 1] \times \cdots \times [0, M_d - 1] \\ &= \{(m_1, \ldots, m_d) \in \mathbb{Z}^d : 0 \le m_j \le M_j - 1\}. \end{aligned} \tag{1.1}$$

A *labeling* of $G(M_1, M_2, \ldots, M_d)$ is a one-to-one mapping

$$\phi : \left[1, \prod_{j=1}^{d} M_j\right] \to G(M_1, M_2, \ldots, M_d). \tag{1.2}$$

A labeling $\phi$ is clearly onto, hence it has a well-defined inverse $\phi^{-1}$. We think of a labeling as giving an access ordering to the $\prod_{j=1}^{d} M_j$ cells in the grid $G(M_1, M_2, \ldots, M_d)$. We consider the general problem of finding labelings with the property that points with labels that are close are spaced far apart in the grid. We measure distance in the grid using the $L^1$-*norm*,

$$||(m_1, \ldots, m_d) - (m_1', \ldots, m_d')|| := \sum_{i=1}^{d} |m_i - m_i'|. \tag{1.3}$$

In the two-dimensional case there are a number of applications for such well-spaced labelings, which motivated this work. One application consists of schemes to

*Received by the editors October 5, 1999; accepted for publication (in revised form) July 5, 2000; published electronically October 25, 2000.
http://www.siam.org/journals/sidma/13-4/36224.html

†AT&T Labs-Research, Florham Park, NJ 07932-0971 (jcl@research.att.com).

address memory locations on chips which have memory arrays containing randomly distributed defects consisting of variable-sized "blobs" of defective memory cells. Such labelings are useful in minimizing the probability of consecutively accessed locations being defective. The data can be preencoded using a block error-correcting code before storing it in such an array, and the usefulness of well-spaced points is to decrease the probability of "burst" errors, which block coding cannot handle. A second application concerns the design of proposed "holographic memories" to store bit-mapped data, in situations where scanning errors introduce "blocks" of blank data; cf. Bruckstein, Holt, and Netravali [3]. In both applications one wants labeling schemes $\phi$ that are simple to calculate and to invert.

In section 2 we give a general construction of mappings $\phi : [1, \prod_{j=1}^{d} M_j] \to G(M_1, \ldots, M_d)$, *called mixed radix vector mappings*, which are mappings

$$\phi(n) = \sum_{j=1}^{r} k_j \mathbf{y}_j, \tag{1.4}$$

in which the vectors $\mathbf{y}_1, \ldots, \mathbf{y}_r \in \mathbb{Z}^d$ are a given set of $r$ vectors, together with a mapping

$$n \longmapsto (k_1, \ldots, k_r), \tag{1.5}$$

which is the mixed radix expansion of $n$ to base $\mathbf{B} = (B_1, B_2, \ldots, B_r)$, as defined in section 2. Such mappings are not necessarily one to one. We give a sufficient condition for such a map to be one to one, which applies in many situations (Theorem 2.1). Suitable choices of the parameters $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_r\}$ and $(B_1, B_2, \ldots, B_r)$ yield well-spaced labelings, particularly in the two-dimensional case.

In this paper we study the two-dimensional case in sections 3 and 4. In section 3 we consider the problem of constructing labelings that have every two consecutively labeled points far apart. We show that every admissible labeling has two consecutively labeled points at $L^1$-distance at most $\lfloor \frac{M_1+M_2}{2} \rfloor$. We find mixed radix vector labelings that achieve separation of consecutive labels by $\lfloor \frac{M_1+M_2}{2} \rfloor - 2$ (Theorem 3.1).

In section 4 we consider the problem of constructing labelings that well-separate $k$ consecutive labels for $k \geq 3$. We prove upper bounds for the attainable separation of $\frac{3M_2}{k}$ if $M_2 \geq \frac{k}{3} M_1$, and $\left(\frac{8M_1M_2}{k}\right)^{1/2}$ if $M_1 \leq M_2 \leq \frac{k}{3} M_1$ (Theorem 4.1). These bounds are shown to be attainable within a multiplicative factor of 4 by mixed radix labelings for many values of $M_1, M_2$, and $k$ (Theorem 4.2).

This paper was motivated by the two-dimensional case and does not consider labelings in dimension $d \geq 3$, except for the criterion of section 2 for mixed radix vector mappings to be one to one on $d$-dimensional grids. We leave as open problems obtaining good upper bounds on the maximum attainable $L^1$-distance between members of a labeling at distance at most $m$, and determining whether suitable mixed radix vector mappings give nearly optimal well-spaced labelings of points for various $d$-dimensional rectangular grids when $d \geq 3$. For the upper bound we note a connection with a generalization of the unsolved problem D1, "spreading points in a square," in Croft, Falconer, and Guy [4, pp. 108–110]. The problem of packing $k$ points in an $d$-dimensional cube, so as to maximize the minimal distance between any two of the points (using $L^1$-norm), certainly gives an upper bound (after rescaling) on how much separation is possible in every consecutive $k$-tuple of a labeling of a $d$-dimensional cubical grid. The results of this paper suggest that for $d$-dimensional grids which are roughly cubical there exist well-spaced labelings with spacing within a multiplicative constant (depending on the dimension $d$) of such an upper bound.

We also note a relation to *magic squares*, and more generally *magic $d$-cubes*. These are a special kind of labeling of the $d$-cube $G(M, M, \dots, M)$ by integers in which all the row sums are required to be equal; cf. [1], [2], [5], [6]. One of the standard constructions of magic squares and magic $d$-cubes using the integers from 1 to $M^d$ is actually a mixed radix vector labeling; see [1], [2].

**2. Mixed radix vector mappings.** We present a general construction of $d$-dimensional mappings

$$\phi : \left[1, \prod_{j=1}^{d} M_j\right] \to G(M_1, M_2, \dots, M_d),$$

which we call mixed radix vector (MRV) mappings. Whenever such a mapping $\phi$ is one to one, then it gives an admissible labeling of

$$G(M_1, M_2, \dots, M_d) = [0, M_1 - 1] \times [0, M_2 - 1] \times \cdots \times [0, M_d - 1].$$

A mixed radix expansion is specified by an $r$-dimensional vector $\mathbf{B} := (B_1, B_2, \dots, B_r)$ of positive integers, with all $B_i > 1$, called the *mixed base*. The *mixed radix expansion to base* $\mathbf{B}$ of an integer $m$ satisfying $0 \le m < \prod_{i=1}^{r} B_i$ is

$$m = \sum_{i=1}^{r} d_i \prod_{j=1}^{i-1} B_j, \text{ with } 0 \le d_i < B_i \quad \text{for} \quad 1 \le i \le r, \tag{2.1}$$

To apply this to the grid $G(M_1, \dots, M_d)$ we need

$$\prod_{i=1}^{r} B_i \ge \prod_{j=1}^{d} M_j. \tag{2.2}$$

The general construction is as follows. An *MRV mapping* of $G(M_1, \dots, M_d)$ is specified by a set of $r \ge 1$ nonzero vectors $\mathbf{y}_1, \dots, \mathbf{y}_r \in \mathbb{Z}^d$ and by a mixed radix base

$$\mathbf{B} = (B_1, B_2, \dots, B_r),$$

such that $\prod_{i=1}^{r} B_i \ge \prod_{j=1}^{d} M_j$. For $1 \le m \le \prod_{j=1}^{d} M_j$ take the mixed radix expansion

$$m - 1 = d_r \prod_{i=1}^{r} B_i + d_{r-1} \prod_{i=1}^{r-1} B_i + \cdots + d_2 B_1 + d_1$$

with $0 \le d_i < B_i$, and associate to it the integer vector

$$\mathbf{v}(m) := \sum_{i=1}^{r} d_i \mathbf{y}_i = (v_1(m), v_2(m), \dots, v_d(m)). \tag{2.3}$$

The MRV map $\phi : [1, \prod_{j=1}^{d} M_j] \to G(M_1, M_2, \dots, M_d)$ is given by

$$\phi(m) = (v_1(m) (\text{mod } M_1), \dots, v_d(m) (\text{mod } M_d)). \tag{2.4}$$

The integer $r$ is called the *rank* of the MRV mapping. It bears no relation to the dimension $d$ of the image space and can be smaller or larger than $d$.

We are interested in the special case of MRV mappings $\phi$ that are one to one. We call these *MRV labelings*. There does not seem to be a simple general criterion for an MRV mapping to be one to one, but we give a sufficient condition in Theorem 2.1 below.

To formulate this sufficient condition we need a definition. Given a set $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_r\}$ of vectors in $\mathbb{Z}^d$, and a grid $G = G(M_1, \ldots, M_d)$, and given $\mathbf{m} = (m_1, \ldots, m_r) \in \mathbb{Z}^r$, set

$$\mathbf{w}(\mathbf{m}) = (w_1, w_2, \ldots, w_m) := m_1\mathbf{y}_1 + \cdots + m_r\mathbf{y}_r. \tag{2.5}$$

We associate to $\mathcal{Y}$ and the grid $G$ the $r$-dimensional integer lattice

$$\Lambda := \Lambda_{\mathcal{Y},G} = \{\mathbf{m} \in \mathbb{Z}^r : w_j \equiv 0 \pmod{M_j} \quad \text{for} \quad 1 \le j \le d\}. \tag{2.6}$$

We call $\Lambda$ the *embedding lattice* of $(\mathcal{Y}, G)$, since it describes how the vectors $\mathcal{Y}$ fall into $G$. The embedding lattice satisfies

$$(M_1 M_2 \cdots M_d \mathbb{Z})^d \subseteq \Lambda \subseteq \mathbb{Z}^d, \tag{2.7}$$

and the index $[\mathbb{Z}^d : \Lambda] := \#(\mathbb{Z}^d/\Lambda)$ always divides $\prod_{j=1}^d M_j$. We say that $\mathcal{Y}$ is *nondegenerate* for the grid $G(M_1, \ldots, M_d)$ if $[\mathbb{Z}^d : \Lambda] = \prod_{j=1}^d M_j$.

THEOREM 2.1. *Suppose that an MRV map $\phi$ of $G(M_1, M_2, \ldots, M_d)$ is specified by the set of vectors $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_r\}$ and the mixed radix base $\mathbf{B} = (B_1, B_2, \ldots, B_r)$.*

(i) *If each nonzero vector $\mathbf{m} = (m_1, m_2, \ldots, m_r)$ in the embedding lattice $\Lambda_{\mathcal{Y},G}$ satisfies*

$$\textit{there is some coordinate } i \textit{ with } \quad |m_i| \ge B_i, \tag{2.8}$$

*then the map $\phi$ is one to one.*

(ii) *If $\Pi_{i=1}^r B_i = \Pi_{j=1}^d M_j$, then* (2.8) *is a necessary and sufficient condition for $\phi$ to be one to one.*

*Proof of Theorem* 2.1. (i). We argue by contradiction. Suppose that $\phi$ is not one to one, so that there exist $1 \le m_1 < m_2 \le M_1 M_2 \cdots M_d$ such that $\phi(m_1) = \phi(m_2)$. Let the mixed radix expansions be

$$m_\ell - 1 = \sum_{i=1}^r d_i^{(\ell)} \prod_{k=1}^{i-1} B_k \text{ for } \ \ell = 1, 2, \tag{2.9}$$

and set $\mathbf{d}^{(\ell)} \equiv (d_1^{(\ell)}, \ldots, d_r^{(\ell)})$ for $\ell = 1, 2$. Now $\phi(m_1) = \phi(m_2)$ means that the vector $\mathbf{w} = (w_1, \ldots, w_d)$ given by

$$\mathbf{w} = \sum_{i=1}^r (d_i^{(1)} - d_i^{(2)})\mathbf{y}_i$$

satisfies $w_j \equiv 0 \pmod{M_j}$ for $1 \le j \le d$. Thus $\mathbf{d}^{(1)} - \mathbf{d}^{(2)} \in \Lambda_{\mathcal{Y},G}$. However the mixed radix expansion property gives

$$-B_i < d_i^{(1)} - d_i^{(2)} < B_i, \quad 1 \le i \le r,$$

because $0 \le d_i^{(\ell)} < B_i$. Since $\mathbf{d}^{(1)} \neq \mathbf{d}^{(2)}$, this contradicts property (2.8).

(ii). We must show that if $\prod_{i=1}^{r} B_i = \prod_{j=1}^{d} M_j$, then any one-to-one MRV labeling map $\phi$ satisfies condition (2.8). Indeed $\{\phi(m) : 1 \le m \le M_1 M_2 \cdots M_d\}$ enumerates the set of $M_1 \cdots M_d = B_1 \cdots B_r$ vectors $\mathbf{w}$ (mod $\mathbf{M}$), given by

$$\mathbf{w} = d_1\mathbf{v}_1 + d_2\mathbf{v}_2 + \cdots + d_r\mathbf{v}_r, \quad \text{with} \quad 0 \le d_i < B_i,\ 1 \le i \le r, \tag{2.10}$$

and

$$\mathbf{w} \pmod{\mathbf{M}} := (w_1 \pmod{M_1},\ w_2 \pmod{M_2}, \ldots, w_d (\text{mod } M_d)).$$

By hypothesis these residues are all distinct, and this implies that $[\mathbb{Z}^r : \Lambda_{\mathcal{Y},G}] \ge M_1 M_2 \cdots M_d$. Thus

$$[\mathbb{Z}^r : \Lambda] = M_1 M_2 \cdots M_d, \tag{2.11}$$

and $\mathcal{Y}$ is nondegenerate. It follows that

$$\mathcal{W} = \{\mathbf{w} : \mathbf{w} \ \text{ given by (2.10)}\}$$

is a fundamental domain for the quotient lattice $\mathbb{Z}^r/\Lambda_{\mathcal{Y},G}$. In particular, given any nonzero vector

$$\mathbf{e} = (e_1, \ldots, e_r) \in \mathbb{Z}^r \ \text{ with } \ -B_i < e_i < B_i \ \text{ for } \quad 1 \le i \le r, \tag{2.12}$$

set

$$\mathbf{w}' = e_1\mathbf{v}_1 + \cdots + e_r\mathbf{v}_r.$$

Then we can find two distinct vectors $\mathbf{w}_1, \mathbf{w}_2$ of the form (2.10) such that

$$\mathbf{w}' = \mathbf{w}_1 - \mathbf{w}_2.$$

Since the $\mathbf{w}_i$ are all distinct ( mod $M$) we conclude that $\mathbf{w}' \not\equiv \mathbf{0}$ ( mod $\mathbf{M}$), and hence $\mathbf{e} \notin \Lambda_{\mathcal{Y},G}$. Thus the embedding lattice $\Lambda_{\mathcal{Y},G}$ contains no vector of the form (2.12) except the zero vector, which verifies (2.8). □

As an example, consider the rectangular grid $G(6,6)$, and take the MRV mapping given by $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4\}$ with

$$\mathbf{y}_1 = (3,0),\ \mathbf{y}_2 = (0,3),\ \mathbf{y}_3 = (1,0), \ \text{ and } \ \mathbf{y}_4 = (0,1)$$

and with radix vector $\mathbf{B} = (2,2,3,3)$. We have $r = 4$ and $\Pi_{i=1}^{4} B_i = \Pi_{i=1}^{2} M_i = 36$. The mixed radix expansion is

$$m - 1 = 12d_4 + 4d_3 + 2d_2 + d_1, \quad 0 \le d_1 < B_i,$$

and

$$\phi(m) = d_1\mathbf{v}_1 + d_2\mathbf{v}_2 + d_3\mathbf{v}_3 + d_4\mathbf{v}_4.$$

The embedding lattice $\Lambda$ in $\mathbb{Z}^4$ is

$$\Lambda = (2\mathbb{Z} \times 2\mathbb{Z} \times 6\mathbb{Z} \times 6\mathbb{Z}) + \{(0,0,0,0),\ (1,0,3,0),\ (0,1,0,3),\ (1,1,3,3)\}.$$

This lattice satisfies the criterion (2.8); hence Theorem 2.1 shows that $\phi$ is one to one. The resulting labeling is pictured below, using matrix notation, so that rows are numbered downward from 0 to 5, and columns are numbered across, from 0 to 5.

$$\begin{bmatrix} 1 & 5 & 9 & 2 & 6 & 10 \\ 13 & 17 & 21 & 14 & 18 & 22 \\ 25 & 29 & 33 & 26 & 30 & 34 \\ 3 & 7 & 11 & 4 & 8 & 12 \\ 15 & 19 & 23 & 16 & 20 & 24 \\ 27 & 31 & 35 & 28 & 32 & 36 \end{bmatrix}$$

We end our discussion of the $d$-dimensional case by commenting on the problem of computing the inverse of the map $\phi$ when it is one to one. This map may be hard to invert in the general case. In the special case that $\Pi_{i=1}^{r} B_i = \Pi_{j=1}^{d} M_j$, the inversion problem reduces to solving a system of linear Diophantine equations, and for this there are polynomial-time algorithms. In the two-dimensional case it appears that there exist good MRV labelings of this sort having rank $r \leq 4$.

**3. Two-dimensional case: Separating consecutive points.** We consider the problem of labeling the $M_1M_2$ cells in an $M_1 \times M_2$ rectangular grid in such a way that consecutively labeled cells are far apart. The rectangular grid is

$$G(M_1, M_2) := \{(i,j) : 0 \leq i \leq M_1 - 1,\ 0 \leq j \leq M_2 - 1\}.$$

Recall that we measure distance in the grid using the Manhattan metric or ($L^1$-norm)

$$||(i,j) - (i',\ j')||_1 := |i - i'| + |j - j'|.$$

An *admissible labeling* of the $M_1 \times M_2$ grid $G(M_1, M_2)$ is a bijection $\phi : [1, M_1M_2] \to G(M_1, M_2)$. The 2-*spacing* $s_2(\phi)$ of a labeling is

$$s_2(\phi) = \min\{||\phi(m) - \phi(m+1)||_1 : 1 \leq m \leq M_1M_2 - 1\}. \tag{3.1}$$

The *circular* 2-*spacing* $w_2^*(\phi)$ is the minimum Manhattan distance between consecutive points in $G_{M_1,M_2}$, viewing $G_{M_1,M_2}$ as a torus. That is,

$$s_2^*(\phi) = \min\{||\phi(m) - \phi(m+1)||_T : 1 \leq m \leq M_1M_2 - 1\}, \tag{3.2}$$

in which $||\cdot||_T$ is the torus Manhattan metric

$$||(i,j) - (i',\ j')||_T := |i - i' \mod{}^* M_1| + |j - j' \mod{}^* M_2|, \tag{3.3}$$

where $r \bmod{}^* M$ denotes the least absolute value residue mod $M$, i.e., that residue which lies in the range $[-M/2,\ (M-1)/2]$. Clearly

$$s_2^*(\phi) \leq s_2(\phi). \tag{3.4}$$

We start with an easy upper bound.

Theorem 3.1. *For any labeling $\phi$,*

$$s_2(\phi) \leq \begin{cases} \left\lfloor \dfrac{M_1 + M_2}{2} \right\rfloor & \text{if } M_1 \text{ or } M_2 \text{ is even,} \\[2ex] \dfrac{M_1 + M_2}{2} - 1 & \text{if } M_1, M_2 \text{ are both odd.} \end{cases} \tag{3.5}$$

*Proof.* The "central cell" in the grid is $\mathbf{v} := \left(\lfloor\frac{M_1+1}{2}\rfloor, \lfloor\frac{M_2+1}{2}\rfloor\right)$. All cells in the grid are at Manhattan distance at most $\lfloor\frac{M_1+M_2}{2}\rfloor$ from $\mathbf{v}$ if $M_1$ or $M_2$ is even and at distance at most $\frac{M_1+M_2}{2} - 1$ from $\mathbf{v}$ if $M_1,\ M_2$ are both odd. The "extreme" cell is $(M_1, M_2)$.

Now let $\phi(\mathbf{v}) = k$. At least one of the values $k-1$ and $k+1$ falls inside the range $[1, M_1M_2]$. If $k'$ denotes this value, then

$$s_2(\phi) \leq ||\phi(k) - \phi(k')||_1 = ||\left(\left\lfloor\frac{M_1+1}{2}\right\rfloor, \left\lfloor\frac{M_2+1}{2}\right\rfloor\right) - (i,j)||_1,$$

which gives (3.5). ◻

THEOREM 3.2. *For all $M_1,\ M_2 \geq 2$ there exists an MRV labeling $\phi$ such that*

$$s_2^*(\phi) = \begin{cases} \left\lfloor\dfrac{M_1+M_2}{2}\right\rfloor - 1 & \textit{if } M_1 \textit{ or } M_2 \textit{ is odd,} \\ \left\lfloor\dfrac{M_1+M_2}{2}\right\rfloor - 2 & \textit{if } M_1 \textit{ and } M_2 \textit{ are even.} \end{cases} \tag{3.6}$$

*Proof.* We exhibit MRV labelings having the desired property. We use different constructions according to the parity of $M_1$ and $M_2$.

*Case* 1. $M_1$ and $M_2$ are both odd.

Set

$L =$ least common multiple (l.c.m.) of $M_1$ and $M_2$,

$G =$ greatest common divisor (g.c.d.) of $M_1$ and $M_2$

and note that $M_1M_2 = LG$. We use the MRV mapping $\phi$ of rank 2 given by

$$\mathbf{y}_1 = \left(\frac{M_1-1}{2}, \frac{M_2+1}{2}\right) \quad \text{and} \quad \mathbf{y}_2 = (1,0),$$

with mixed radix $\mathbf{B} = (L, G)$. If $m = iL + j$ with $0 \leq i < G$ and $0 \leq j < L$, then

$$\phi(m) = \left(i + j\left(\frac{M_1-1}{2}\right) \pmod{M_1},\ j\left(\frac{M_2+1}{2}\right) \pmod{M_2}\right).$$

We prove that $\phi$ is an MRV labeling. Suppose that $\phi(m) = \phi(m')$ with associated radix expansions $(i,j)$ and $(i',j')$. Then

$$j\left(\frac{M_2-1}{2}\right) \equiv j'\left(\frac{M_2-1}{2}\right) \pmod{M_2}.$$

Since $M_2$ is odd, 2 is invertible (mod $M_2$); hence this gives $j \equiv j'$ (mod $M_2$), so $j = j'$. Now

$$i + j\left(\frac{M_1-1}{2}\right) = i + j'\left(\frac{M_1-1}{2}\right) \equiv i' + j'\left(\frac{M_1-1}{2}\right) \pmod{M_1};$$

hence $i \equiv i'$ (mod $M_1$) so $i = i'$, and $\phi$ is one to one.

We now show that

$$s_2^*(\phi) \geq \left\lfloor\frac{M_1+M_2}{2}\right\rfloor - 1. \tag{3.7}$$

To see this, note that incrementing $m$ by 1 increments $j$ by 1 (mod $L$), while

$$L\left(\frac{M_1-1}{2},\ \frac{M_2-1}{2}\right) \equiv (0,0) \pmod{\mathbf{M}}.$$

Thus we always add the vector $\left(\frac{M_1-1}{2},\ \frac{M_2-1}{2}\right)$, which has torus Manhattan length $\frac{M_1+M_2}{2}-1$. At certain steps the vector (1,0) is also added, which gives the vector $\left(\frac{M_1+1}{2},\ \frac{M_2-1}{2}\right)$, which also has torus Manhattan length $\frac{M_1+M_2}{2}-1$. Thus (3.10) follows.

*Case* 2. One of $M_1,\ M_2$ is even and the other is odd.

We treat the case $M_1$ is even and $M_2$ is odd, the other case being similar. For $1 \le K \le M_1M_2$ set $L^* = 2M_2$ and consider the MRV mapping $\phi$ with

$$\mathbf{y}_1 = \left(\frac{M_1}{2},\ \frac{M_2-1}{2}\right) \text{ and } \mathbf{y}_2 = (1,0),$$

and mixed radix base $\mathbf{B} := \left(2M_2,\ \frac{M_1}{2}\right)$. We will show that this is the desired MRV labeling. We have

$$\phi(m) = \left(i + j\frac{M_1}{2} \pmod{M_1},\ j\left(\frac{M_2-1}{2}\right) \pmod{M_2}\right), \tag{3.8}$$

where

$$m-1 := iL^* + j, \text{ with } 0 \le j \le L^*-1,\ \ 0 \le i < \frac{M_1}{2}. \tag{3.9}$$

To see that $\phi$ is one to one, observe that the second coordinate of $\phi(m)$ determines $j \pmod{M_2}$, since $M_2$ is odd. Then the first coordinate determines $i (\text{mod } \frac{M_1}{2})$, and also $j(\text{mod } 2)$, which determines $j \pmod{2M_2}$. Now $m$ is reconstructible by (3.9).

Next, since $\phi(m+1)-\phi(m)$ is one of $\left(\frac{M_1}{2},\ \frac{M_2-1}{2}\right)$ by $\left(\frac{M_1+2}{2},\ \frac{M_2-1}{2}\right)$, we obtain

$$s_2^*(\phi) \ge \frac{M_1+M_2-3}{2} = \left\lfloor \frac{M_1+M_2}{2} \right\rfloor - 1,$$

as desired.

*Case* 3. $M_1$ and $M_2$ are both even.

This is the most complicated case. A rank-3 MRV labeling is required. Define $L^*$ to be the least $\ell > 0$ such that

$$\ell\left(\frac{M_1}{2}-1\right) \equiv 0 \pmod{M_1},$$

$$\ell\left(\frac{M_2}{2}-1\right) \equiv 0 \pmod{M_2}.$$

Then $L^*$ is given by

$$L^* = \begin{cases} \frac{1}{2}\ \text{l.c.m.}\ (M_1, M_2) & \text{if } \frac{M_1M_2}{4} \text{ is odd,} \\ \text{l.c.m.}\ (M_1, M_2) & \text{if } \frac{M_1M_2}{4} \text{ is even.} \end{cases}$$

Set $G^* = \text{g.c.d. } (M_1, M_2)$ so that $M_1 M_2 = H^* G^* L^*$, with

$$H^* = \begin{cases} 2 & \text{if } \dfrac{M_1 M_2}{4} \text{ is odd,} \\ \\ 1 & \text{if } \dfrac{M_1 M_2}{4} \text{ is even.} \end{cases}$$

We use the MRV mapping $\phi$ defined by

$$\mathbf{y}_1 = \left( \frac{M_1 - 2}{2}, \frac{M_2 - 2}{2} \right), \quad \mathbf{y}_2 = (0,1), \quad \text{and} \quad \mathbf{y}_3 = (1,0),$$

with mixed radix base $\mathbf{B} = (B_1, B_2, B_3) := (L^*, G^*, H^*)$. In the case that $H^* = 1$ this simplifies to an MRV mapping on two generators, since $\mathbf{y}_3$ can be omitted. If

$$m = \ell G^* L^* + i L^* + j$$

with

$$0 \le j \le L^* - 1; \; 0 \le i \le G^* - 1; \; 0 \le \ell \le H^* - 1,$$

then

$$\phi(m) := \left( \ell + j \left( \frac{M_1 - 2}{2} \right) \pmod{M_1}, \; i + j \left( \frac{M_2 - 2}{2} \right) \pmod{M_2} \right). \tag{3.10}$$

We omit a proof that $\phi$ is one to one, which can be derived from (3.10) by an argument similar to that in Case 2.

To establish the bound

$$s_2^*(\phi^*) \ge \frac{M_1 + M_2}{2} - 2,$$

use the fact that each step from $\phi(m)$ to $\phi(m+1)$ is one of

$$\left( \frac{M_1 - 2}{2}, \frac{M_2 - 2}{2} \right), \; \left( \frac{M_1}{2}, \frac{M_2 - 2}{2} \right), \text{ or } \left( \frac{M_1}{2}, \frac{M_2}{2} \right). \qquad \square$$

*Examples.* As a Case 1 example, take $M_1 = 5$, $M_2 = 5$, in which case $L = 5$, $G = 5$. Here $\left( \frac{M_1 - 1}{2}, \frac{M_2 - 1}{2} \right) = (2,2)$ and the labeling is given below. (We use matrix notation, with rows numbered 0 to 4, reading down, and columns numbered 0 to 4, reading across.)

$$\begin{bmatrix} 1 & 24 & 17 & 15 & 8 \\ 6 & 4 & 22 & 20 & 13 \\ 11 & 9 & 2 & 25 & 18 \\ 16 & 14 & 7 & 5 & 23 \\ 21 & 19 & 12 & 10 & 3 \end{bmatrix}$$

In this example

$$s_2^*(\phi) = 4 = \left\lfloor \frac{M_1 + M_2}{2} \right\rfloor - 1.$$

Note that $\phi(5)$ and $\phi(6)$ are at distance 4 in the torus Manhattan metric.

As a Case 3 example, for $M_1 = 4,\ M_2 = 6$, we have $L^* = 12$, $G^* = 2$, and $M^* = 1$ and obtain the labeling (in matrix form)

$$\begin{bmatrix} 1 & 13 & 5 & 17 & 9 & 21 \\ 10 & 22 & 2 & 14 & 6 & 18 \\ 7 & 19 & 11 & 23 & 3 & 15 \\ 4 & 16 & 8 & 20 & 12 & 24 \end{bmatrix}.$$

**4. Two-dimensional case: Separating several points.** We consider the problem of finding labelings $\phi$ of $G(M_1, M_2)$ that well-separate all sequences of $k$ consecutive points. We define the *k-spacing* of a labeling to be

$$(4.1)\qquad s_k(\phi) := \min\{||\phi(m) - \phi(m+j)|| : 1 \le m < m+j \le M_1M_2,\ 1 \le j \le k\}.$$

We define the *circular k-spacing* $s_k^*(\phi)$ analogously to be

$$(4.2)\qquad s_k^*(\phi) := \min\{||\phi(m) - \phi(m+j)||_T : 1 \le m < m+j \le M_1M_2,\ 1 \le j \le k\},$$

using the torus Manhattan distance measure (3.3). Note that $s_k^*(\phi) \le s_k(\phi)$.

How large can one make $s_k(\phi)$? We define

$$S_k(M_1, M_2) := \max\{s_k(\phi) : \phi \ \text{ a labeling for } G_{M,N}\}.$$

Upper bounds for $S_k(M_1, M_2)$ depend on the shape of the rectangular grid $G(M_1, M_2)$. Using symmetry considerations we may reduce to the case that $M_1 \le M_2$. If the grid is narrow in one direction, there is an upper bound proportional to $\frac{M_1+M_2}{k}$, while if $M_1$ and $M_2$ are within a constant ratio of one another, there is an upper bound proportional to $\frac{M_1+M_2}{\sqrt{k}}$, as the following result shows.

THEOREM 4.1. *Suppose that $k \ge 3$ and that $M_1 \le M_2$.*

(i) *If $M_2 \ge \frac{k}{3}M_1$, then*

$$(4.3)\qquad S_k(M_1, M_2) \le \frac{3M_2}{k}.$$

(ii) *If $M_1 \le M_2 \le \frac{k}{3}M_1$, then*

$$(4.4)\qquad S_k(M_1, M_2) \le \left(\frac{8M_1M_2}{k}\right)^{1/2}.$$

*Proof.* Let $V(\mathbf{x}; D)$ denote the set of cells within Manhattan distance at most $D$ of a cell $\mathbf{x}$ in the square lattice $\mathbb{Z}^2$. The number of such cells is

$$1 + 4\binom{D+1}{2} = 2D^2 + 2D + 1.$$

We obtain upper bounds for $S = S_k(M_1, M_2)$ using a packing argument, based on the geometric fact that none of the regions $V(\mathbf{x}_i; D) \cap G(M_1, M_2)$ can overlap for $D = \lfloor \frac{1}{2}S \rfloor$. If we define

$$N(\mathbf{x}; D) := |V(\mathbf{x}; D) \cap G(M_1, M_2)|,$$

then we have the bound

$$\sum_{i=1}^{k} N\left(\mathbf{x}_i, \left\lfloor \frac{1}{2}S \right\rfloor\right) \leq M_1 M_2. \tag{4.5}$$

To bound $S$ from above we need lower bounds for $N(\mathbf{x}; D)$.

CLAIM. (i) *If $D \leq M_1 \leq M_2$, then*

$$N(\mathbf{x}; D) \geq \binom{D+2}{2} = \frac{1}{2}D^2 + \frac{3}{2}D + 1, \tag{4.6}$$

*and equality occurs if $\mathbf{x}$ is a corner cell of $G(M_1, M_2)$.*

(ii) *If $M_1 \leq D \leq M_2$, then*

$$N(\mathbf{x}; D) \geq (D - M_1)M_1 + \binom{M_1+1}{2} = DM_1 - \frac{1}{2}M_1^2 + \frac{1}{2}M_1, \tag{4.7}$$

*and equality occurs if $\mathbf{x}$ is a corner cell of $G(M_1, M_2)$.*

The claim follows by inspection of the possible ways that the region $N(\mathbf{x}, D)$ may intersect the boundary of $G(M_1, M_2)$. Figure 4.1 illustrates two cases for $D \leq M_1$; Figure 4.2 illustrates cases with $M_1 \leq D \leq M_2$. The distinction below $D \leq M_1$ and $D \geq M_1$ arises because for $D \geq M_1$ two opposite sides of $N(\mathbf{x}, D)$ always lie on the boundary of the rectangle $G(M_1, M_2)$. In all cases the corner cell $\mathbf{x}$ minimizes the number of points in $V(\mathbf{x}, D) \cap G(M_1, M_2)$.

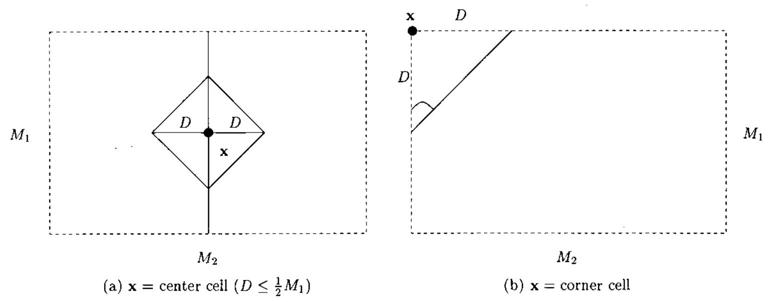


FIG. 4.1. *Regions $V(\mathbf{x}, D) \cap G(M_1, M_2)$ for $D \leq M_1 \leq M_2$.*

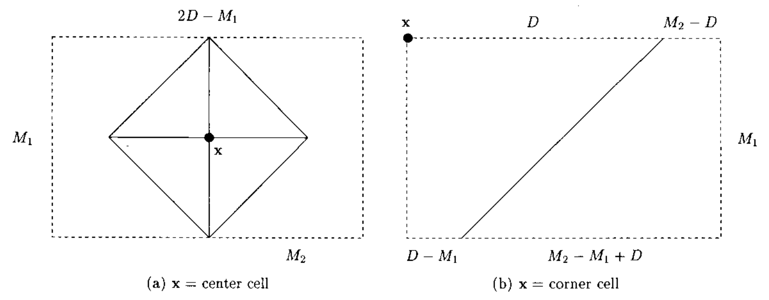


FIG. 4.2. *Regions $V(\mathbf{x}, D) \cap G(M_1, M_2)$ for $M_1 \leq D \leq M_2$.*

We note that when $M_1 \le M_2$ the bound (4.7) of the claim is actually valid for all $D \le M_2$, because it is weaker than the bound (4.6) when $D \le M_1$. However, (4.6) can be applied only when we know that $D = \lfloor \frac{1}{2}S \rfloor \le M_1$. Note that $D = \lfloor \frac{1}{2}S \rfloor \le \frac{1}{2}(M_1 + M_2) \le M_2$.

We obtain upper bounds for $S$ by substituting the bounds of the claim into (4.5). Substituting (4.7) in (4.5) yields

$$k \left( \left\lfloor \frac{1}{2}S \right\rfloor M_1 - \frac{1}{2}M_1^2 + \frac{1}{2}M_1 \right) \le M_1 M_2;$$

hence

$$\frac{1}{2}S \le \left\lfloor \frac{1}{2}S \right\rfloor + \frac{1}{2} \le \frac{M_2}{k} + \frac{1}{2}\frac{M_1}{k}; \tag{4.8}$$

since $M_1 \le M_2$, this yields

$$S \le \frac{3M_2}{k}. \tag{4.9}$$

This bound is universally valid and gives (i) as a special case.

Now if $M_2 \le \frac{k}{3}M_1$, then (4.9) gives $S \le M_1$. In this case we may legitimately apply the bound (4.6) in (4.5) to obtain

$$k \left( \frac{1}{2} \left( \left\lfloor \frac{1}{2}S \right\rfloor \right)^2 + \frac{3}{2} \left\lfloor \frac{1}{2}S \right\rfloor + 1 \right) \le M_1 M_2.$$

Thus

$$\left( \frac{1}{2}S + 1 \right)^2 \le \left( \left\lfloor \frac{1}{2}S \right\rfloor + \frac{3}{2} \right)^2 = \left\lfloor \frac{1}{2}S \right\rfloor^2 + 3 \left\lfloor \frac{1}{2}S \right\rfloor + \frac{9}{4} \le \frac{2M_1M_2}{k} + \frac{1}{4},$$

which yields

$$(S+2)^2 \le \frac{8M_1M_2}{k} + 1, \tag{4.10}$$

from which the bound of (ii) follows. ☐

The upper bounds of Theorem 4.1 are conservative and can potentially be improved by a multiplicative constant, which varies between $\sqrt{2}$ and 2, for most values of $M_1$ and $M_2$. This occurs because "nearly all" cells $\mathbf{x} \in G(M_1, M_2)$ have $N(\mathbf{x}, D)$ at least twice as large as the value of $N(\mathbf{x}, D)$ of a corner cell, but $N(\mathbf{x}, D)$ is never more than four times as large the value of $N(\mathbf{x}, D)$ for a corner cell.

We show that there exist MRV labelings that achieve $k$-spacings within a multiplicative constant of $\frac{1}{3}$ of the upper bound (i) of Theorem 4.1.

THEOREM 4.2. *Suppose that $k \ge 3$ and consider a grid $G(M_1, M_2)$ with $M_2 \ge M_1 \ge 3$. If $M_2 \ge \frac{k}{3}M_1$, then there exists an MRV labeling $\phi$ with*

$$s_k^*(\phi) \ge \left\lfloor \frac{M_2}{k} \right\rfloor - 2. \tag{4.11}$$

*Proof.* We use an MRV labeling $\phi$ that fills the rows of $G(M_1, M_2)$ one at a time. The hypotheses imply $M_2 \ge k$, so $\lfloor \frac{M_2}{k} \rfloor \ge 1$. Let $\phi$ be the MRV mapping of rank 3 that uses the vectors

$$\mathbf{y}_1 = \left( 0, \left\lfloor \frac{M_2}{k} \right\rfloor \right), \quad \mathbf{y}_2 = (0,1), \quad \mathbf{y}_3 = (1,0)$$

and the base $\mathbf{B} = (B_1, B_2, B_3)$ given by

$$B_1 = \frac{M_2}{\text{g.c.d.}\left(M_2, \left\lfloor \frac{M_2}{k} \right\rfloor\right)}, \quad B_2 = \text{g.c.d.}\left(M_2, \left\lfloor \frac{M_2}{k} \right\rfloor\right), \quad B_3 = M_1.$$

If $B_2 = 1$, then the rank drops to 2, in which case $\mathbf{y}_2$ is omitted. We have $B_1 B_2 B_3 = M_1 M_2$ and the lattice $\Lambda = \Lambda_{\mathcal{Y},G}$ is the set of vectors $(m_1, m_2, m_3) \in \mathbb{Z}^3$ such that

$$m_3 \equiv 0 \pmod{M_1}, \tag{4.12a}$$

$$m_1 \left\lfloor \frac{M_2}{k} \right\rfloor + m_2 \equiv 0 \pmod{M_2}. \tag{4.12b}$$

The criterion of Theorem 2.1(ii) for $\Lambda$ is easily checked: either $|m_1| \geq B_1$ or $|m_2| \geq B_2$ follows from (4.12b) if $(m_1, m_2) \neq (0,0)$ while (4.12a) gives $|m_3| \geq M_1 = B_3$ if $m_3 \neq 0$. Thus $\phi$ is one to one, and hence is an MRV labeling.

Any $k$ consecutive values of $\mathbf{y}_1$ have each pair of values separated by at least $||\mathbf{y}_1|| = \lfloor \frac{M_2}{k} \rfloor$ in their second coordinate, in the torus metric. Also the cycle of $\mathbf{y}_1$ values which occur before a repeated value is of length at least $k$; hence the updates by $\mathbf{y}_2$ and $\mathbf{y}_3$ occur at time intervals more than $k$ units apart. It follows that each $k$ consecutive vectors differ from a translate of consecutive multiples of $\mathbf{y}_1$ by at most $||\mathbf{y}_2|| + ||\mathbf{y}_3|| \leq 2$. This yields (4.11). ◻

Constructing MRV labelings to get within a multiplicative constant factor of the optimal spacing in the remaining upper bound range (ii) of Theorem 4.1 appears complicated, and a general proof would seem to require consideration of many different MRV mappings. We do not attempt it here. In the special case that $M_1 = M_2 = M$, $k = \ell^2$, and $\ell$ divides $M$, a good MRV labeling is easy to come by. It is of rank 4 with

$$\mathbf{y}_1 = \left(0, \frac{M}{\ell}\right), \quad \mathbf{y}_2 = \left(\frac{M}{\ell}, 0\right), \quad \mathbf{y}_3 = (0,1), \quad \text{and} \quad \mathbf{y}_4 = (1,0),$$

with mixed radix $\mathbf{B} = (\ell, \ell, \frac{M}{\ell}, \frac{M}{\ell})$. In this example

$$s_k(\phi) \geq \left(\left\lfloor \frac{M_1 M_2}{k} \right\rfloor\right)^{1/2} - 2.$$

A special case of this mapping appears in the example at the end of section 2, with $M = 6$, $k = 4$, and $\ell = 2$. It has

$$s_4(\phi) = 2 \geq \left(\left\lfloor \frac{36}{4} \right\rfloor\right)^{1/2} - 2 = 1.$$

We conjecture that for $k \geq 3$ and $M_1 \leq M_2 \leq \frac{k}{3} M_1$ there exist MRV labelings that achieve

$$s_k(\phi) \geq c_0 \left(\left\lfloor \frac{M_1 M_2}{k} \right\rfloor\right)^{1/2} - 2 \tag{4.13}$$

for some positive constant $c_0$, e.g., $c_0 = \frac{1}{2}$.

**Acknowledgment.** The author thanks the referee for helpful comments and for pointing out reference [4, Problem D1].

## REFERENCES

[1] A. Adler, *Magic cubes and the* 3*-adic zeta function*, Math. Intelligencer, 14 (1992), pp. 14–23.

[2] A. Adler and S.-Y. R. Li, *Magic $N$-cubes and Prouhet sequences*, Amer. Math. Monthly, 84 (1977), pp. 618–627.

[3] A. M. Bruckstein, R. J. Holt, and A. Netravali, *Holographic representations of images*, IEEE Trans. Image Process., 7 (1998), pp. 1583–1597.

[4] H. T. Croft, K. J. Falconer, and R. J. Guy, *Unsolved Problems in Geometry*, Springer-Verlag, New York, 1991.

[5] M. Kraitchik, *Traite de Carrés Magiques*, Gauther–Villiers, Paris, 1930.

[6] R. P. Stanley, *Magic labelling of graphs, symmetric magic squares, systems of parameters and Cohen-Macaulay rings*, Duke Math. J., 43 (1976), pp. 511–531.

SIAM J. DISCRETE MATH.
Vol. 13, No. 4, pp. 535–570

# LEARNING POLYNOMIALS WITH QUERIES: THE HIGHLY NOISY CASE*

ODED GOLDREICH[†], RONITT RUBINFELD[‡], AND MADHU SUDAN[§]

**Abstract.** Given a function $f$ mapping $n$-variate inputs from a finite field $F$ into $F$, we consider the task of reconstructing a list of all $n$-variate degree $d$ polynomials that agree with $f$ on a tiny but nonnegligible fraction, $\delta$, of the input space. We give a randomized algorithm for solving this task. The algorithm accesses $f$ as a black box and runs in time polynomial in $\frac{n}{\delta}$ and exponential in $d$, provided $\delta$ is $\Omega(\sqrt{d/|F|})$. For the special case when $d = 1$, we solve this problem for all $\epsilon \overset{\text{def}}{=} \delta - \frac{1}{|F|} > 0$. In this case the running time of our algorithm is bounded by a polynomial in $\frac{1}{\epsilon}$ and $n$. Our algorithm generalizes a previously known algorithm, due to Goldreich and Levin [in *Proceedings of the* 21*st Annual ACM Symposium on Theory of Computing,* Seattle, WA, ACM Press, New York, 1989, pp. 25–32.], that solves this task for the case when $F = \mathrm{GF}(2)$ (and $d = 1$).

In the process we provide new bounds on the number of degree $d$ polynomials that may agree with any given function on $\delta \geq \sqrt{d/|F|}$ fraction of the inputs. This result is derived by generalizing a well-known bound from coding theory on the number of codewords from an error-correcting code that can be "close" to an arbitrary word; our generalization works for codes over arbitrary alphabets, while the previous result held only for binary alphabets.

**Key words.** multivariate polynomials, noisy reconstruction, error-correcting codes

**AMS subject classification.** 68Q15

**PII.** S0895480198344540

## 1. Introduction.

We consider the following archetypal reconstruction problem:
**Given**: An oracle (black box) for an arbitrary function $f : F^n \to F$, a class of functions $\mathcal{C}$, and a parameter $\delta$.
**Output**: A list of all functions $g \in \mathcal{C}$ that agree with $f$ on at least $\delta$ fraction of the inputs.

The reconstruction problem can be interpreted in several ways within the framework of computational learning theory. First, it falls within the framework of learning with persistent noise. Here one assumes that the function $f$ is derived from some function in the class $\mathcal{C}$ by "adding" noise to it. Typical works in this direction either tolerate only small amounts of noise [2, 42, 21, 39] (i.e., that the function is modified only at a small fraction of all possible inputs) or assume that the noise is random [1, 26, 20, 25, 33, 13, 36] (i.e., that the decision of whether or not to modify the

*Received by the editors September 14, 1998; accepted for publication (in revised form) August 22, 2000; published electronically November 21, 2000. A preliminary version of this paper appeared in the Proceedings of the 36*th Annual Symposium on Foundations of Computer Science*, Milwaukee, WI, IEEE, New York, 1995, pp. 294–303.

http://www.siam.org/journals/sidma/13-4/34454.html

[†]Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel 76100 (oded@wisdom.weizmann.ac.il). The research of this author was supported by grant 92-00226 from the United States–Israel Binational Science Foundation (BSF), Jerusalem, Israel.

[‡]NEC Research Institute, 4 Independence Way, Princeton, NJ, 08540 (ronitt@research.nj.nec.com). This work was done when this author was at Cornell University. This research was supported in part by ONR Young Investigator Award N00014-93-1-0590 and grant 92-00226 from the United States–Israel Binational Science Foundation (BSF), Jerusalem, Israel.

[§]MIT Laboratory for Computer Science, Cambridge, MA 02139 (madhu@mit.edu). Part of this work was done when this author was at the IBM Thomas J. Watson Research Center. This research was supported in part by a Sloan Foundation Fellowship and NSF Career Award CCR-9875511.

function at any given input is made by a random process). In contrast, we take the setting to an extreme by considering a *very large amount* of (possibly adversarially chosen) noise. In particular, we consider situations in which the noise disturbs the outputs for almost all inputs.

A second interpretation of the reconstruction problem is within the framework of "agnostic learning" introduced by Kearns, Shapire, and Sellie [23] (see also [29, 30, 24]). In the setting of agnostic learning, the learner is to make no assumptions regarding the natural phenomenon underlying the input-output relationship of the function, and the goal of the learner is to come up with a simple explanation that best fits the examples. Therefore the best explanation may account for only part of the phenomenon. In some situations, when the phenomenon appears very irregular, providing an explanation that fits only part of it is better than nothing. Kearns, Shapire, and Sellie did not consider the use of queries (but rather examples drawn from an arbitrary distribution) since they were skeptical that queries could be of any help. We show that queries do seem to help (see below).

Yet another interpretation of the reconstruction problem, which generalizes the "agnostic learning" approach, is the following. Suppose that the natural phenomena can be explained by several simple explanations that together cover most of the input-output behavior but not all of it. Namely, suppose that the function $f$ agrees almost everywhere with one of a small number of functions $g_i \in \mathcal{C}$. In particular, assume that each $g_i$ agrees with $f$ on at least a $\delta$ fraction of the inputs but that for some (say $2\delta$) fraction of the inputs $f$ does not agree with any of the $g_i$'s. This setting was investigated by Ar et al. [3]. The reconstruction problem described above may be viewed as a (simpler) abstraction of the problem considered in [3]. As in the case of learning with noise, there is no explicit requirement in the setting of [3] that the noise level be small, but all their results require that the fraction of inputs left unexplained by the $g_i$'s be smaller than the fraction of inputs on which each $g_i$ agrees with $f$. Our relaxation (and results) do not impose such a restriction on the noise and thus make the setting more appealing and closer in spirit to "agnostic learning."

**1.1. Our results.** In this paper, we consider the special case of the reconstruction problem when the hypothesis class is the set of $n$-variate polynomials of bounded total degree $d$. (The total degree of a monomial $\prod_i x_i^{d_i}$ is $\sum_i d_i$, that is, the sum of the degrees of the variables in the monomial. The total degree of a polynomial is the maximum total degree of monomials with nonzero coefficient in the polynomial. For example, the total degree of the polynomial $x_1^2 x_2^3 + 5x_2^4$ is 5.) The most interesting aspect of our results is that they relate to *very small* values of the parameter $\delta$ (the fraction of inputs on which the hypothesis has to fit the function $f$). Our main results are as follows:

- An algorithm that given $d$, $F$, and $\delta = \Omega(\sqrt{d/|F|})$ and, provided oracle access to an arbitrary function $f : F^n \to F$, runs in time $(n/\delta)^{O(d)}$ and outputs a list including all degree $d$ polynomials that agree with $f$ on at least a $\delta$ fraction of the inputs.
- An algorithm that given $F$ and $\epsilon > 0$ and, provided oracle access to an arbitrary function $f : F^n \to F$, runs in time poly$(n/\epsilon)$ and outputs a list including all linear functions (degree $d = 1$ polynomials) that agree with $f$ on at least a $\delta \overset{\text{def}}{=} \frac{1}{|F|} + \epsilon$ fraction of the inputs.
- A new bound on the number of degree $d$ polynomials that may agree with a given function $f : F^n \to F$ on a $\delta \geq \sqrt{d/|F|}$ fraction of the inputs. This bound

is derived from a more general result about the number of codewords from an error-correcting code that may be close to a given word.

A special case of interest is when the function $f$ is obtained by picking an arbitrary degree $d$ polynomial $p$, and letting $f$ agree with $p$ on an *arbitrary* $\delta = \Omega(\sqrt{\frac{d}{|F|}})$ fraction of the inputs and be set at random otherwise.[1] In this case, with high probability, only one polynomial (i.e., $p$) agrees with $f$ on a $\delta$ fraction of the inputs (see section 5). Thus, in this case, the above algorithm will output only the polynomial $p$.

*Additional Remarks.*

1. Any algorithm for the explicit reconstruction problem as stated above would need to output all the coefficients of such a polynomial, requiring time at least $\binom{n+d}{d}$. Moreover, the number of such polynomials could grow as a function of $\frac{1}{\delta}$. Thus it seems reasonable that the running time of such a reconstruction procedure should grow as a polynomial function of $\frac{1}{\delta}$ and $\binom{n}{d}$.
   We stress that the above comment does not apply to "implicit reconstruction" algorithms as discussed in section 1.4.
2. For $d < |F|$, the condition $\delta > \frac{d}{|F|}$ seems a natural barrier for our investigation, since there are *exponentially many* (in $n$) degree $d$ polynomials that are at distance $\approx \frac{d}{|F|}$ from some functions (see Proposition 4.8).
3. Queries seem essential to our learning algorithm. We provide two indications to our belief, both referring to the special case of $F = \mathrm{GF}(2)$ and $d = 1$. First, if queries are not allowed, then a solution to the reconstruction problem yields a solution to the longstanding open problem of "decoding random (binary) linear codes." (Note that each random example given to the reconstruction algorithm corresponds to a random linear equation on the information variables. We admit that the longstanding open problem is typically stated for a linear number of equations, but nothing is known even in case the number of equations is polynomial in the information length.)
   Another well-studied problem that reduces to the problem of noisy reconstruction is the problem of "learning parity with noise" [20], which is commonly believed to be hard when one is only allowed uniformly and independently chosen examples [20, 7, 22]. Learning parity with noise is considered hard even for random noise, whereas here the noise is adversarial.
4. In section 6, we give evidence that the reconstruction problem may be hard, for $\delta$ very close to $d/|F|$, even in the case where $n = 2$. A variant is shown to be hard even for $n = 1$.

**1.2. A coding theory perspective.** We first introduce the formal definition of an error-correcting code (see, e.g., [31]). For positive integers $N, K, D$, and $q$, an $[N, K, D]_q$ error-correcting code is a collection of $q^K$ sequences of $N$-elements each from $\{1, \dots, q\}$, called codewords, in which no two sequences have a "Hamming distance" of less than $D$ (i.e., every pair of codewords disagree on at least $D$ locations).

Polynomial functions lead to some of the simplest known constructions of error-correcting codes: A function from $F^n$ to $F$ may be viewed as an element of $F^{|F|^n}$—by writing down explicitly the function's value on all $|F|^n$ inputs. Then the "distance property" of polynomials yields that the set of sequences corresponding to bounded-degree polynomial functions form an error-correcting code with nontrivial parameters

[1]This is different from "random noise" as the set of corrupted inputs is selected adversarially—only the values at these inputs are random.

(for details, see Proposition 4.3). Specifically, the set of $n$-variate polynomials of total degree $d$ over $F = \mathrm{GF}(q)$ yields an $[N, K, D]_q$ error-correcting code with $N = |F|^n$, $K = \binom{n+d}{d}$, and $D = (|F| - d) \cdot |F|^{n-1}$. These constructions have been studied in the coding theory literature. The case $n = 1$ leads to "Reed–Solomon codes," while the case of general $n$ is studied under the name "Reed–Muller codes."

Our reconstruction algorithm may be viewed as an algorithm that takes an arbitrary word from $F^{|F|^n}$ and finds a list of all codewords from the Reed–Muller code that agree with the given word in $\delta$ fraction of the coordinates (i.e., $1 - \delta$ fraction of the coordinates have been corrupted by errors). This task is referred to in the literature as the "list-decoding" problem [11]. For codes achieved by setting $d$ such that $d/|F| \to 0$, our list decoding algorithm recovers from errors when the rate of errors approaches 1. We are not aware of any other case where an approach other (and better) than brute-force can be used to perform list decoding with the error-rate approaching 1. Furthermore, our decoding algorithm works without examining the entire codeword.

**1.3. Related previous work.** For the sake of scholarly interest, we discuss several related areas in which related work has been done. In this subsection, it would be more convenient to refer to the error-rate $1 - \delta$ rather than to the rate of agreement $\delta$.

*Polynomial interpolation.* When the noise rate is 0, our problem is simply that of polynomial interpolation. In this case the problem is well analyzed and the reader is referred to [48], for instance, for a history of the polynomial interpolation problem.

*Self-correction.* In the case when the noise rate is positive but small, one approach used to solving the reconstruction problem is to use *self-correctors*, introduced independently in [8] and [28]. Self-correctors convert programs that are known to be correct on a fraction $\delta$ of inputs into programs that are correct on each input. Self-correctors for values of $\delta$ that are larger than 3/4 have been constructed for several algebraic functions [5, 8, 9, 28, 34], and in one case this was done for $\delta > 1/2$ [15].[2] We stress that self-correctors correct a given program using only the information that the program is supposed to be computing a function from a given class (e.g., a low-degree polynomial). Thus, when the error is larger than $\frac{1}{2}$ and the class contains more than a single function, such self-correction is no longer possible since there could be more than one function in the class that agrees with the given program on an $\delta < 1/2$ fraction of the inputs.

*Cryptography and learning theory.* In order to prove the security of a certain "hardcore predicate" relative to any "one-way function," Goldreich and Levin solved a special case of the (explicit) reconstruction problem [17]. Specifically, they considered the linear case (i.e., $d = 1$) for the Boolean field (i.e., $F = \mathrm{GF}(2)$) and any agreement rate that is bigger than the error-rate (i.e., $\delta > \frac{1}{2}$). Their ideas were subsequently used by Kushilevitz and Mansour [25] to devise an algorithm for learning Boolean decision trees.

**1.4. Subsequent work.** At the time this work was first published [18] no algorithm other than brute force was known for reconstructing a list of degree $d$ polynomials agreeing with an arbitrary function on a vanishing fraction of inputs, for

[2]Specifically, self-correctors correcting $\frac{1}{\Theta(d)}$ fraction of error for $f$ that are degree $d$ polynomial functions over a finite field $F$, $|F| \geq d + 2$, were found by [5, 28]. For $d/|F| \to 0$, the fraction of errors that a self-corrector could correct was improved to almost 1/4 by [14] and then to almost 1/2 by [15] (using a solution for the univariate case given by [45]).

any $d \geq 2$. Our algorithm solves this problem with exponential dependence on $d$, but with polynomial dependence on $n$, the number of variables. Subsequently some new reconstruction algorithms for polynomials have been developed. In particular, Sudan [40] and Guruswami and Sudan [19] have provided new algorithms for reconstructing *univariate* polynomials from large amounts of noise. Their running time depends only polynomially in $d$ and works for $\delta = \Omega(\sqrt{d/|F|})$. Notice that the agreement required in this case is larger than the level at which our NP-hardness result (of section 6) holds. The results of [40] also provide some reconstruction algorithms for multivariate polynomials, but not for as low an error as given here. In addition, the running time grows exponentially with $n$. Wasserman [44] gives an algorithm for reconstructing polynomials from noisy data that works *without* making queries. The running time of the algorithm of [44] also grows exponentially in $n$ and polynomially in $d$.

As noted earlier (see remark 1 in section 1.1), the running time of any *explicit* reconstruction algorithm has to have an exponential dependence on either $d$ or $n$. However, this need not be true for *implicit* reconstruction algorithms: By the latter term we mean algorithms that produce as output a sequence of oracle machines, such that for every multivariate polynomial that has agreement $\delta$ with the function $f$, one of these oracle machines, given access to $f$, computes that polynomial. Recently, Arora and Sudan [4] gave an algorithm for this implicit reconstruction problem. The running time of their algorithm is bounded by a polynomial in $n$ and $d$, and it works correctly provided that $\delta \geq (d^{O(1)})/|F|^{\Omega(1)}$; that is, their algorithm needs a much higher agreement but works in time polynomial in all parameters. The reader may verify that such an implicit reconstruction algorithm yields an algorithm for the explicit reconstruction problem with running time that is polynomial in $\binom{n+d}{d}$ (e.g., by applying noise-free polynomial-interpolation to each of the oracle machines provided above, and testing the resulting polynomial for agreement with $f$). Finally, Sudan, Trevisan, and Vadhan [41] have recently improved the result of [4], further reducing the requirement on $\delta$ to $\delta > 2\sqrt{d/|F|}$. The algorithm of [41] thus subsumes the algorithm of this paper for all choices of parameters, except $d = 1$.

**1.5. Rest of this paper.** The rest of the paper is organized as follows. In section 2 we motivate our algorithm, starting with the special case of the reconstruction of linear polynomials. The general case algorithm is described formally in section 3, along with an analysis of its correctness and running time assuming an upper bound on the number of polynomials that agree with a given function at $\delta$ fraction of the inputs. In section 4 we provide two such upper bounds. These bounds do not use any special (i.e., algebraic) property of polynomials, but rather apply in general to collections of functions that have large distance between them. In section 5 we consider a random model for the noise applied to a function. Specifically, the output either agrees with a fixed polynomial or is random. In such a case we provide a stronger upper bound (specifically, 1) on the number of polynomials that may agree with the black box. In section 6 we give evidence that the reconstruction problem may be hard for small values of the agreement parameter $\delta$, even in the case when $n = 1$. We conclude with an application of the linear-polynomial reconstruction algorithm to complexity theory: Specifically, we use it in order to prove the security of new, generic, hard-core functions (see section 7).

*Notation.* In what follows, we use GF(q) to denote the finite field on $q$ elements. We assume arithmetic in this field (addition, subtraction, multiplication, division, and comparison with zero) may be performed at unit cost. For a finite set $A$, we use the notation $a \in_R A$ to denote that $a$ is a random variable chosen uniformly at random from $A$. For a positive integer $n$, we use $[n]$ to denote the set $\{1, \ldots, n\}$.

**2. Motivation to the algorithm.** We start by presenting the algorithm for the linear case, and next present some of the ideas underlying the generalization to higher degrees. We stress that whereas section 2.1 provides a full analysis of the linear case, section 2.2 merely introduces the additional ideas that will be employed in dealing with the general case. The presentation in section 2.1 is aimed to facilitate the generalization from the linear case to the general case.

**2.1. Reconstructing linear polynomials.** We are given oracle access to a function $f : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$ and need to find a polynomial (or actually all polynomials) of degree $d$ that agrees with $f$ on at least a $\delta = \frac{d}{q} + \epsilon$ fraction of the inputs, where $\epsilon > 0$. Our starting point is the linear case (i.e., $d = 1$); namely, we are looking for a polynomial of the form $p(x_1, \ldots, x_n) = \sum_{i=1}^{n} p_i x_i$. In this case our algorithm is a generalization of an algorithm due to Goldreich and Levin [17].[3] (The original algorithm is regained by setting $q = 2$.)

We start with a definition: The *i*-prefix of a linear polynomial $p(x_1, \ldots, x_n)$ is the polynomial that results by summing up all of the monomials in which only the first $i$ variables appear. That is, the $i$-prefix of the polynomial $\sum_{j=1}^{n} p_j x_j$ is $\sum_{j=1}^{i} p_j x_j$. The algorithm proceeds in $n$ rounds, so that in the $i$th round we find a list of candidates for the $i$-prefixes of $p$.

In the $i$th round, the list of $i$-prefixes is generated by extending the list of $(i-1)$-prefixes. A simple (and inefficient) way to perform this extension is to first extend each $(i-1)$-prefix in all $q$ possible ways, and then to "screen" the resulting list of $i$-prefixes. A good screening is the essence of the algorithm. It should guarantee that the $i$-prefix of a correct solution $p$ does pass and that not too many other prefixes pass (as otherwise the algorithm consumes too much time).

The screening is done by subjecting each candidate prefix, $(c_1, \ldots, c_i)$, to the following test. Pick $m = \mathrm{poly}(n/\epsilon)$ sequences uniformly from $\mathrm{GF}(q)^{n-i}$. For each such sequence $\bar{s} = (s_{i+1}, \ldots, s_n)$ and for every $\sigma \in \mathrm{GF(q)}$, estimate the quantity

$$P_{\bar{s}}(\sigma) \stackrel{\text{def}}{=} \mathbf{P}_{r_1,\ldots,r_i \in \mathrm{GF}(q)} \left[ f(\bar{r}, \bar{s}) = \sum_{j=1}^{i} c_j r_j + \sigma \right], \tag{2.1}$$

where $\bar{r} = (r_1, \ldots, r_i)$. The value $\sigma$ can be thought of as a guess for $\sum_{j=i+1}^{n} p_j s_j$. For every fixed suffix $\bar{s}$, all these probabilities can be approximated simultaneously by using a sample of $\mathrm{poly}(n/\epsilon)$ sequences $(r_1, \ldots, r_i)$, regardless of $q$. If one of these probabilities is significantly larger than $1/q$, then the test accepts due to this suffix, and if no suffix makes the test accept, then it rejects. The actual algorithm is presented in Figure 2.1.

Observe that a candidate $(c_1, \ldots, c_i)$ passes the test of Figure 2.1 if for at least one sequence of $\bar{s} = (s_{i+1}, \ldots, s_n)$, there exists a $\sigma$ so that the estimate for $P_{\bar{s}}(\sigma)$ is greater than $\frac{1}{q} + \frac{\epsilon}{3}$. Clearly, for a correct candidate (i.e., $(c_1, \ldots, c_i)$ that is a prefix of a polynomial $p = (p_1, \ldots, p_n)$ having at least $\frac{1}{q} + \epsilon$ agreement with $f$) and $\sigma = \sum_{j=i+1}^{n} p_j s_j$, it holds that $\mathbf{E}_{\bar{s}}[P_{\bar{s}}(\sigma)] \geq \frac{1}{q} + \epsilon$. Using Markov's inequality, it follows that for a correct candidate, an $\epsilon/2$ fraction of the suffixes are such that for

[3] We refer to the original algorithm as in [17], not to a simpler algorithm that appears in later versions (cf. [27, 16]).

```
Test-prefix($f, \epsilon, n, (c_1, \dots, c_i)$)
  Repeat $\text{poly}(n/\epsilon)$ times:
    Pick $\bar{s} = s_{i+1}, \dots, s_n \in_R \text{GF(q)}$.
    Let $t \stackrel{\text{def}}{=} \text{poly}(n/\epsilon)$.
    for $k = 1$ to $t$ do
      Pick $\bar{r} = r_1, \dots, r_i \in_R \text{GF(q)}$
      $\sigma^{(k)} \leftarrow f(\bar{r}, \bar{s}) - \sum_{j=1}^{i} c_j r_j$.
    endfor
    If $\exists\ \sigma$ s.t. $\sigma^{(k)} = \sigma$ for at least $\frac{1}{q} + \frac{\epsilon}{3}$ fraction of the $k$'s
      then output accept and halt.
  endRepeat.
  If all iterations were completed without accepting, then reject.
```

FIG. 2.1. *Implementing the screening process.*

each such suffix $\bar{s}$ and some $\sigma$, it holds that $P_{\bar{s}}(\sigma) \geq \frac{1}{q} + \frac{\epsilon}{2}$; thus the correct candidate passes the test with overwhelming probability. On the other hand, for any suffix $\bar{s}$, if a prefix $(c_1, \dots, c_i)$ is to pass the test (with nonnegligible probability) due to suffix $\bar{s}$, then it must be the case that the polynomial $\sum_{j=1}^{i} c_j x_j$ has at least agreement rate of $\frac{1}{q} + \frac{\epsilon}{4}$ with the function $f'(x_1, \dots, x_i) \stackrel{\text{def}}{=} f(x_1, \dots, x_i, s_{i+1}, \dots, s_n)$. It is possible to bound the number of ($i$-variate) polynomials that have so much agreement with any function $f'$. (Section 4 contains some such bounds.) Thus, in each iteration, only a small number of prefixes pass the test, thereby limiting the total number of prefixes that may pass the test in any one of the $\text{poly}(n/\epsilon)$ iterations.

The above yields a $\text{poly}(nq/\epsilon)$-time algorithm. In order to get rid of the $q$ factor in running-time, we need to modify the process by which candidates are formed. Instead of extending each $(i-1)$-prefix, $(c_1, \dots, c_{i-1})$, in all $q$ possible ways, we do the following: We pick uniformly $\bar{s} \stackrel{\text{def}}{=} (s_{i+1}, \dots, s_n) \in \text{GF}(q)^{n-i}$, $\bar{r} \stackrel{\text{def}}{=} (r_1, \dots, r_{i-1}) \in \text{GF}(q)^{i-1}$, and $r', r'' \in \text{GF}(q)$, and solve the system of equations

$$r'x + y = f(r_1, \dots, r_{i-1}, r', s_{i+1}, \dots, s_n) - \sum_{j=1}^{i-1} c_j r_j, \tag{2.2}$$

$$r''x + y = f(r_1, \dots, r_{i-1}, r'', s_{i+1}, \dots, s_n) - \sum_{j=1}^{i-1} c_j r_j \tag{2.3}$$

using the solution for $x$ as the value of the $i$th coefficient (i.e., set $c_i = x$). This extension process is repeated $\text{poly}(n/\epsilon)$ many times, obtaining at most $\text{poly}(n/\epsilon)$ candidate $i$-prefixes, per each candidate $(i-1)$-prefix. We then subject each $i$-prefix in the list to the screening test (presented in Figure 2.1), and keep only the candidates that pass the test.

We need to show that if the $(i-1)$-prefix of a correct solution is in the list of candidates (at the beginning of round $i$), then the $i$-prefix of this solution will be found in the extension process. Let $p = (p_1, \dots, p_n)$ be a correct solution (to the reconstruction problem for $f$). Then $\mathbf{P}_{\bar{r},r,\bar{s}}[p(\bar{r}, r, \bar{s}) = f(\bar{r}, r, \bar{s})] \geq \frac{1}{q} + \epsilon > \epsilon$. It follows that for at least an $\epsilon/2$ fraction of the sequences $(\bar{r}, \bar{s})$, the polynomial $p$ satisfies $p(\bar{r}, r, \bar{s}) = f(\bar{r}, r, \bar{s})$ for at least an $\epsilon/2$ fraction of the possible $r$'s. Let $\sigma$ represent the value of the sum $\sum_{j=i+1}^{n} p_j s_j$, and note that $p(\bar{r}, r, \bar{s}) = \sum_{j=1}^{i-1} p_j r_j + p_i r + \sigma$. Then, with probability $\Omega(\epsilon^3)$ over the choices of $r_1, \ldots, r_{i-1}, s_{i+1}, \ldots, s_n$ and $r', r''$, the following two equations hold:

$$r' p_i + \sigma = f(r_1, \ldots, r_{i-1}, r', s_{i+1}, \ldots, s_n) - \sum_{j=1}^{i-1} p_j r_j,$$

$$r'' p_i + \sigma = f(r_1, \ldots, r_{i-1}, r'', s_{i+1}, \ldots, s_n) - \sum_{j=1}^{i-1} p_j r_j,$$

and $r' \neq r''$. (That is, with probability at least $\frac{\epsilon}{2}$, the pair $(\bar{r}, \bar{s})$ is good, and conditioned on this event $r'$ is good with probability at least $\frac{\epsilon}{2}$, and similarly for $r''$ losing a term of $\frac{1}{q} < \frac{\epsilon}{4}$ to account for $r'' \neq r'$. We may assume that $1/q < \epsilon/4$, since otherwise $q < 4/\epsilon$ and we can afford to perform the simpler procedure above.) Thus, with probability $\Omega(\epsilon^3)$, solving the system (2.2)–(2.3) with $(c_1, \ldots, c_{i-1}) = (p_1, \ldots, p_{i-1})$ yields $x = p_i$. Since we repeat the process $\text{poly}(n/\epsilon)$ times for each $(i-1)$-prefix, it follows that the correct prefix always appears in our candidate list.

Recall that correct prefixes pass the screening process with overwhelmingly high probability. Using Theorem 4.5 (of section 4) to bound the number of prefixes passing the screening process, we have the following.

THEOREM 2.1. *Given oracle access to a function $f$ and parameters $\epsilon, k$, our algorithm runs in* $\text{poly}(\frac{k \cdot n}{\epsilon})$*-time and outputs, with probability at least* $1 - 2^{-k}$*, a list satisfying the following properties:*

1. *The list contains all linear polynomials that agree with $f$ on at least a $\delta = \frac{1}{q} + \epsilon$ fraction of the inputs.*
2. *The list does not contain any polynomial that agrees with $f$ on less than a $\frac{1}{q} + \frac{\epsilon}{4}$ fraction of the inputs.*

**2.2. Generalizing to higher degree.** We remind the reader that in this subsection we merely introduce the additional ideas used in extending the algorithm from the linear case to the general case. The algorithm itself is presented and analyzed in section 3.

Dealing with polynomials of degree $d > 1$ is more involved than dealing with linear polynomials; still, we employ a similar strategy. Our plan is again to "isolate" the terms/monomials in the first $i$ variables and find candidates for their coefficients. In particular, if $p(x_1, \ldots, x_n)$ is a degree $d$ polynomial on $n$ variables, then $p(x_1, \ldots, x_i, 0, \ldots, 0)$ is a degree $\leq d$ polynomial on $i$ variables that has the same coefficients as $p$ on all monomials involving only variables in $\{1, \ldots, i\}$. Thus, $p(x_1, \ldots, x_i, 0, \ldots, 0)$ is the $i$-prefix of $p$.

We show how to extend a list of candidates for the $(i-1)$-prefixes of polynomials agreeing with $f$ into a list of candidates for the $i$-prefixes. Suppose we get the $(i-1)$-prefix $p$ that we want to extend. We select $d+1$ distinct elements $r^{(1)}, \ldots, r^{(d+1)} \in \text{GF}(q)$, and consider the functions

$$f^{(j)}(x_1, \ldots, x_{i-1}) \stackrel{\text{def}}{=} f(x_1, \ldots, x_{i-1}, r^{(j)}, 0, \ldots, 0) - p(x_1, \ldots, x_{i-1}). \tag{2.4}$$

Suppose that $f$ equals some degree $d$ polynomial and that $p$ is indeed the $(i-1)$-prefix of this polynomial. Then $f^{(j)}$ is a polynomial of degree $d-1$ (since all the degree $d$ monomials in the first $i$ variables have been canceled by $p$). Furthermore, *given an explicit representation* of $f^{(1)}, \ldots, f^{(d+1)}$, we can find (by interpolation) the extension of $p$ to a $i$-prefix. The last assertion deserves some elaboration.

Consider the $i$-prefix of $f$, denoted $p' = p'(x_1, \dots, x_{i-1}, x_i)$. In each $f^{(j)}$, the monomials of $p'$ that agree on the exponents of $x_1, \dots, x_{i-1}$ are collapsed together (since $x_i$ is instantiated and so monomials containing different powers of $x_i$ are added together). However, using the $d+1$ collapsed values, we can retrieve the coefficients of the different monomials (in $p'$). That is, for each sequence of exponents $(e_1, \dots, e_{i-1})$ such that $\sum_{j=1}^{i-1} e_j \leq d$, we retrieve the coefficients of all the $(\prod_{j=1}^{i-1} x^{e_j}) \cdot x_i^k$ in $p'$, by interpolation that refers to the coefficients of $\prod_{j=1}^{i-1} x^{e_j}$ in the $f^{(\ell)}$'s.[4]

To complete the high level description of the procedure we need to show how to obtain the polynomials representing the $f^{(j)}$'s. Since in reality we have only access to a (possibly highly noisy) oracle for the $f^{(j)}$'s, we use the main procedure for finding a list of candidates for these polynomials. We point out that the recursive call is to a problem of degree $d-1$, which is lower than the degree we are currently handling.

The above description ignores a real difficulty that may occur: Suppose that the agreement rate of $f$ with some $p^*$ is at least $\delta$, and so we need to recover $p^*$. For our strategy to work, the agreement rate of the $f^{(j)}$'s with $p^*(\dots, 0^{n-i})$ must be close to $\delta$. However, it may be the case that $p^*$ does not agree with $f$ at all on the inputs in $\mathrm{GF}(q)^i 0^{n-i}$, although $p^*$ does agrees with $f$ on a $\delta$ fraction of inputs in $\mathrm{GF}(q)^n$. Then solving the subproblem (i.e., trying to retrieve polynomials close to the $f^{(j)}$'s) gives us no information about $p^*$. Thus, we must make sure that the agreement rate on the subproblems on which we recurse is close to the original agreement rate. This can be achieved by applying a random linear transformation to the coordinate system as follows: Pick a random nonsingular matrix $R$ and define new variables $y_1, \dots, y_n$ as $(y_1, \dots, y_n) = \bar{y} \equiv R\bar{x}$ (each $y_i$ is a random linear combination of the $x_i$'s and vice versa). This transformation can be used to define a new instance of the reconstruction problem in terms of the $y_i$'s, and for the new instance the agreement rate on the subproblems on which we recurse is indeed close to the original agreement rate. Observe that

1. the total degree of the problem is preserved;
2. the points are mapped pairwise independently, and so the fraction of agreement points in all subspaces of the new problem is close to the agreement rate in the original space; and
3. one can easily transform the coordinate system back to the $x_i$'s, and so it is possible to construct a new black box consistent with $f$ that takes $\bar{y}$ as an input.

(It may be noted that the transformation does not preserve other properties of the polynomial, e.g., its sparsity.)

*Comment.* The above solution to the above difficulty is different than the one in the original version of this paper [18]. The solution there was to pick many different suffixes (instead of $0^{n-i}$) and to argue that at least in one of them the agreement rate is preserved. However, picking many different suffixes creates additional problems, which needed to be dealt with carefully. This resulted in a more complicated algorithm in the original version.

## 3. Algorithm for degree $d > 1$ polynomials.

Recall that we are given oracle access to a function $f : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$, and need to find all polynomials of degree $d$ that agrees with $f$ on at least a $\delta$ fraction of the inputs.

[4]Let $c_k$ be the coefficient of $(\prod_{j=1}^{i-1} x^{e_j}) \cdot x_i^k$ in $p'$, and $v_\ell$ be the coefficient of $\prod_{j=1}^{i-1} x^{e_j}$ in $f^{(\ell)}$. Then, $v_\ell = \sum_{k=0}^{d} (r^{(\ell)})^k c_k$, and the $c_k$'s can be found given the $v_\ell$'s.

The main algorithm Find-all-poly will use several subroutines: Compute-coefficients, Test-valid, Constants, Brute-force, and Extend. The main algorithm is recursive, in $n$ (the number of variables) and $d$ (the degree), with the base case $d = 0$ being handled by the subroutine Constants and the other bases cases corresponding to $n \leq 4$ being handled by the subroutine Brute-force. Most of the work is done in Find-all-poly and Extend, which are mutually recursive.

The algorithms have a number of parameters in their input. We describe the commonly occurring parameters first:

- $q$ is the size of the field we will be working with; i.e., $F = \mathrm{GF(q)}$. (Unlike other parameters, the field never changes in the recursive calls.)
- $f$ will be a function from $\mathrm{GF(q)}^n$ to $\mathrm{GF(q)}$ given as an oracle to the current procedure, and $n$ will denote the number of variables of $f$.
- $d$ will denote the degree of the polynomial we are hoping to reconstruct, and $\delta$ will denote the agreement parameter. Typically, the algorithm will have to reconstruct all degree $d$ polynomials having agreement at least $\delta$ with $f$.

Many of the algorithms are probabilistic and make two-sided error.

- $\psi$ will be the error parameter controlling the probability with which a valid solution may be omitted from the output.
- $\phi$ will be the error parameter controlling the error with which an invalid solution is included in the output list.

Picking a random element of $\mathrm{GF(q)}$ is assumed to take unit time, as are field operations and calls to the oracle $f$.

The symbol $x$ will typically stand for a vector in $\mathrm{GF(q)}^n$, while the notation $x_i$ will refer to the $i$th coordinate of $x$. When picking a sequence of vectors, we will use superscripts to denote the vectors in the sequence. Thus, $x_i^{(j)}$ will denote the $i$th coordinate of the $j$th element of the sequence of vectors $x^{(1)}, x^{(2)}, \ldots$. For two polynomials $p_1$ and $p_2$, we write $p_1 \equiv p_2$ if $p_1$ and $p_2$ are identical. (In this paper, we restrict ourselves to polynomials of degree less than the field size; thus identity of polynomials as functions is equivalent to identity of polynomials as a formal sum of monomials.) We now generalize the notion of the prefix of a polynomial in two ways. We extend it to arbitrary functions, and then extend it to arbitrary suffixes (and not just $0^i$).

DEFINITION 3.1. *For $1 \leq i \leq n$ and $a_1, \ldots, a_{n-i} \in F$, the $(a_1, \ldots, a_{n-i})$-*prefix *of a function $f : F^n \to F$, denoted $f|_{a_1,\ldots,a_{n-i}}$, is the $i$-variate function $f|_{a_1,\ldots,a_{n-i}} : F^i \to F$, given by $f|_{a_1,\ldots,a_{n-i}}(x_1, \ldots, x_i) = f(x_1, \ldots, x_i, a_1, \ldots, a_{n-i})$. The $i$-*prefix *of $f$ is the function $f|_{0^{n-i}}$.*

When specialized to a polynomial $p$, the $i$-prefix of $p$ yields a polynomial on the variables $x_1, \ldots, x_i$ whose coefficients are exactly the coefficients of $p$ on monomials involving only $x_1, \ldots, x_i$.

Fixing a field $\mathrm{GF(q)}$, we will use the notation $N_{n,d,\delta}$ to denote the maximum (over all possible $f$) of the number of polynomials of degree $d$ in $n$ variables that have agreement $\delta$ with $f$. In this section we will first determine our running time as a function of $N_{n,d,\delta}$, and only next use bounds on $N_{n,d,\delta}$ (proven in section 4) to derive the absolute running times. We include the intermediate bounds since it is possible that the bounds of section 4 may be improved, and this would improve our running time as well. By definition, $N_{n,d,\delta}$ is monotone nondecreasing in $d$ and $n$ and monotone nonincreasing in $\delta$. These facts will be used in the analysis.

**3.1. The subroutines.** We first axiomatize the behavior of each of the subroutines. Next we present an implementation of the subroutine and analyze it with respect to the axiomatization.

(P1) Constants$(f, \delta, n, q, \psi)$, with probability at least $1-\psi$, returns every degree 0 (i.e., constant) polynomial $p$ such that $f$ and $p$ agree on $\delta$ fraction of the points.[5]

Constants works as follows: Set $k = O(\frac{1}{\delta^2}\log\frac{1}{\psi})$ and pick $x^{(1)}, \ldots, x^{(k)}$ independently and uniformly at random from $\mathrm{GF(q)}^n$. Output the list of all constants $a$ (or equivalently the polynomial $p_a = a$) such that $|\{i \in [k] | f(x^{(i)}) = a\}| \geq \frac{3}{4}\delta k$.

An easy application of Chernoff bounds indicates that the setting $k = O(\frac{1}{\delta^2}\log\frac{1}{\psi})$ suffices to ensure that the error probability is at most $\psi$. Thus the running time of Constants is bounded by the time to pick $x^{(1)}, \ldots, x^{(k)} \in \mathrm{GF(q)}^n$ which is $O(kn) = O(\frac{1}{\delta^2} n \log\frac{1}{\psi})$.

PROPOSITION 3.2. Constants$(f, \delta, n, q, \psi)$ *satisfies* (P1). *Its running time is* $O(\frac{1}{\delta^2} n \log\frac{1}{\psi})$.

Another simple procedure is the testing of agreement between a given polynomial and a black box.

(P2) Test-valid$(f, p, \delta, n, d, q, \psi, \phi)$ returns true, with probability at least $1-\psi$, if $p$ is an $n$-variate degree $d$ polynomial with agreement at least $\delta$ with $f$. It returns false with probability at least $1-\phi$ if the agreement between $f$ and $p$ is less than $\frac{\delta}{2}$. (It may return anything if the agreement is between $\frac{\delta}{2}$ and $\delta$.)

Test-valid works as follows: Set $k = O(\frac{1}{\delta^2}\log\frac{1}{\min\{\psi,\phi\}})$ and pick $x^{(1)}, \ldots, x^{(k)}$ independently and uniformly at random from $\mathrm{GF(q)}^n$. If $f(x^{(i)}) = p(x^{(i)})$ for at least $\frac{3}{4}\delta$ fraction of the values of $i \in [k]$, then output true else false.

Again an application of Chernoff bounds yields the correctness of Test-valid. The running time of Test-valid is bounded by the time to pick the $k$ points from $\mathrm{GF(q)}^n$ and the time to evaluate $p$ on them, which is $O(\frac{1}{\delta^2}(\log\frac{1}{\min\{\psi,\phi\}})\binom{n+d}{d})$.

PROPOSITION 3.3. Test-valid$(f, p, \delta, n, d, q, \psi, \phi)$ *satisfies* (P2). *Its running time is bounded by* $O(\frac{1}{\delta^2}(\log\frac{1}{\min\{\psi,\phi\}})\binom{n+d}{d})$.

Next we describe the properties of a "brute-force" algorithm for reconstructing polynomials.

(P3) Brute-force$(f, \delta, n, d, q, \psi, \phi)$ returns a list that includes, with probability $1-\psi$, every degree $d$ polynomial $p$ such that $f$ and $p$ agree on $\delta$ fraction of the points. With probability at least $1-\phi$ it does not output any polynomial $p$ whose agreement with $f$ is less than $\frac{\delta}{2}$.

Notice that the goal of Brute-force is what one would expect to be the goal of Find-all-poly. Its weakness will be its running time, which is doubly exponential in $n$ and exponential in $d$. However, we invoke it only for $n \leq 4$. In this case its running time is of the order of $\delta^{-d^4}$. The description of Brute-force is given in Figure 3.1.

LEMMA 3.4. Brute-force$(f, \delta, n, d, q, \psi, \phi)$ *satisfies* (P3). *Its running time is at most* $O(\frac{kl^3}{\delta^2}(\log\frac{k}{\phi}))$, *where* $l = \binom{n+d}{d}$ *and* $k = O((\delta - \frac{d}{q})^{-l}(\log\frac{1}{\psi}))$.

*Proof.* The running time of Brute-force is immediate from its description (using the fact that a naive interpolation algorithm for a (multivariate) polynomial with $l$ coefficients runs in time $O(l^3)$ and the fact that each call to Test-valid takes at most $O(\frac{l}{\delta^2}\log\frac{k}{\phi})$ time). If a polynomial $p$ that is the output of the multivariate

[5]Notice that we do not make any claims about the probability with which constants that do not have significant agreement with $f$ may be reported. In fact we do not need such a condition for our analysis. If required, such a condition may be explicitly enforced by "testing" every constant that is returned for sufficient agreement. Note also that the list is allowed to be empty if no polynomial has sufficiently large agreement.

```
Brute-force($f, \delta, n, d, q, \psi, \phi$)
  Set $l = \binom{n+d}{d}$
      $k = O\left((\delta - \frac{d}{q})^{-l}\left(\log \frac{1}{\psi}\right)\right)$
      $\mathcal{L} \leftarrow \lambda$.
  Repeat $k$ times
      Pick $x^{(1)}, \ldots, x^{(l)} \in_R \mathrm{GF(q)}^n$.
      Multivariate interpolation step:
        Find $p : \mathrm{GF(q)}^n \rightarrow \mathrm{GF(q)}$ of degree $d$ s.t.  $\forall i \in [l]$, $p(x^{(i)}) = f(x^{(i)})$.
      If Test-valid($f, p, \delta, n, d, q, \frac{1}{2}, \phi/k$) then $\mathcal{L} \leftarrow \mathcal{L} \cup \{p\}$.
  endRepeat
return($\mathcal{L}$)
```

FIG. 3.1. Brute-force.

interpolation step has agreement less than $\frac{\delta}{2}$ with $f$, then by the correctness of Test-valid it follows that $p$ is passed with probability at most $\phi/k$. Summing up over the $k$ iterations, we have that the probability that any polynomial with agreement less than $\frac{\delta}{2}$ is included in the output list is at most $\phi$.

To prove that with probability at least $1-\psi$, Test-valid outputs every polynomial $p$ with $\delta$ agreement $f$, let us fix $p$ and argue that in any one of the $k$ iterations, $p$ is likely to be added to the output list with probability $\zeta = \frac{1}{2(\delta - \frac{d}{q})^l}$. The lemma follows from the fact that the number of iterations is a sufficiently large multiple of $\frac{1}{\zeta}$.

To prove that with probability at least $\zeta$ the polynomial $p$ is added to $\mathcal{L}$ (in a single iteration), we show that with probability at least $2\zeta$ the polynomial interpolated in the iteration equals $p$. The lemma follows from the fact that Test-valid will return true with probability at least $\frac{1}{2}$.

To show that $p$ is the polynomial returned in the interpolation step, we look at the task of finding $p$ as the task of solving a linear system. Let $\vec{p}$ denote the $l$ dimensional vector corresponding to the coefficients of $p$. Let $M$ be the $l \times l$ dimensional matrix whose rows correspond to the points $x^{(1)}, \ldots, x^{(l)}$ and whose columns correspond to the monomials in $p$. Specifically, the entry $M_{i,j}$, where $j$ corresponds to the monomial $x_1^{d_1} \ldots x_n^{d_n}$, is given by $(x_1^{(i)})^{d_1} \ldots (x_n^{(i)})^{d_n}$. Finally let $\vec{f}$ be the vector $(f(x^{(1)}), \ldots, f(x^{(l)}))$. To show that $p$ is the polynomial returned in this step, we show that with high probability, $M$ is of full rank and $p(x^{(i)}) = f(x^{(i)})$ for every $i$.

The last assertion is proven by induction on $i$. Let $M^{(i)}$ denote the $i \times l$ matrix with the first $i$ rows of $M$. Fix $x^{(1)}, \ldots, x^{(i-1)}$ such that $p(x^{(j)}) = f(x^{(j)})$ for every $j \in [i-1]$. We argue that with probability at least $\delta - \frac{d}{q}$ over the choice of $x^{(i)}$, it holds that $p(x^{(i)}) = f(x^{(i)})$ and the rank of $M^{(i)}$ is greater than that of $M^{(i-1)}$. It is easy to see that $f(x^{(i)}) = p(x^{(i)})$ with probability at least $\delta$. To complete the proof it suffices to establish that the probability, over a random choice of $x^{(i)}$, that $M^{(i)}$ has the same rank as $M^{(i-1)}$ is at most $\frac{d}{q}$. Consider two polynomials $p_1$ and $p_2$ such that $p_1(x^{(j)}) = p_2(x^{(j)})$ for every $j \in [i-1]$. Then for the rank of $M^{(i)}$ to be the same as the rank of $M^{(i-1)}$, it must be that $p_1(x^{(i)}) = p_2(x^{(i)})$ (else the solutions to the $i$th system are not the same as the solutions to the $i-1$th system). But for distinct polynomials $p_1$ and $p_2$ the event $p_1(x^{(i)}) = p_2(x^{(i)})$ happens with probability at most $\frac{d}{q}$ for randomly chosen $x^{(i)}$. This concludes the proof of the lemma. ∎

As an extension of univariate interpolations, we have the following.

(P4) Compute-coefficients$(p^{(1)},\dots,p^{(d+1)},r^{(1)},\dots,r^{(d+1)},n,d,q,\psi)$ takes as input $d+1$ polynomials $p^{(j)}$ in $n-1$ variables of degree $d-1$ and $d+1$ values $r^{(j)} \in \mathrm{GF(q)}$ and returns a degree $d$ polynomial $p : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$ such that $p|_{r^{(j)}} \equiv p^{(j)}$ for every $j \in [d+1]$, if such a polynomial $p$ exists (otherwise it may return anything).

Compute-coefficients works as a simple interpolation algorithm: Specifically it finds $d+1$ univariate polynomials $h_1,\dots,h_{d+1}$ such that $h_i(r^{(j)})$ equals 1 if $i=j$ and 0 otherwise and then returns the polynomial $p(x_1,\dots,x_n) = \sum_{j=1}^{d+1} h_j(x_n)\cdot p^{(j)}(x_1,\dots,x_{n-1})$. Note that indeed

$$\begin{aligned} p(x_1,\dots,x_{n-1},r^{(j)}) &= \sum_{k=1}^{d+1} h_k(x_n)\cdot p^{(k)}(x_1,\dots,x_{n-1}) \\ &= p^{(j)}(x_1,\dots,x_{n-1}). \end{aligned}$$

Note that the polynomials $h_i(x) = \prod_{j\in\{1,\dots,d+1\},j\neq i}(\frac{x-r^{(j)}}{r^{(i)}-r^{(j)}})$ depend only on the $r^{(j)}$'s. (Thus, it suffices to compute them once, rather than computing them from scratch for each monomial of $p$ as suggested in section 2.2.)

PROPOSITION 3.5. Compute-coefficients$(p^{(1)},\dots,p^{(d+1)},r^{(1)},\dots,r^{(d+1)},n,d,q,\psi)$ *satisfies* (P4). *Its running time is* $O(d^2\binom{n+d}{d})$.

**3.2. The main routines.** As mentioned earlier, the main subroutines are Find-all-poly and Extend, whose inputs and properties are described next. They take, among other inputs, a special parameter $\alpha$ which will be fixed later. For the sake of simplicity, we do not require Find-all-poly and Extend at this point to output only polynomials with good agreement. We will consider this issue later when analyzing the running times of Find-all-poly and Extend.

(P5) Find-all-poly$(f,\delta,n,d,q,\psi,\phi,\alpha)$ returns a list of polynomials containing every polynomial of degree $d$ on $n$ variables that agrees with $f$ on at least a $\delta$ fraction of the inputs. Specifically, the output list contains every degree $d$ polynomial $p$ with agreement $\delta$ with $f$, with probability at least $1-\psi$.

The algorithm is described formally in Figure 3.2. Informally, the algorithm uses the ("trivial") subroutines for the base cases $n \le 4$ or $d = 0$, and in the remaining (interesting) cases it iterates a randomized process several times. Each iteration is initiated by a random linear transformation of the coordinates. Then in this new coordinate system, Find-all-poly finds (using the "trivial" subroutine Brute-force) a list of all 4-variate polynomials having significant agreement with the 4-prefix of the oracle.[6] It then extends each polynomial in the list one variable at a time till it finds the $n$-prefix of the polynomial (which is the polynomial itself). Thus the crucial piece of the work is relegated to the subroutine Extend, which is supposed to extend a given $(i-1)$-prefix of a polynomial with significant agreement with $f$ to its $i$-prefix. The goals of Extend are described next.

[6]In principle we could apply Brute-force for any constant number of variables (and not just 4). However, since the running time is doubly-exponential in the number of variables, we try to use Brute-force only for a small number of variables. The need for using Brute-force when the number of variables is very small comes about due to the fact that in such a case (e.g., two variables) the randomization of the coordinate system does not operate well. Furthermore, applying Brute-force for univariate polynomials seems unavoidable. For simplicity of exposition, we choose to apply Brute-force also for 2-, 3-, and 4-variate polynomials. This allows better settings of some parameters and simplifies the calculations at the end of the proof of Lemma 3.6.

To simplify our notation, from here onwards we assume that the elements of the field are named $0, \ldots, q-1$. In particular, we often use $p|_j$, for some integer $0 \leq j \leq d$, to represent the $(j)$-prefix of a function $p$.

(P6) Extend$(f, p, \delta, n, d, q, \psi, \phi, \alpha)$ takes as input a degree $d$ polynomial $p$ in $n-1$ variables and, with probability at least $1-\psi$, returns a list of degree $d$ polynomials in $n$ variables that includes every polynomial $p^*$ that satisfies the following conditions:

1. $p^*$ has agreement at least $\delta$ with $f$.
2. $p^*|_j$ has agreement at least $\alpha \cdot \delta$ with $f|_j$ for every $j \in \{0, \ldots, d\}$.
3. $p^*|_0 \equiv p$.

Figure 3.3 describes the algorithm formally. Extend returns all $n$-variable extensions $p^*$, of a given $(n-1)$-variable polynomial $p$, provided $p^*$ agrees with $f$ in a strong sense: $p^*$ has significant agreement with $f$ and each $p^*|_j$ has significant agreement with $f|_j$ (for every $j \in \{0, \ldots, d\}$). (The latter agreement requirement is slightly lower than the former.) To recover $p^*$, Extend first invokes Find-all-poly to find the polynomials $p^*|_j$ for $d+1$ values of $j$. This is feasible only if a polynomial $p^*|_j$ has good agreement with $f|_j$, for every $j \in \{0, \ldots, d\}$. Thus, it is crucial that when Extend is called with $f$ and $p$, all $p^*$'s with good agreement with $f$ also satisfy the stronger agreement property (above). We will show that the calling program (i.e., Find-all-poly at the higher level of recursion) will, with high probability, satisfy this property, by virtue of the random linear transformation of coordinates.

All the recursive calls (of Find-all-poly within Extend) always involve a smaller degree parameter, thereby ensuring that the algorithms terminate (quickly). Having found a list of possible values of $p^*|_j$, Extend uses a simple interpolation (subroutine Compute-coefficients) to find a candidate for $p^*$. It then uses Test-valid to prune out the many invalid polynomials that are generated this way, returning only polynomials that are close to $f$.

We now go on to the formal analysis of the correctness of Find-all-poly and Extend.

## 3.3. Correctness of Find-all-poly and Extend.

LEMMA 3.6. *If $\alpha \leq 1 - \frac{1}{q}$, $\delta \geq \frac{d+1}{q}$, and $q \geq 3$, then* Find-all-poly *satisfies* (P5) *and* Extend *satisfies* (P6).

*Proof.* We prove the lemma by a double induction, first on $d$ and for any fixed $d$, we perform induction on $n$. We shall rely on the properties of Compute-coefficients, Test-valid, Constants, and Brute-force, as established above.

Assume that Find-all-poly is correct for every $d' < d$ (for every $n' \leq n$ for any such $d'$). We use this to establish the correctness of Extend$(f, p, n', d, q, \psi, \alpha)$ for every $n' \leq n$. Fix a polynomial $p^*$ satisfying the hypothesis in (P6). We will prove that $p^*$ is in the output list with probability $1 - \frac{\psi}{N_{n',d,\delta}}$. The correctness of Extend follows from the fact that there are at most $N_{n',d,\delta}$ such polynomials $p^*$ and the probability that there exists one for which the condition is violated is at most $\psi$.

To see that $p^*$ is part of the output list, notice that, by the inductive hypothesis on Find-all-poly, when invoked with agreement parameter $\alpha \cdot \delta$, it follows that for any fixed $j \in \{0, \ldots, d\}$, the polynomial $p^*|_j - p$ is included in $\mathcal{L}^{(j)}$ with probability $1 - \frac{\psi}{2(d+1)N_{n',d,\delta}}$. This follows from the fact that $p^*|_j - p$ and $f|_j - p$ have agreement at least $\alpha \cdot \delta$, the fact that $p^*|_j - p = p^*|_j - p^*|_0$ is a degree $d-1$ polynomial,[7] and thus, by

[7]To see that $p^*|_j - p^*|_0$ is a polynomial of total degree at most $d-1$, notice that $p^*(x_1, \ldots, x_n)$ can be expressed uniquely as $r(x_1, \ldots, x_{n-1}) + x_n q(x_1, \ldots, x_n)$, where degree of $q$ is at most $d-1$. Thus $p^*|_j - p^*|_0 = j \cdot q(x_1, \ldots, x_{n-1}, j)$ is also of degree $d-1$.

```
Find-all-poly($f, \delta, n, d, q, \psi, \phi, \alpha$);
If $d = 0$ return(Constants($f, \delta, n, q, \psi$));
If $n \le 4$ return(Brute-force($f, \delta, n, d, q, \psi, \phi$));

$\mathcal{L} \leftarrow \{\}$;
Repeat $O(\log \frac{N_{n,d,\delta}}{\psi})$ times:
  Pick a random nonsingular $n \times n$ matrix $R$ over $\mathrm{GF(q)}$
  Pick a random vector $b \in \mathrm{GF(q)}^n$.
  Let $g$ denote the oracle given by $g(y) = f(R^{-1}(y - b))$.
  $\mathcal{L}_4 \leftarrow$ Brute-force($g|_{0^{n-4}}, \delta, 4, d, q, \frac{1}{10n}, \phi$).

  for $i = 5$ to $n$ do
    $\mathcal{L}_i \leftarrow \{\}$ /* List of $(d, i)$-prefixes */
    for every polynomial $p \in \mathcal{L}_{i-1}$ do
      $\mathcal{L}_i = \mathcal{L}_i \cup$ Extend($g|_{0^{n-i}}, p, \delta, i, d, q, \frac{1}{10n}, \phi, \alpha$)
    endfor
  endfor

  Untransform $\mathcal{L}_n$:   $\mathcal{L}'_n \leftarrow \{p'(x) \overset{\mathrm{def}}{=} p(Rx + b) | p \in \mathcal{L}_n\}$.
  $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{L}'_n$.
endRepeat
return($\mathcal{L}$)
```

FIG. 3.2. Find-all-poly.

```
Extend($f, \delta, p, n, d, q, \psi, \phi, \alpha$).

  $\mathcal{L}' \leftarrow \{\}$.
  $\mathcal{L}^{(0)} \leftarrow \{\bar{0}\}$ (where $\bar{0}$ is the constant $0$ polynomial).
  for $j = 1$ to $d$ do
      $f^{(j)} \leftarrow f|_j - p$.
      $\mathcal{L}^{(j)} \leftarrow$Find-all-poly($f^{(j)}, \alpha \cdot \delta, n, d-1, q, \frac{\psi}{2N_{n,d,\alpha\cdot\delta}(d+1)}, \phi, \alpha$).
  endfor

  for every $(d+1)$-tuple $(p^{(0)}, \ldots, p^{(d)})$ with $p^{(k)} \in \mathcal{L}^{(k)}$ do
    $p' \leftarrow$ Compute-coefficients($p^{(0)}, \ldots, p^{(d)}, 0, \ldots, d;\ n, d, q$).
    if Test-valid($f, p + p', \delta, n, d, q, \psi/(2N_{n,d,\alpha\cdot\delta}), \phi$) then
      $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{p + p'\}$;
  endfor
  return($\mathcal{L}'$).
```

FIG. 3.3. Extend.

the inductive hypothesis on the correctness of Find-all-poly, such a polynomial should be in the output list. By the union bound, we have that for every $j \in \{0, \ldots, d\}$, the polynomial $p^*|_j - p$ is included in $\mathcal{L}^{(j)}$ with probability $1 - \frac{\psi}{2N_{n',d,\alpha\cdot\delta}}$, and in such a case $p^* - p$ will be one of the polynomials returned by an invocation of Compute-coefficients. In such a case $p^*$ will be tested by Test-valid and accepted with probability at least

$1 - \frac{\psi}{2N_{n',d,\alpha\cdot\delta}}$. Again summing up all the error probabilities, we have that $p^*$ is in the output list with probability at least $1 - \frac{\psi}{N_{n',d,\alpha\cdot\delta}}$. This concludes the correctness of Extend.

We now move on to the correctness of Find-all-poly$(f, \delta, n, d, q, \psi, \phi, \alpha)$. Here we will try to establish that for a fixed polynomial $p$ with agreement $\delta$ with $f$, the polynomial $p$ is added to the list $\mathcal{L}$ with constant probability in each iteration of the Repeat loop. Thus the probability that it is not added in any of the iterations is at most $\frac{\psi}{N_{n,d,\delta}}$ and thus the probability that there exists a polynomial that is not added in any iteration is at most $\psi$. We may assume that $n \geq 5$ and $d \geq 1$ (or else correctness is guaranteed by the trivial subroutines).

Fix a degree $d$ polynomial $p$ with agreement $\delta$ with the function $f : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$. We first argue that $(R, b)$ form a "good" linear transformation with constant probability. Recall that from now onwards Find-all-poly works with the oracle $g : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$ given by $g(y) = f(R^{-1}(y - b))$. Analogously define $p'(y) = p(R^{-1}(y - b))$, and notice $p'$ is also a polynomial of degree $d$. For any $i \in \{5, \ldots, n\}$ and $j \in \{0, \ldots, d\}$, we say that $(R, b)$ is *good for* $(i, j)$ if the agreement between $g|_{j,0^{n-i}}$ and $p'|_{j,0^{n-i}}$ is at least $\alpha\delta$. Lemma 3.7 (below) shows that the probability that $(R, b)$ is good for $(i, j)$ with probability at least $1 - \frac{1}{q^{i-1}} \cdot (2 + \frac{1}{\delta(1-\alpha)^2})$. Now call $(R, b)$ *good* if it is good for every pair $(i, j)$, where $i \in \{5, \ldots, n\}$ and $j \in \{0, \ldots, d\}$. Summing up the probabilities that $(R, b)$ is not good for $(i, j)$ we find that $(R, b)$ is not good with probability at most

$$\begin{aligned}
&\sum_{j=0}^{d}\sum_{i=5}^{n}\left(2 + \frac{1}{\delta(1-\alpha)^2}\right)\cdot q^{-i+1} \\
&= (d+1)\cdot\left(2 + \frac{1}{\delta(1-\alpha)^2}\right)\cdot\sum_{i=5}^{n} q^{-i+1} \\
&< (d+1)\cdot\left(2 + \frac{1}{\delta(1-\alpha)^2}\right)\cdot\frac{q^{-3}}{q-1} \\
&\leq \frac{2}{q^2(q-1)} + \frac{1}{q-1} \quad \left(\text{using } \alpha \leq 1 - \tfrac{1}{q},\ \delta \geq \tfrac{d+1}{q}, \text{ and } d+1 \leq q\right) \\
&\leq \frac{11}{18} \quad (\text{using } q \geq 3).
\end{aligned}$$

Conditioned upon $(R, b)$ being good and relying on the property of Brute-force, it follows that $\mathcal{L}_4$ contains the 4-prefix of $p$ with probability at least $1 - \frac{1}{10n}$. Inductively, we have that the $i$-prefix of $p$ is not contained in the list $\mathcal{L}_i$ with probability at most $\frac{i}{10n}$. (By the inductive hypothesis on Extend, with probability at most $\frac{1}{10n}$, the $(i-1)$-prefix of $p$ is in $\mathcal{L}_{i-1}$ and yet the $i$-prefix is not returned by Extend.) Thus, with probability at most $\frac{1}{10}$, the polynomial $p$ is not included in $\mathcal{L}_n$ (conditioned upon $(R, b)$ being good). Adding back the probability that $(R, b)$ is not good, we conclude that with probability at most $\frac{11}{18} + \frac{1}{10} < \frac{3}{4}$, the polynomial $p$ is not in $\mathcal{L}_n$ in any single iteration. This concludes the proof of the correctness of Find-all-poly. ∎

**3.4. Analysis of the random linear transformation.** We now fill in the missing lemma establishing the probability of the "goodness" of a random linear transformation.

LEMMA 3.7. *Let $f$ and $g$ be functions mapping* $\mathrm{GF(q)}^n$ *to* $\mathrm{GF(q)}$ *that have $\delta$ agreement with each other, and let $R$ be a random nonsingular $n \times n$ matrix and $b$ be* *a random element of* $\mathrm{GF(q)}^n$*. Then, for every* $i \in \{1, \ldots, n\}$ *and* $j \in \mathrm{GF(q)}$

$$\Pr_{R,b}\left[f'|_{j,0^{n-i}} \textit{ and } g'|_{j,0^{n-i}} \textit{ have less than } \alpha\delta \textit{ agreement}\right] \le \frac{1}{q^{i-1}} \cdot \left(2 + \frac{1}{\delta(1-\alpha)^2}\right),$$

*where* $f'(y) = f(R^{-1}(y-b))$ *and* $g'(y) = g(R^{-1}(y-b))$*.*

*Proof.* Let $G = \{x \in \mathrm{GF(q)}^n | f(x) = g(x)\}$ be the set of "good" points. Observe that $\delta = |G|/q^n$. Let $S_{R,b} = \{x \in \mathrm{GF(q)}^n | Rx + b \text{ has } j0^{n-i} \text{ as suffix}\}$. Then we wish to show that

$$\Pr_{R,b}\left[\frac{|S_{R,b} \cap G|}{|S_{R,b}|} < \alpha \cdot \frac{|G|}{q^n}\right] \le \frac{1}{q^{i-1}}\left(2 + \frac{1}{\delta(1-\alpha)^2}\right). \tag{3.1}$$

Observe that the set $S_{R,b}$ can be expressed as the preimage of $(j, 0^{n-i})$ in the map $\pi : \mathrm{GF(q)}^n \to \mathrm{GF(q)}^m$, where $m = n - i + 1$, given by $\pi(x) = R'x + b'$, where $R'$ is the $m \times n$ matrix obtained by taking the bottom $m$ rows of $R$ and $b'$ is the vector obtained by taking the last $m$ elements of $b$. Note that $R'$ is a uniformly distributed $m \times n$ matrix of full rank over GF(q) and $b'$ is just a uniformly distributed $m$ dimensional vector over GF(q). We first analyze what happens when one drops the full-rank condition on $R'$.

CLAIM 3.8. *Let* $R'$ *be a random* $m \times n$ *matrix over* GF(q) *and* $b'$ *be a random element of* $\mathrm{GF(q)}^m$*. For some fixed vector* $\vec{s} \in \mathrm{GF(q)}^m$ *let* $S = \{x | R'x + b' = \vec{s}\}$*. Then, for any set* $G \subseteq \mathrm{GF(q)}^n$*,*

$$\Pr_{R',b'}\left[\frac{|S \cap G|}{|S|} < \alpha \cdot \frac{|G|}{q^n}\right] \le \frac{q^m}{(1-\alpha)^2|G|} + q^{-(n-m)}.$$

*Proof.* We rewrite the probability in the claim as

$$\begin{aligned}
&\Pr_{R',b'}\left[|S \cap G| < \alpha \cdot \frac{|G| \cdot |S|}{q^n}\right] \\
&\le \Pr_{R',b'}\left[|S \cap G| < \alpha \cdot \frac{|G| \cdot q^{n-m}}{q^n} \quad \text{or} \quad |S| > q^{n-m}\right] \\
&\le \Pr_{R',b'}\left[|S \cap G| < \alpha \cdot \frac{|G|}{q^m}\right] \; + \; \Pr_{R',b'}\left[|S| > q^{n-m}\right].
\end{aligned}$$

The event in the second term occurs only if the matrix $R'$ is not full rank, and so the second term is bounded by $q^{-(n-m)}$ (see Claim 3.9). We thus focus on the first term.

For $x \in G \subseteq \mathrm{GF(q)}^n$, let $I(x)$ denote an indicator random variable that is 1 if $x \in S$ (i.e., $R'x + b' = \vec{s}$) and 0 otherwise. Then, the expected value of $I(x)$, over the choice of $(R', b')$, is $q^{-m}$. Furthermore, the random variables $I(x_1)$ and $I(x_2)$ are independent, for any distinct $x_1$ and $x_2$. Now, $|S \cap G| = \sum_{x \in G} I(x)$, and we are interested in the probability that the sum $\sum_{x \in G} I(x)$ is smaller than $\alpha \cdot |G| \cdot q^{-m}$ (whereas the expected value of the sum is $|G| \cdot q^{-m}$). A standard application of Chebyschev's inequality yields the desired bound.[8] ∎

To fill the gap caused by the "full-rank clause" (in the above discussion), we use the following claim.

[8]Specifically, we obtain a probability bound of $\frac{|G| \cdot q^{-m}}{((1-\alpha)\cdot(|G| \cdot q^{-m}))^2} = \frac{q^m}{(1-\alpha)^2 \cdot |G|}$ as required.

CLAIM 3.9. *The probability that a randomly chosen* $m \times n$ *matrix over* GF(q) *is not of full rank is at most* $q^{-(n-m)}$.

*Proof.* We can consider the matrix as being chosen one row at a time. The probability that the $j$th row is dependent on the previous $j-1$ rows is at most $q^{j-1}/q^n$. Summing up over $j$ going from 1 to $m$ we get that the probability of getting a matrix not of full rank is at most $q^{-(n-m)}$. □

Finally we establish (3.1). Let $E_{R',b'}$ denote the event that $\frac{|S\cap G|}{|S|} < \alpha \cdot \frac{|G|}{q^n}$ (recall that $S = S_{R',b'}$) and let $F_{R',b'}$ denote the event that $R'$ is of full row rank. Then considering the space of uniformly chosen matrices $R'$ and uniformly chosen vectors $b'$ we are interested in the quantity

$$\begin{aligned}\Pr_{R',b'}[E_{R',b'}|F_{R',b'}] &\leq \Pr_{R',b'}[E_{R',b'}] + \Pr_{R',b'}[\neg(F_{R',b'})]\\ &\leq \frac{q^m}{(1-\alpha)^2|G|} + 2\cdot q^{-(n-m)}.\end{aligned}$$

The lemma follows by substituting $m = n-i+1$ and $|G| = \delta \cdot 2^n$. □

### 3.5. Analysis of the running time of Find-all-poly.

LEMMA 3.10. *For integers* $d_0, n_0, q$, *and* $\alpha, \delta_0 \in [0,1]$ *satisfying* $\alpha^{d_0}\delta_0 \geq 2d_0/q$, *let* $M = \max_{0\leq d\leq d_0}\{N_{n_0,d,(\alpha^{d_0-d})\cdot(\delta_0/2)}\}$. *Then, with probability* $1-\phi\cdot(n_0^2(d_0+1)^2 M\log M)^{d_0+1}\cdot\log(1/\psi_0)$, *the running time of* Find-all-poly$(f,\delta_0,n_0,d_0,q,\psi_0,\phi,\alpha)$ *is bounded by a polynomial in* $M^{d_0+1}$, $(n_0+d_0)^{d_0}$, $(\frac{1}{\alpha^{d_0}\delta_0})^{(d_0+4)^4}$, $\log\frac{1}{\psi_0}$, *and* $\log\frac{1}{\phi}$.

*Proof.* We fix $n_0$ and $d_0$. Observe that in all recursive calls to Find-all-poly, $\delta$ and $d$ are related by the invariant $\delta = \alpha^{d_0-d}\delta_0$. Now, assuming the algorithms run correctly, they should return only polynomials with agreement at least $\delta/2$ (which motivates the quantity $M$). Further, in all such calls, we have that $\alpha^{d_0}\delta_0 - \frac{d}{q} \geq \alpha^{d_0}\delta_0/2$. Observe further that the parameter $\phi$ never changes and the parameter $\psi$ affects only the number of iterations of the outermost call to Find-all-poly. In all other calls, this parameter (i.e., $\psi$) is at least $\psi_1 \overset{\text{def}}{=} \frac{1}{20n_0(d_0+1)M}$. Assume for simplicity that $\psi_0 \leq \psi_1$. Let $T_1, T_2, T_3$, and $T_4$ denote the maximum running time of any of the subroutine calls to Constants, Test-valid, Brute-force, and Compute-coefficients, respectively. Let $T = \max\{T_1, T_2, T_3, T_4\}$. Then

$$\begin{aligned}T_1 &= O\left(\frac{n}{\alpha^{2d_0}\delta_0^2}\cdot\log\frac{1}{\psi_0}\right),\\ T_2 &= O\left(\frac{1}{\alpha^{2d_0}\delta_0^2}\cdot\binom{n_0+d_0}{d_0}\cdot\log\frac{1}{\min\{\psi_0,\phi\}}\right),\\ T_3 &= O\left(\frac{kl^3}{(\alpha^{d_0}\delta_0/2)^2}\cdot\log\frac{k}{\phi}\right),\\ &\qquad\text{where } l = O((d_0+4)^4) \text{ and } k = O\left(\alpha^{-(d_0+4)^4}\cdot(\delta_0/2)^{-(d_0+4)^4}\cdot\log\tfrac{1}{\psi_0}\right),\\ T_4 &= O\left(d_0^2\cdot\binom{n_0+d_0}{d_0}\right).\end{aligned}$$

Note that all the above quantities are upper bounded by polynomials in $(n_0+d_0)^{d_0}$, $(\frac{2}{\alpha^{d_0}\delta_0})^{(d_0+4)^4}$, $\log M$, $\log\frac{1}{\phi}$, and thus so is $T$. In what follows we show that the running time is bounded by some polynomial in $(n_0 d_0 M)^{(d_0+1)}$ and $T$ and this will suffice to prove the lemma.

Let $P(d)$ denote an upper bound on the probability that any of the recursive calls made to $\mathsf{Find\text{-}all\text{-}poly}$ by $\mathsf{Find\text{-}all\text{-}poly}(f, \alpha^{d_0-d}\delta_0, n, d, q, \psi, \phi, \alpha)$ returns a list of length greater than $M$, maximized over $f$, $1 \le n \le n_0$, $\psi \ge \psi_0$. Let $F(d)$ denote an upper bound on the running time on $\mathsf{Find\text{-}all\text{-}poly}(f, \alpha^{d_0-d}\delta_0, n, d, q, \psi, \phi, \alpha)$, conditioned upon the event that no recursive call returns a list of length greater than $M$. Similarly let $E(d)$ denote an upper bound on the running time of $\mathsf{Extend}$, under the same condition.

We first derive recurrences for $P$. Notice that the subroutine $\mathsf{Constants}$ never returns a list of length greater than $\frac{2}{\alpha^{d_0}\delta_0}$ (every constant output must have a fraction of $\frac{\alpha^{d_0}\delta_0}{2}$ representation in the sampled points). Thus $P(0) = 0$. To bound $P(d)$ in other cases, we observe that every iteration of the Repeat loop in $\mathsf{Find\text{-}all\text{-}poly}$ contributes an error probability of at most $\phi$ from the call to $\mathsf{Brute\text{-}force}$, and at most $n_0 - 4$ times the probability that $\mathsf{Extend}$ returns an invalid polynomial (i.e., a polynomial with agreement less than $\delta_d/2$ with its input function $f$). The probability that $\mathsf{Extend}$ returns such an invalid polynomial is bounded by the sum of $(d+1)\cdot P(d-1)$ (from the recursive calls to $\mathsf{Find\text{-}all\text{-}poly}$) and $M^{d+1}\cdot\phi$ (from the calls to $\mathsf{Test\text{-}valid}$). (Notice that to get the final bound we use the fact that we estimate this probability only when previous calls do not produce too long a list.) Finally the number of iterations of the Repeat loop in $\mathsf{Find\text{-}all\text{-}poly}$ is at most $\log(M/\psi)$, by the definition of $M$. Recall that in the outer most call of $\mathsf{Find\text{-}all\text{-}poly}$, we have $\psi = \psi_0$ whereas in all other calls $\psi \ge \psi_1$, where $\log(1/\psi_1) = \log(20n_0(d_0+1)M) < n_0(d_0+1)\log M$, for sufficiently large $n_0$. Thus summing up all the error probabilities, we have

$$P(d) \;<\; \log(M/\psi)\cdot n_0 \cdot \big((d+1)\cdot P(d-1) + M^{d+1}\cdot\phi\big)\,,$$

where for $d = d_0$ we use $\psi = \psi_0$ and otherwise $\psi = \psi_1$. It follows that

$$\begin{aligned} P(d_0) &< \log(M/\psi_0)\cdot n_0 \cdot \big((d_0+1)\cdot P(d_0-1) + M^{d+1}\cdot\phi\big) \\ &< \log(M/\psi_0)\cdot n_0\cdot(d_0+1)\cdot\big((n_0\cdot(d_0+1))^2\log M\big)^{d_0}\cdot M^{d+1}\cdot\phi \\ &< \big(n_0^2\cdot(d_0+1)^2\cdot M\log M\big)^{d_0+1}\cdot\phi\cdot\log(1/\psi_0). \end{aligned}$$

A similar analysis for $F$ and $E$ yields the following recurrences:

$$\begin{aligned} &F(0) \le T, \\ &F(d) \le n_0^2(d_0+1)(\log M)\cdot E(d), \\ &E(d) \le (d+1)F(d-1) + M^{d+1}T. \end{aligned}$$

Solving the recurrence yields $F(d) \le (n_0^2(d_0+1)^2 M\log M)^{d+1}T$. This concludes the proof of the lemma. □

LEMMA 3.11. *For integers $d_0, n_0$, and $\alpha, \delta_0 \in [0,1]$, let*

$$M = \max_{0\le d\le d_0}\{N_{n_0,d,(\alpha^{d_0-d})\cdot(\delta_0/2)}\}.$$

*If $\alpha \ge 1 - \frac{1}{d_0+1}$ and $\delta_0 \ge 2e\sqrt{\frac{d_0}{q}}$, then $M \le O(\frac{1}{\delta_0^2})$.*

*Proof.* We use part 2 of Theorem 4.4, which claims that $N_{n,d,\delta} \le \frac{1}{\delta^2-(d/q)}$, provided $\delta^2 \ge d/q$. Let $\delta_d = \alpha^{d_0-d}\delta_0$. Then $\delta_d/2 \ge (1-\frac{1}{d_0+1})^{d_0+1}\cdot(\delta_0/2) \ge \delta_0/2e \ge \sqrt{d/q}$, by the condition in the lemma. Thus $M$ is at most $\frac{1}{\delta_d^2-(d/q)} \le \frac{2}{\delta_d^2} = O(\frac{1}{\delta_0^2})$. □

THEOREM 3.12. *Given oracle access to a function $f$ and suppose $\delta, k, d,$ and $q$ are parameters satisfying $\delta \geq \max\{\frac{d+1}{q}, 2e\sqrt{d/q}\}$ and $q \geq 3$. Let $\alpha = 1 - \frac{1}{d+1}$, $\psi = 2^{-k}$, and $\phi = 2^{-k} \cdot (n(d+1)\frac{1}{\delta_0^2})^{-2(d+1)}$. Then, given oracle access to a function $f : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$, the algorithm* Find-all-poly$(f, \delta, n, d, q, \psi, \phi, \alpha)$ *runs in* $\mathrm{poly}((k \cdot nd/\delta)^{O(d^4)})$*-time and outputs, with probability at least $1-2^{-k}$, a list containing all degree $d$ polynomials that agree with $f$ on at least an $\delta$ fraction of the inputs. Furthermore, the list does not contain any polynomials that agree with $f$ on less than an $\frac{\delta}{2}$ fraction of the inputs.*

*Remarks.*

1. Thus, combining Theorems 2.1 and 3.12, we get reconstruction algorithms for all $d < q$, provided $\delta$ is large enough. Specifically, for the case $q = 2$ and $d = 1$, we invoke Theorem 2.1.
2. The constant $2e$ in the lower bound on $\delta$ can be replaced by $(1+\epsilon)e^{d/q}$, for any $\epsilon > 0$, by recalibrating the subroutine Test-valid and by setting $\alpha = 1-\frac{1}{q}$.

*Proof.* The main part of the correctness claim follows from Lemma 3.6, and the running-time bound follows from Lemmas 3.10 and 3.11. (In particular, note that the condition $\alpha^{d_0}\delta_0 \geq 2d/q$ from Lemma 3.10 is met, since $\alpha^{d_0} = \frac{1}{e}$ and $\delta_0 \geq 2\sqrt{d/q} \geq 2d/q$.) The furthermore part follows from the proof of Lemma 3.10. ☐

## 4. Counting: Worst case.

In this section we give a worst-case bound on the number of polynomials that agree with a given function $f$ on $\delta$ fraction of the points. In the case of linear polynomials our bound works for any $\delta > \frac{1}{q}$, while in the general case our bound works only for $\delta$ that is large enough. The bounds are derived using a very elementary property of polynomial functions, namely that two of them do not agree on too many points. In fact we first state and prove bounds for any generic "error-correcting code" and then specialize the bound to the case of polynomials.

### 4.1. General error-correcting bounds.

We first recall the standard definition of error-correcting codes. To do so we refer to strings over an alphabet $[q]$. For a string $R \in [q]^N$ ($R$ for received word) and $i \in [N]$, we let $R(i)$ denote the $i$th coordinate of $R$. The Hamming distance between strings $R_1$ and $R_2$, denoted $\Delta(R_1, R_2)$, is the number of coordinates $i$ where $R_1(i) \neq R_2(i)$.

DEFINITION 4.1 (error-correcting code). *For integers $N, K, D$, and $q$ an $[N, K, D]_q$ code is a family of $q^K$ strings from $[q]^N$ such that for any two distinct strings in the family, the Hamming distance between them is at least $D$. That is, if $\mathcal{C} \subseteq [q]^N$ is an $[N, K, D]_q$ code, then $|\mathcal{C}| = q^K$ and for every $C_1 \neq C_1 \in \mathcal{C}$ it holds that $\Delta(C_1, C_2) \geq D$.*

In the following theorem we take an arbitrary word $R \in [q]^N$ and consider the number of codewords that may have a Hamming distance of at most $(1-\delta)\cdot N$ from $R$ (i.e., codewords that agree with $R$ on at least $\delta \cdot N$ coordinates). We give an upper bound provided $\delta$ is sufficiently large (as a function of $D/N$).

THEOREM 4.2. *Let $N, D$, and $q$ satisfy $\frac{D}{N} < 1$ and define $\gamma \overset{\text{def}}{=} 1 - \frac{D}{N} > 0$. Let $\delta > 0$ and $R \in [q]^N$. Suppose that $C_1, \ldots, C_m \in [q]^N$ are distinct codewords from an $[N, K, D]_q$ code that satisfy $\Delta(R, C_j) \leq (1-\delta)\cdot N$ for all $j \in \{1, \ldots, m\}$. Then the following bounds hold:*

1. *If $\delta > \sqrt{2 + \frac{\gamma}{4}} \cdot \sqrt{\gamma} - \frac{\gamma}{2}$, then $m < \frac{2}{\delta + \frac{\gamma}{2}}$. It follows that if $\delta > \sqrt{2\gamma}$, then $m < 2/\delta$.*
2. *If $\gamma \geq \frac{1}{q}$ and $\delta > \frac{1}{q} + \sqrt{(\gamma - \frac{1}{q}) \cdot (1 - \frac{1}{q})}$, then $m \leq \frac{(1-\gamma)\cdot(1-\frac{1}{q})}{(\delta-(1/q))^2 - (1-\frac{1}{q})(\gamma - \frac{1}{q})}$.*

*It follows that if* ($\gamma \geq \frac{1}{q}$ *and*) $\delta > \min\{\sqrt{\gamma}, \frac{1}{q} + \sqrt{\gamma - \frac{1}{q}}\}$, *then* $m \leq \frac{1-\gamma}{\delta^2 - \gamma} < \frac{1}{\delta^2-\gamma}$. *In particular, for* $\gamma = \frac{1}{q}$, *the bounds hold for every* $\delta > \frac{1}{q}$.

For small $\gamma$, the latter (simpler) expressions given in each of the two parts of the theorem provide good approximations to the former (tighter) expressions. The fact that the former expressions imply the latter ones is obvious for part 1 and is proved below for part 2.

*Additional remarks.*

1. The bounds in the two parts of the theorem apply in different situations and yield different bounds on $m$. The first bound applies for somewhat larger values of $\delta$ and yields a stronger bound that is $O(\frac{1}{\delta})$. The second bound applies also for smaller values of $\delta$ and yields a bound that grows as $\Theta(\frac{1}{\delta^2})$.
2. Note that part 2 considers only codes with distance $D \leq (1 - 1/q) \cdot N$ (i.e., $\gamma \geq 1/q$). Still, the bound $m \leq \frac{(1-\gamma)\cdot(1-\frac{1}{q})}{(\delta-(1/q))^2-(1-\frac{1}{q})(\gamma-\frac{1}{q})}$ holds also in case $\gamma < 1/q$, provided $\delta \geq 1/q$. (See footnote 9 at the end of the proof of part 2.) We mention that it is well known that codes with distance $D \geq (1 - 1/q) \cdot N$ have at most $qN$ codewords, which immediately implies $m \leq qN \leq N/\gamma$ (for any $\gamma \geq 1/q$ regardless of $\delta$).

*Proof of part* 1. The bound in part 1 is proven by a simple inclusion-exclusion argument. For any $m' \leq m$, we count the number of coordinates $i \in [N]$ that satisfy the property that one of the first $m'$ codewords agrees with $R$ on coordinate $i$. Namely, let $\chi_j(i) = 1$ if $C_j(i) = R(i)$ and $\chi_j(i) = 0$ otherwise. Then, by inclusion-exclusion we get

$$\begin{aligned} N &\geq |\{i : \exists j\ \chi_j(i) = 1\}| \\ &\geq \sum_{j=1}^{m'} \sum_i \chi_j(i) - \sum_{1\leq j_1<j_2\leq m'} \sum_i \chi_{j_1}(i)\chi_{j_2}(i) \\ &\geq m' \cdot \delta N - \binom{m'}{2} \cdot \max_{1\leq j_1<j_2\leq m'} |\{i : C_{j_1}(i) = C_{j_2}(i)\}|, \end{aligned}$$

where the last inequality is due to the fact that $C_j$ agrees with $R$ on at least $\delta N$ coordinates. Since two codewords $R_1$ and $R_2$ can agree on at most $N - D$ coordinates, we get

$$\text{for all } m' \leq m, \quad m'\delta N - \frac{m'(m'-1)}{2} \cdot (N - D) \leq N. \tag{4.1}$$

Consider the function $g(y) \overset{\text{def}}{=} \frac{\gamma}{2} \cdot y^2 - (\delta + \frac{\gamma}{2}) \cdot y + 1$. Then (4.1) says that $g(m') \geq 0$ for every integer $m' \leq m$. Let $\alpha_1$ and $\alpha_2$ be the roots of $g$. To establish part 1 we show that

- the roots $\alpha_1$ and $\alpha_2$ are both real numbers,
- the roots are both nonnegative,
- $|\alpha_1 - \alpha_2| > 1$,
- $\min(\alpha_1, \alpha_2) < \frac{2}{\delta+\frac{\gamma}{2}}$.

Without loss of generality, suppose $\alpha_1 \leq \alpha_2$. It follows that $m \leq \alpha_1$, since otherwise $g(m') < 0$ for every $m' \in (\alpha_1, \alpha_2)$ and in particular for the integer $m' = \lfloor \alpha_1 \rfloor + 1$, in contradiction to the above (i.e., $g(m') \geq 0$ for every $m' \leq m$).

Let $\beta = \gamma/2$. Then $g(y) = \beta y^2 - (\beta + \delta) \cdot y + 1$. The roots $\alpha_1$ and $\alpha_2$ are real, provided that $\zeta \overset{\text{def}}{=} (\beta + \delta)^2 - 4\beta$ is positive which follows from a stronger requirement (see below). Without loss of generality, suppose $\alpha_1 \le \alpha_2$. To guarantee $\alpha_2 - \alpha_1 > 1$, we require $2 \cdot \frac{\sqrt{\zeta}}{2\beta} > 1$, which translates to $\zeta > \beta^2$ (and hence $\zeta > 0$ as required above). We need to show that

$$(\beta + \delta)^2 - 4\beta > \beta^2,$$

which occurs if $\delta > \sqrt{\beta^2 + 4\beta} - \beta$. Plugging in the value of $\beta$ we find that the last inequality is exactly what is guaranteed in the hypothesis of part 1 of the theorem statement. Thus $\alpha_1$ and $\alpha_2$ are real and $\alpha_2 - \alpha_1 > 1$. Last, we bound the smaller root $\alpha_1$. First we prove the upper bound:

$$\begin{aligned}
\alpha_1 &= \frac{\beta + \delta - \sqrt{(\beta+\delta)^2 - 4\beta}}{2\beta} \\
&= \frac{\beta+\delta}{2\beta} \cdot \left[1 - \left(1 - \frac{4\beta}{(\beta+\delta)^2}\right)^{1/2}\right] \\
&< \frac{\beta+\delta}{2\beta} \cdot \left[1 - \left(1 - \frac{4\beta}{(\beta+\delta)^2}\right)\right] \\
&= \frac{2}{\beta+\delta},
\end{aligned}$$

where the inequality follows by $\zeta > 0$. Again by plugging in the value of $\beta$ we get the desired bound. For the lower bound, consider the first equality in the above displayed set of inequalities and note that since $\beta > 0$, we have

$$\alpha_1 \; = \; \frac{\beta + \delta - \sqrt{(\beta+\delta)^2 - 4\beta}}{2\beta} \; > \; 0. \qquad \square$$

*Proof of part* 2. We first introduce some notation. In what follows we will use the arithmetic of integers modulo $q$ to simplify some of our notation. This arithmetic will be used on the letters of the alphabet, i.e., the set $[q]$. For $j \in \{1, \ldots, m\}$ and $i \in [N]$ let $\Gamma_j(i) = 1$ if $C_j(i) \neq R(i)$ and 0 otherwise. (Notice that $\Gamma_j(i) = 1 - \chi_j(i)$.) For $j \in \{1, \ldots, m\}$, $t \in \{0, \ldots, q-1\}$, and $i \in [N]$ let $\Gamma_j^{(t)}(i) = 1$ if $C_j(i) - R(i) \equiv t \pmod{q}$ and 0 otherwise. Thus $\Gamma_j(i) = 1$ if and only if there exists $t \neq 0$ such that $\Gamma_j^{(t)}(i) = 1$. Let $w_j \overset{\text{def}}{=} |\{i : C_j(i) \neq R(i)\}| = \sum_i \Gamma_j(i)$ and let $\overline{w} = \frac{\sum_{j=1}^m w_j}{m}$. The fact that the $C_j$'s are close to $R$ implies that $w_j \le (1-\delta) \cdot N$ for all $j$.

Our proof generalizes a proof due to S. Johnson (cf. MacWilliams and Sloane [31]) for the case $q = 2$. The central quantity used to bound $m$ in the binary case can be generalized in one of the two following ways:

$$\begin{aligned}
S &\equiv \sum_{j_1, j_2, i} \Gamma_{j_1}(i)\Gamma_{j_2}(i), \\
S' &\equiv \sum_{j_1, j_2, i} \sum_{t \neq 0} \Gamma_{j_1}^{(t)}(i)\Gamma_{j_2}^{(t)}(i).
\end{aligned}$$

The first quantity sums, over all $j_1, j_2$, the number of coordinates for which $C_{j_1}$ and $C_{j_2}$ both differ from $R$. The second quantity sums, over all $j_1, j_2$, the number of coordinates where $C_{j_1}$ and $C_{j_2}$ agree with each other but disagree from $R$ by $t$. (Notice that the two quantities are the same for the case $q = 2$.) While neither one of the two quantities are sufficient for our analysis, their sum provides good bounds.

*Lower bound on $S + S'$.* The following bound is shown using counting arguments which consider the worst way to place a given number of differences between the $C_j$'s and $R$. Let $N_i = |\{j|C_j(i) \neq R(i)\}| = \sum_j \Gamma_j(i)$ and let $N_i^{(t)} = |\{j|C_j(i) - R(i) \equiv t \pmod q\}| = \sum_j \Gamma_j^{(t)}(i)$. Note that $\sum_i N_i = \sum_i \sum_{t\neq 0} N_i^{(t)} = m\overline{w}$. We can lower bound $S$ as follows:

$$S = \sum_{j_1,j_2,i} \Gamma_{j_1}(i)\Gamma_{j_2}(i) = \sum_i N_i^2 \geq \frac{(m\overline{w})^2}{N},$$

where the last inequality above follows from the fact that subject to the condition $\sum_i N_i = m\overline{w}$, the sum of $N_i$'s squared is minimized when all the $N_i$'s are equal. Similarly, using $\sum_i \sum_{t\neq 0} N_i^{(t)} = m\overline{w}$, we lower bound $S'$ as follows:

$$S' = \sum_{j_1,j_2,i} \sum_{t\neq 0} \Gamma_{j_1}^{(t)}(i)\Gamma_{j_2}^{(t)}(i) = \sum_i \sum_{t\neq 0} (N_i^{(t)})^2 \geq \frac{(m\overline{w})^2}{(q-1)N}.$$

By adding the two lower bounds above we obtain

$$S + S' \geq \frac{(m\overline{w})^2}{N} + \frac{(m\overline{w})^2}{(q-1)N} = \frac{\frac{q}{q-1}m^2\overline{w}^2}{N}. \tag{4.2}$$

*Upper bound on $S + S'$.* For the upper bound we perform a careful counting argument using the fact that the $C_j$'s are codewords from an error-correcting code. For fixed $j_1, j_2 \in \{1, \dots, m\}$ and $t_1, t_2 \in [q]$, let

$$M_{t_1t_2}^{(j_1j_2)} \equiv |\{i|\Gamma_{j_1}^{(t_1)}(i) = \Gamma_{j_2}^{(t_2)}(i) = 1\}|.$$

For every $j_1, j_2$, we view the $M_{t_1t_2}^{(j_1j_2)}$'s as elements of a $q \times q$ matrix $M^{(j_1j_2)}$. Now, $S$ and $S'$ can be expressed as sums of some of the elements of the matrices $M^{(j_1j_2)}$. Summing over the $(q-1)\times(q-1)$ minors of all the matrices we get

$$S = \sum_{j_1,j_2} \sum_{t_1\neq 0} \sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)}$$

and summing the diagonal elements of $M^{(j_1j_2)}$ over all $j_1j_2$, we get

$$S' = \sum_{j_1j_2} \sum_{t\neq 0} M_{tt}^{(j_1j_2)}.$$

We start by upper bounding the internal sum above for fixed pair $(j_1, j_2)$, $j_1 \neq j_2$. Since the $C_j$'s are codewords from an $[N, K, D]_q$ code we have $R_{j_1}(i) = R_{j_2}(i)$ for at most $N - D$ values of $i$, so

$$\sum_{t\neq 0} M_{tt}^{(j_1j_2)} \leq N - D - M_{00}^{(j_1j_2)} = \gamma N - M_{00}^{(j_1j_2)}.$$

Note that the sum of the values of all elements of $M^{(j_1j_2)}$ equals $N$, and $N - w_{j_1}$ (resp., $N - w_{j_2}$) is equal to the sum of the values of the 0th column (resp., row) of $M^{(j_1j_2)}$.

To bound the remaining term in the summation above we use inclusion-exclusion as follows:

$$\begin{aligned}
&\sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)} \\
&= \sum_{t_1}\sum_{t_2} M_{t_1t_2}^{(j_1j_2)} - \sum_{t_1} M_{t_10}^{(j_1j_2)} - \sum_{t_2} M_{0t_2}^{(j_1j_2)} + M_{00}^{(j_1j_2)} \\
&= N - (N - w_{j_1}) - (N - w_{j_2}) + M_{00}^{(j_1j_2)} \\
&= w_{j_1} + w_{j_2} - N + M_{00}^{(j_1j_2)}.
\end{aligned}$$

Combining the bounds above we have (for $j_1 \neq j_2$)

$$\begin{aligned}
\sum_{t\neq 0} M_{tt}^{(j_1j_2)} + \sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)} &\le (\gamma N - M_{00}^{(j_1j_2)}) + (w_{j_1} + w_{j_2} - N + M_{00}^{(j_1j_2)}) \\
&= w_{j_1} + w_{j_2} - (1-\gamma)\cdot N.
\end{aligned}$$

(The key point above is the cancellation of $M_{00}^{(j_1j_2)}$.) Observe that if $j_1 = j_2 = j$, then the quantity $\sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(jj)} = \sum_{t\neq 0} M_{tt}^{(jj)} = w_j$.

We now combine the bounds above as follows:

$$\begin{aligned}
S + S' &= \sum_j \left( \sum_{t\neq 0} M_{tt}^{(jj)} + \sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(jj)} \right) + \sum_{j_1\neq j_2} \left( \sum_{t\neq 0} M_{tt}^{(j_1j_2)} + \sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)} \right) \\
&\le 2\sum_j w_j + \sum_{j_1\neq j_2} (w_{j_1} + w_{j_2} - (1-\gamma)N) \\
&= 2m^2\overline{w} - m(m-1)(1-\gamma)N.
\end{aligned}$$

Thus, we get

$$S + S' \;\le\; (2\overline{w} - (1-\gamma)\cdot N)\cdot m^2 + (1-\gamma)\cdot N\cdot m. \tag{4.3}$$

*Putting it together.* Combining (4.2) and (4.3) and letting $\overline{\delta} = 1 - \overline{w}/N$, we get

$$\begin{aligned}
m &\le (1-\gamma)\cdot \frac{1}{(\frac{\overline{w}}{N})^2\frac{q}{q-1} + 1 - \gamma - 2\cdot\frac{\overline{w}}{N}} \\
&= (1-\gamma)\cdot \frac{1}{(1-\overline{\delta})^2\frac{q}{q-1} + 1 - \gamma - 2(1-\overline{\delta})},
\end{aligned}$$

provided $(1-\overline{\delta})^2\frac{q}{q-1} + 1 - \gamma - 2(1-\overline{\delta}) \ge 0$. Let $g(x) \overset{\text{def}}{=} \frac{q}{q-1}x^2 - 2x + (1-\gamma)$. Note that $g(x)$ is monotone decreasing when $x \le \frac{q-1}{q}$. Note further that $\frac{1}{q} \le \delta \le \overline{\delta}$ and thus we get

$$m \le (1-\gamma)\cdot\frac{1}{g(1-\delta)},$$

provided $g(1-\delta) > 0$. We need to bound $\delta$ so that $g(1-\delta) > 0$. Observe first that $g(x) = \frac{q}{q-1}\cdot(\frac{q-1}{q} - x)^2 - (\gamma - \frac{1}{q})$. Thus $g(x) > 0$ if $\frac{q-1}{q} - x > \sqrt{\frac{q-1}{q}\cdot(\gamma - \frac{1}{q})}$. (Note that the expression in the square root is nonnegative, since $\gamma \geq \frac{1}{q}$.)[9] In other words, $g(1-\delta) > 0$, provided $\delta > \frac{1}{q} + \sqrt{(1-\frac{1}{q}) \cdot (\gamma - \frac{1}{q})}$. In this case the bound obtained on $m$ is $\frac{1-\gamma}{g(1-\delta)} = \frac{1-\gamma}{\frac{q}{q-1}\cdot(\delta-\frac{1}{q})^2 - (\gamma-\frac{1}{q})}$. This is exactly as claimed in the main part of part 2.

We now move on to prove secondary bounds claimed in part 2. First, we show that $g(1-\delta) > 0$ for $\delta > \frac{1}{q} + \sqrt{\gamma - \frac{1}{q}}$. This follows immediately from the above and the inequality

$$\frac{1}{q} + \sqrt{\gamma - \frac{1}{q}} \;\; > \;\; \frac{1}{q} + \sqrt{\left(1 - \frac{1}{q}\right) \cdot \left(\gamma - \frac{1}{q}\right)}.$$

Next, we verify that $g(1-\delta) > 0$ for every $\delta > \sqrt{\gamma}$. Let $x = 1-\delta$. Then $1-x = \delta > \sqrt{\gamma}$. In this case we have

$$\begin{aligned} g(x) &= \left(1 + \frac{1}{q-1}\right) x^2 - 2x + 1 - \gamma \\ &= (1-x)^2 + \frac{1}{q-1} x^2 - \gamma \\ &\geq (1-x)^2 - \gamma \\ &> 0. \end{aligned}$$

Thus $g(1-\delta) > 0$ provided $\delta > \min\{\sqrt{\gamma}, \frac{1}{q} + \sqrt{\gamma - \frac{1}{q}}\}$. We now derive the claimed upper bounds on $m$. Setting $x = 1-\delta$, and using $g(x) \geq (1-x)^2 - \gamma$, we get $g(1-\delta) \geq \delta^2 - \gamma$. Thus $m \leq \frac{1-\gamma}{g(1-\delta)} \leq \frac{1-\gamma}{\delta^2-\gamma} < \frac{1}{\delta^2-\gamma}$. □

**4.2. The special case of polynomials.** Recall that a function $f : \mathrm{GF(q)}^n \rightarrow \mathrm{GF(q)}$ may be viewed as a string of length $q^n$ with letters from the set $[q]$. Viewed in this way we get the following construction of a code using multivariate polynomials. These codes are known as Reed–Muller codes in the coding theory literature.

PROPOSITION 4.3. *The collection of degree $d$ polynomials in $n$ variables over* GF(q) *form an* $[N, K, D]_q$ *code for* $N = q^n$, $K = \binom{n+d}{d}$, *and* $D = (q-d) \cdot q^{n-1}$.

*Proof.* The parameters $N$ and $K$ follow by definition. The distance bound $D$ is equivalent to the well-known fact [10, 38, 46] that two degree $d$ (multivariate) polynomials over GF(q) may agree in at most $d/q$ fraction of the inputs. □

Combining Theorem 4.2 with Proposition 4.3 (and using $\gamma = \frac{d}{q}$ in the theorem), we get the following upper bound on the number of polynomials with $\delta$ agreement with an arbitrary function.

THEOREM 4.4. *Let $\delta > 0$ and $f : \mathrm{GF}(q)^n \rightarrow \mathrm{GF}(q)$. Suppose that $p_1, \ldots, p_m$ : $\mathrm{GF}(q)^n \rightarrow \mathrm{GF}(q)$ are distinct degree $d$ polynomials that satisfy*

$$\Pr_{x \in \mathrm{GF}(q)^n} [f(x) = p_i(x)] \geq \delta \quad \textit{for all } i \in \{1, \ldots, m\}.$$

[9]For $\gamma < \frac{1}{q}$, the function $g$ is positive everywhere. However, to use the inequality $g(1-\delta) \leq g(1-\overline{\delta})$, we need $\delta \geq \frac{1}{q}$. This gives the bound claimed in the additional remark 2 after Theorem 4.2.

*Then the following bounds hold:*

1. *If* $\delta > \sqrt{2+\frac{d}{4q}} \cdot \sqrt{\frac{d}{q}} - \frac{d}{2q}$, *then* $m < \frac{2}{\delta + \frac{d}{2q}}$. *In particular, if* $\delta > \sqrt{2d/q}$, *then* $m < 2/\delta$.
2. *If* $\delta > \frac{1+\sqrt{(d-1)(q-1)}}{q}$, *then* $m \le \frac{(q-d)(q-1)}{q^2} \cdot \frac{1}{\left(\delta - \frac{1}{q}\right)^2 - \frac{(q-1)(d-1)}{q^2}}$. *In particular, if* $\delta > \min\left\{\sqrt{\frac{d}{q}}, \frac{1}{q} + \sqrt{\frac{d-1}{q}}\right\}$, *then* $m < \frac{1}{\delta^2 - (d/q)}$.

We emphasize the special case of linear polynomials (i.e., $d = 1$).

THEOREM 4.5. *Let* $\epsilon > 0$ *and* $f : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$. *Suppose that* $p_1, \dots, p_m : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$ *are distinct linear functions that satisfy* $\Pr_{x\in \mathrm{GF}(q)^n}[f(x) = p_i(x)] \ge \frac{1}{q} + \epsilon$ *for all* $i \in \{1, \dots, m\}$. *Then* $m \le (1 - \frac{1}{q})^2 \cdot \frac{1}{\epsilon^2} \le \frac{4}{\epsilon^2}$.

*Proof.* Just substitute $d = 1$ and $\delta = \frac{1}{q} + \epsilon$ in the main part of part 2 of Theorem 4.4. □

**4.3. On the tightness of the upper bounds.** We show that several aspects of the bounds presented above are tight. We start with the observation that Theorem 4.2 cannot be extended to smaller $\delta$ without (possibly) relying on some special properties of the code.

PROPOSITION 4.6. *Let* $\delta_0, \gamma_0$ *satisfy the identity*

$$\delta_0 = \frac{1}{q} + \sqrt{\left(\gamma_0 - \frac{1}{q}\right) \cdot \left(1 - \frac{1}{q}\right)}. \tag{4.4}$$

*Then for any* $\epsilon > 0$, *and for sufficiently large* $N$, *there exists an* $[N, K, D]_q$ *code* $\mathcal{C}$, *with* $\frac{N-D}{N} \le \gamma_0 + \epsilon$, *a word* $R \in [q]^N$ *and* $M \ge 2^{\Omega(\epsilon^2 N)}$ *codewords* $C_1, \dots, C_M \in \mathcal{C}$ *such that* $\Delta(R, C_j) \le (1 - (\delta_0 - \epsilon)) \cdot N$, *for every* $j \in [M]$.

*Remark.* The proposition above should be compared against part 2 of Theorem 4.2. That part says that for $\delta_0$ and $\gamma_0$ satisfying (4.4) and *any* $[N, K, D]_q$ code with $\frac{N-D}{N} = \gamma_0$, there exist at most $O(\frac{1}{\delta_0^2})$ codewords at distance at most $(1 - \delta_0) \cdot N$ from any string of length $N$. In contrast, the proposition says that if $\delta_0$ is reduced slightly (to $\delta_0 - \epsilon$) and $\gamma_0$ increased slightly (to $\gamma_0 + \epsilon$), then there could be exponentially many codewords at this distance.

*Proof.* The bound is proven by a standard probabilistic argument. The code $\mathcal{C}$ will consist only of the codewords $C_1, \dots, C_M$ that will be close to the string $R$. The codewords $C_j$'s are chosen randomly and independently by the following process. Let $p \in [0, 1]$, to be determined shortly.

For every codeword $C_j$, each coordinate is chosen independently as follows: With probability $p$ it is set to be 1, and with probability $1 - p$ it is chosen uniformly from $\{2, \dots, q\}$. The string $R$ is simply $1^N$.

Observe that for any fixed $j$, the expected number of coordinates where $R$ and $C_j$ agree is $pN$. Thus with probability at most $2^{-\Omega(\epsilon^2 N)}$, the agreement between $R$ and $C_j$ is less than $(p - \epsilon)N$. It is possible to set $M = 2^{\Omega(\epsilon^2 N)}$ so that the probability that there exists such a word $C_j$ is less than $\frac{1}{2}$.

Similarly, the expected agreement between $C_i$ and $C_j$ is $(p^2 + \frac{(1-p)^2}{q-1}) \cdot N$. Thus the probability that the agreement between a fixed pair is $\epsilon N$ larger than this number is at most $2^{-\Omega(\epsilon^2 N)}$. Again it is possible to set $M = 2^{\Omega(\epsilon^2 N)}$ such that the probability that such a pair $C_i$ and $C_j$ exists is less than $\frac{1}{2}$.

Thus there is a positive probability that the construction yields an $[N, \Omega(\frac{\epsilon^2 N}{\log q}), D]_q$ code with $\frac{N-D}{N} = p^2 + \frac{(1-p)^2}{q-1} + \epsilon$ so that all codewords are within a distance of $(1-(p-\epsilon))N$ of the word $R$. Thus, the setting $\delta_0 = p$ and $\gamma_0 = p^2 + \frac{(1-p)^2}{q-1}$ would yield the proposition, once it is verified that this setting satisfies (4.4). The latter fact is easily verified by the following algebraic manipulations, starting with our setting of $\delta_0$ and $\gamma_0$:

$$\begin{aligned}
\gamma_0 = \delta_0^2 + \frac{(1-\delta_0)^2}{q-1}& \\
&\Leftrightarrow \frac{q}{q-1}\cdot\delta_0^2 - \frac{2}{q-1}\cdot\delta_0 + \frac{1}{q-1} - \gamma_0 = 0\\
&\Leftrightarrow \delta_0^2 - \frac{2}{q}\cdot\delta_0 + \frac{1}{q} - \frac{q-1}{q}\cdot\gamma_0 = 0\\
&\Leftrightarrow \left(\delta_0 - \frac{1}{q}\right)^2 = \left(\gamma_0 - \frac{1}{q}\right)\cdot\left(1-\frac{1}{q}\right)\\
&\Leftrightarrow \delta_0 = \frac{1}{q} + \sqrt{\left(\gamma_0 - \frac{1}{q}\right)\cdot\left(1-\frac{1}{q}\right)}.
\end{aligned}$$

This concludes the proof. □

Next we move on to the tightness of the bounds regarding polynomials. We show that Theorem 4.5 is tight for $\delta = O(1/q)$, whereas part 1 of Theorem 4.4 is tight for $\delta = \Theta(1/\sqrt{q})$ and $d = 1$. The results below show that for a given value of $\delta$ that meets the conditions of the appropriate theorem, the value of $m$ cannot be made much smaller.

PROPOSITION 4.7. *Given a prime $p$, and an integer $k$ satisfying $1 < k \le p/3$, let $\delta = k/p$. Then there exists a function $f : \mathrm{GF}(p) \to \mathrm{GF}(p)$ and at least $m \stackrel{\text{def}}{=} \frac{1}{18(k-1)\delta^2}$ linear functions $f_1, \ldots, f_m : \mathrm{GF}(p) \to \mathrm{GF}(p)$ such that $|\{x | f_i(x) = f(x)\}| \ge \delta p = k$ for all $i \in \{1, \ldots, m\}$. Furthermore, if $\delta > \sqrt{1/p}$, then $m > \frac{1}{\delta} - 1$.* For $\delta = \frac{2}{p} = \frac{1}{p} + \epsilon$, we get $m = \frac{1}{18\delta^2}$ (which establishes tightness of the bound $m \le \frac{4}{\epsilon^2} = \frac{16}{\delta^2}$ given in Theorem 4.5). For $\delta = \sqrt{\frac{2}{p}} + \frac{1}{p} > \sqrt{\frac{2}{p}}$, we get $m > \frac{1}{\delta} - 1$ (which establishes tightness of the bound $m \le \frac{2}{\delta}$ given for $d = 1$ in part 1 of Theorem 4.4).

*Proof.* We start by constructing a relation $R \subset \mathrm{GF}(p) \times \mathrm{GF}(p)$ such that $|R| \le p$ and there exist many linear functions $g_1, \ldots, g_m$ such that $|R \cap \{(x, g_i(x)) | x \in \mathrm{GF}(p)\}| \ge k$ for all $i$. Later we show how to transform $R$ and the $g_i$'s so that $R$ becomes a function that still agrees with each transformed $g_i$ on $k$ inputs.

Let $l = \lfloor p/k \rfloor$ and recall that $\delta = k/p$. Notice $l \approx \frac{1}{\delta}$ and $l \ge \frac{1}{\delta} - 1$. The relation $R$ consists of the $k \cdot l \le p$ pairs in the square $\{(i,j) | 0 \le i < k, 0 \le j < l\}$. Let $\mathcal{G}$ be the set of all linear functions that agree with $R$ in at least $k$ places. We shall show that $\mathcal{G}$ has size at least $1/(18\delta^2(k-1))$. Given nonnegative integers $a, b$ subject to $a \cdot (k-1) + b < l$, consider the linear function $g_{a,b}(x) = ax + b \bmod p$. Then $g_{a,b}(i) \in \{0, \ldots, l-1\}$ for every such $(a,b)$ and $i \in \{0, \ldots, k-1\}$. Thus, $g_{a,b}(i)$ intersects $R$ in $k$ places. Last, we observe that there are at least $1/(18\delta^2(k-1))$ distinct pairs $(a,b)$ such that $a \cdot (k-1) + b < l$. Fixing any $a < l$, there are at least

$l-(k-1)a-1$ possible values for $b$, and so that the total number of pairs is at least

$$\begin{aligned}\sum_{a=0}^{\frac{l-1}{k-1}} l-(k-1)a-1 &= \left(\frac{l-1}{k-1}+1\right)\cdot(l-1)-(k-1)\cdot\frac{\frac{l-1}{k-1}\cdot\left(\frac{l-1}{k-1}+1\right)}{2}\\ &> \frac{(l-1)^2}{2(k-1)}\\ &\geq \frac{(1-2\delta)^2}{2\delta^2(k-1)} \qquad (\text{using } l\geq \tfrac{1-\delta}{\delta})\\ &\geq \frac{1}{18\delta^2(k-1)} \qquad (\text{using } \delta\leq\tfrac{1}{3}).\end{aligned}$$

Next, we convert the relation $R$ into a function in two stages. First we *stretch* the relation by a factor of $l$ to get a new relation $R'$. That is, $R' \stackrel{\text{def}}{=} \{(l\cdot i,j)|(i,j)\in R\}$. We modify the functions $g_{a,b}\in\mathcal{G}$ accordingly. That is, $g'_{a,b}(x)\stackrel{\text{def}}{=} g_{a,b}(l^{-1}\cdot x)=(a\cdot l^{-1})x+b$, where $l^{-1}$ is the multiplicative inverse of $l$ (mod $p$) and $g_{a,b}(x)=ax+b$. Thus, if $g_{a,b}(i)=j$, then $g'_{a,b}(l\cdot i)=j$, and so if $(i,g_{a,b}(i))\in R$, then $(l\cdot i, g'_{a,b}(l\cdot i))\in R'$. It follows that if $g_{a,b}$ agrees with $R$ on at least $k$ places, then $g'_{a,b}$ agrees with $R'$ on at least $k$ places. Thus, letting $\mathcal{G}'$ denote the set of linear functions that agree with $R'$ in $k$ places, we have $g'_{a,b}\in\mathcal{G}'$ if $g_{a,b}\in\mathcal{G}$. Moreover, the map from $\mathcal{G}$ to $\mathcal{G}'$ is one-to-one (i.e., $g_{a,b}$ is mapped to $g'_{a,b}\equiv g_{l^{-1}\cdot a,b}$), implying $|\mathcal{G}'|\geq|\mathcal{G}|$. (Actually, the argument above extends to show that $|\mathcal{G}'|=|\mathcal{G}|$.)

We note that for all $a<l$ (which in turn is smaller than $p/2$), it holds that $l^{-1}\cdot a\not\equiv -1(\text{mod } p)$. (This is the case since otherwise $a\equiv -l\equiv p-l(\text{mod } p)$, in contradiction to $a<p/2$.)

Last we introduce a slope to $R'$, so that it becomes a function. Specifically, $R''\stackrel{\text{def}}{=}\{(i+j,j)|(i,j)\in R'\}=\{(l\cdot i+j,j)|(i,j)\in R\}$. Notice that for any two distinct $(i_1,j_1),(i_1,j_2)\in R''$, we have $i_1\neq i_2$ (since $i_1=l\cdot i'_1+j_1$, $i_2=l\cdot i'_2+j_2$, and $j_1,j_2\in\{0,\ldots,l-1\}$), and so $R''$ can be extended to a function $f:\text{GF}(p)\to\text{GF}(p)$ (i.e., if $(i,j)\in R''$, then $j=f(i)$). Now for every function $g'(x)=a'x+b'\in\mathcal{G}'$, consider the function $g''(x)=a''x+b''$, where $a''=a'/(1+a')$ and $b''=b'/(1+a')$ (and recalling that $a'\not\equiv -1(\text{mod } p)$). Observe that if $g'(x)=y$, then

$$\begin{aligned}g''(x+y) &= \frac{a'}{1+a'}\cdot(x+g'(x))+\frac{b'}{1+a'}\\ &= \frac{a'}{1+a'}\cdot(x+a'x+b')+\frac{1}{1+a'}\cdot b'\\ &= a'x+b' \;=\; y.\end{aligned}$$

Thus, if $g'$ agrees with $R'$ in at least $k$ places, then $g''$ agrees with $R''$ in at least $k$ places (since $(x,g'(x))\in R'$ implies $(x+g'(x),g''(x+g'(x)))\in R''$ and $x_1+g'(x_1)=(a'+1)\cdot x_1+b'_1\neq(a'+1)\cdot x_2+b'_1=x_2+g'(x_2)$ for all $x_1\neq x_2$), and hence $g''$ agrees with $f$ in at least $k$ places. Again, the mapping of $g'$ to $g''$ is one-to-one (since the system $a''=a'/(1+a')$ and $b''=b'/(1+a')$ has at most one solution in $(a',b')$). Thus, if we use $\mathcal{G}''$ to denote the set of linear functions that agree with $f$ in $k$ places, then we have $|\mathcal{G}''|\geq|\mathcal{G}'|\geq|\mathcal{G}|\geq\frac{1}{18\delta^2(k-1)}$, as desired.

For the furthermore clause, observe that if $\delta>\sqrt{1/p}$, then our setting dictates $l-1<\sqrt{p}<k$ and so $\frac{l-1}{k-1}<1$. Actually, in this case we may use $\{g_{0,b}:b=0,\ldots,l-1\}$ in place of $\mathcal{G}$, $\mathcal{G}'$, and $\mathcal{G}''$, and derive $|\mathcal{G}|\geq l\geq\frac{1}{\delta}-1$. $\square$

Finally we note that the bounds in Theorem 4.4 always require $\delta$ to be larger than $d/q$. Such a threshold is also necessary, or else there can be exponentially many degree $d$ polynomials close to the given function. This is shown in the following proposition.

PROPOSITION 4.8. *Let $q$ be a prime-power, $d < q$, and $\delta = \frac{d}{q} - \frac{d-1}{q^2}$. Then, there exists an $n$-variate function $f$ over $\mathrm{GF}(q)$, and at least $q^{n-1}$ degree $d$ polynomials that agree with $f$ on at least a $\delta$ fraction of the inputs.*

Note that for $d = 1$ we have $\delta = \frac{1}{q}$. Also, by a minor extension of the following proof, we may use in role of $f$ any $n$-variate degree $d$ polynomial over $\mathrm{GF}(q)$.

*Proof.* We use the all-zero function in role of $f$. Consider the family of polynomials having the form $\prod_{i=1}^{d-1}(x_1 - i) \cdot \sum_{i=2}^{n} c_i x_i$, where $c_2, \ldots, c_n \in \mathrm{GF}(q)$. Clearly, each member of this family is a degree $d$ polynomial and the family contains $q^{n-1}$ different polynomials. Now, each polynomial in the family is zero on inputs $(a_1, \ldots, a_n)$ satisfying either $a_1 \in \{1, \ldots, (d-1)\}$ or $\sum_{i=2}^{n} c_i a_i = 0$, where the $c_i$'s are these specifying the polynomial in the collection. Since at least a $\frac{d-1}{q} + (1 - \frac{d-1}{q}) \cdot \frac{1}{q}$ fraction of the inputs satisfy this condition, the proposition follows. $\square$

**5. Counting: A random case.** In this section we present a bound on the number of polynomials that can agree with a function $f$ if $f$ is chosen to look like a polynomial $p$ on some domain $D$ and random on other points. Specifically, for $|D| \geq 2(d+1) \cdot q^{n-1}$, we show that with high probability $p$ itself is the only polynomial that agrees with $f$ on at least $|D|$ (and even $|D|/2$) points.

THEOREM 5.1. *Let $\delta \geq \frac{2(d+1)}{q}$. Suppose that $D$ is an arbitrary subset of density $\delta$ in $\mathrm{GF}(q)^n$ and $p(x_1, \ldots, x_n)$ is a degree $d$ polynomial. Consider a function $f$ selected as follows:*

1. *$f$ agrees with $p$ on $D$;*
2. *the value of $f$ on each of the remaining points is uniformly and independently chosen. That is, for every $x \in \overline{D} \stackrel{\text{def}}{=} \mathrm{GF}(q)^n \setminus D$, the value of $f(x)$ is selected at random in* GF(q).

*Then, with probability at least $1 - \exp\{(n^d \log_2 q) - \delta^2 q^{n-2}\}$, the polynomial $p$ is the only degree $d$ polynomial that agrees with $f$ on at least a $\delta/2$ fraction of the inputs.*

Thus, for functions constructed in this manner, the output of our reconstruction algorithm will be a single polynomial, namely, $p$ itself.

*Proof.* We use the fact that for two polynomials $p_1 \neq p_2$ in $\mathrm{GF}(q)^n$, $p_1(x) = p_2(x)$ on at most $d/q$ fraction of the points in $\mathrm{GF}(q)^n$ [10, 38, 46]. Thus, except for $p$, no other degree $d$ polynomial can agree with $f$ on more than $\frac{d}{q} \cdot q^n$ points in $D$. The probability that any polynomial $p'$ agrees with $f$ on more than a $\frac{1}{q} + \epsilon$ fraction of the points in $\overline{D}$ is at most $\exp\{-\epsilon^2 q^n\}$. Furthermore, in order to agree with $f$ on more than an $\frac{\delta}{2}$ fraction of all points, $p'$ must agree with $f$ on at least $(\frac{\delta}{2} - \frac{d}{q}) \cdot q^n$ of the points in $\overline{D}$, and so we can use $\epsilon \geq \frac{(\delta/2)-(d/q)}{1-\delta} - \frac{1}{q} > \frac{\delta}{2} - \frac{d+1}{q} + \frac{\delta \cdot ((\delta/2)-(d/q))}{q} \geq \frac{\delta}{q}$. Thus, the probability that there exists a degree $d$ $n$-variate polynomial, other than $p$, that agrees with $f$ on at least an $\delta/2$ fraction of all points is at most $q^{n^d} \cdot \exp\{-(\frac{\delta}{q})^2 q^n\}$, and the theorem follows. $\square$

**6. Hardness results.** In this section we give evidence that the (explicit or implicit) reconstruction problem may be hard for some choices of $d$ and the agreement parameter $\delta$, even in the case when $n = 1$. We warn the reader that the problems shown to be hard do differ in some very significant ways from the reconstruction problems considered in previous sections. In particular, the problems will consider functions and relations defined on some finite subset of a large field, either the field of rational numbers or a sufficiently large field of prime order, where the prime is specified in binary. The hardness results use the "large" field size crucially.

Furthermore, the agreement threshold for which the problem is shown hard is very small. For example, the hardness results of section 6.2, defines a function $f : H_1 \times H_2 \to F$, where $F$ is a large field and $H_1, H_2$ are small subsets of $F$. In such a hardness result, one should compare the threshold $\delta$ of agreement that is required, against $\frac{d}{\max\{|H_1|,|H_2|\}}$, since the latter ratio that determines the "distance" between two polynomials on this subset of the inputs. Our hardness results typically hold for $\delta \approx \frac{d+2}{\max\{|H_1|,|H_2|\}}$. We stress that the agreement is measured as a fraction of the subset mentioned above, rather than as a fraction of the $n$-tuples over the field (in case it is finite), which is much smaller.

## 6.1. NP-hardness for a variant of the univariate reconstruction problem.

We define the following (*variant* of the) interpolation problem PolyAgree:

**Input:** Integers $d, k, m$, and a set of pairs $P = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ such that for all $i \in [m]$, $x_i \in F$, $y_i \in F$, where $F$ is either the field of rationals or a prime field given by its size in binary.[10]

**Question:** Does there exist a degree $d$ polynomial $p : F^n \to F$ for which $p(x_i) = y_i$ for at least $k$ different $i$'s?

We stress that the pairs in $P$ are *not* required to have distinct $x$-components (i.e., $x_i = x_j$ may hold for some $i \neq j$). Our result takes advantage of this fact.

THEOREM 6.1. PolyAgree *is NP-hard.*

*Remark.* This result should be contrasted with the results of [40, 19]. They show that PolyAgree is easy provided $k \geq \sqrt{dm}$, while our result shows it is hard without this condition. In particular, the proof uses $m = 2d + 3$ and $k = d + 2$ (and so $k < \sqrt{dm}$). Furthermore, our result is established using a set of pairs in which $x_i = x_j$ holds for some $i \neq j$, whereas this never happens when given oracle access to a function (as in previous sections and in [40, 19]). In the next subsection, we show that it is hard even in the oracle setting when the number of variables is at least two.

*Proof.* We present the proof for the case of the field of rational numbers only. It is easy to verify that the proof also holds if the field $F$ has prime order that is sufficiently large (see parenthetical comments at the end of the proof for further details.)

We reduce from subset sum: Given integers $B, a_1, \ldots, a_\ell$, does there exist a subset of the $a_i$'s that sum to $B$ (without loss of generality, $a_i \neq 0$ for all $i$).

In our reduction we use the fact that degree $d$ polynomials satisfy certain interpolation identities. In particular, let $\alpha_i = (-1)^{i+1}\binom{d+1}{i}$ for $1 \leq i \leq d+1$ and $\alpha_0 = -1$. Then $\sum_{i=0}^{d+1} \alpha_i f(i) = 0$ if and only if $(0, f(0)), (1, f(1)), \ldots, (d+1, f(d+1))$ lies on a degree $d$ univariate polynomial.

We construct the following instance of PolyAgree. Set $d = l - 1$, $m = 2d + 3$, and $k = d+2$. Next, set $x_i \leftarrow i$, $x_{d+1+i} \leftarrow i$, $y_i \leftarrow a_i/\alpha_i$, and $y_{d+1+i} \leftarrow 0$ for $1 \leq i \leq d+1$. Finally, set $x_{2d+3} \leftarrow 0$ and $y_{2d+3} \leftarrow B$.

No polynomial can pass through both $(x_i, y_i) = (i, a_i/\alpha_i)$ and $(x_{d+1+i}, y_{d+1+i}) = (i, 0)$ for any $i$, since $a_i \neq 0$. We show that there is a polynomial of degree $d$ that passes through $(0, B)$ and one of either $(i, 0)$ or $(i, a_i/\alpha_i)$ for each $1 \leq i \leq d+1$ if and only if there is a subset of $a_1, \ldots, a_{d+1}$ whose sum is $B$.

[10]When $F$ is the field of rational numbers, the input elements are assumed to be given as a ratio of two $N$-bit integers. In such a case the input size is measured in terms of the total bit length of all inputs.

Assume that there is a polynomial $p$ of degree $d$ that passes through $(0, B)$ and one of $(i, 0)$ and $(i, a_i/\alpha_i)$ for each $1 \le i \le d+1$. Let $S$ denote the set of indices for which $p(i) = a_i/\alpha_i$ (and $p(i) = 0$ for $i \in [d+1]\backslash S$). Then

$$0 = \sum_{i=0}^{d+1} \alpha_i p(i) = \alpha_0 \cdot B + \sum_{i \in S} \alpha_i \cdot \frac{a_i}{\alpha_i} = -B + \sum_{i \in S} a_i. \tag{6.1}$$

Similarly, if there is set of indices $S$ such that $\sum_{i \in S} a_i = B$, then we define $f$ so that $f(0) = B$, $f(i) = a_i/\alpha_i$ for $i \in S$, and $f(i) = 0$ for $i \in [d+1]\backslash S$. Observing that $\sum_{i=0}^{d+1} \alpha_i f(i) = 0$ it follows that there is a degree $d$ polynomial that agrees with $f$ on $i = 0, \ldots, d+1$.

For the case where $F$ is a finite field of order $q$, we assume that the integers $B$ and $a_1, \ldots, a_{d+1}$ are all multiples of $\alpha_i$ for every $i$. (This assumption can be realized easily by multiplying all integers in the input by $\text{lcm}(|\alpha_0|, \ldots, |\alpha_{d+1}|)$.) Further we pick $q > |B| + \sum_{i=1}^{d+1} |a_i|$. The only change to the proof is that the equalities in (6.1) directly hold only modulo $q$. At this stage, we use the condition $q > |B| + \sum_{i=1}^{d+1} |a_i|$ to conclude that $B = \sum_{i \in S} a_i$. ◻

**6.2. NP-hardness of the reconstruction problem for $n \ge 2$.** In the above problem, we did not require that the $x_i$'s be distinct. Thus this result does not directly relate to the black box model used in this paper. The following result applies to our black box model for $n$-variate functions, for any $n \ge 2$.

We define a multivariate version of PolyAgree that requires that the $x_i$'s be distinct. We actually define a parameterized family $\mathsf{FunctionalPolyAgree}_n$, for any $n \ge 1$.
**Input:** Integer $d$, a field $F$, a finite subset $H \subseteq F^n$, a rational number $\delta$, and a function $f : H \to F$, given as a table of values.
**Question:** Does there exist a degree $d$ polynomial $p : F^n \to F$ for which $p(x) = f(x)$ for at least $\delta$ fraction of the $x$'s from $H$?

THEOREM 6.2. *For every $n \ge 2$, $\mathsf{FunctionalPolyAgree}_n$ is NP-hard.*

*Proof.* We prove the theorem for $n = 2$. The other cases follow by simply making an instance where only the values of first two variables vary in the set $H$ and the remaining variables are assigned some fixed value (say 0).

The proof of this theorem builds on the previous proof. As above we reduce from subset sum. Given an instance $B, a_1, \ldots, a_l$ of the subset sum problem, we set $d = l - 1$ and $k = 2(d+1)$ and $F$ to be the field of rationals. (We could also work over any prime field GF$(q)$, provided $q \ge |B| + \sum_{i=1}^{n} |a_i|$.) Let $\delta = \frac{d+3}{2(d+2)}$. We set $H_1 = \{0, \ldots, d+1\}$, $H_2 = [2k]$, and let $H = H_1 \times H_2$. For $i \in H_1$ we let $\alpha_i = (-1)^{i+1}\binom{d+1}{i}$ as before. For $i \in H_1 - \{0\}$, let $y_i = a_i/\alpha_i$ as before. The function $f$ is defined as follows:

$$f(i,j) = \begin{cases} B & \text{if } i = 0, \\ y_i & \text{if } i \in H_1 - \{0\} \text{ and } j \in [k], \\ 0 & \text{otherwise (i.e., if } i \in H_1 - \{0\} \text{ and } j \in \{k+1, \ldots, 2k\}. \end{cases}$$

This completes the specification of the instance of the $\mathsf{FunctionalPolyAgree}_2$ problem. We now argue that if the subset sum instance is satisfiable, then there exists a polynomial $p$ with agreement $\delta$ (on inputs from $H$) with $f$. Let $S \in [l]$ be a subset such that $\sum_{i \in S} a_i = B$. Then the function

$$p(i,j) \stackrel{\text{def}}{=} p'(i) \stackrel{\text{def}}{=} \begin{cases} B & \text{if } i = 0, \\ y_i & \text{if } i \in S, \\ 0 & \text{if } i \in H_1 \setminus S \end{cases}$$

is a polynomial in $i$ of degree $d$ (since $\sum_{i=0}^{d+1} \alpha_i p'(i) = -B + \sum_{i \in S} a_i = 0$). Furthermore, $p$ and $f$ agree in $2k+k(d+1)$ inputs from $H$. In particular $p(0,j) = f(0,j) = B$ for every $j \in [2k]$, $p(i,j) = f(i,j) = y_i$ if $i \in S$ and $j \in [k]$ and $p(i,j) = f(i,j) = 0$ if $i \notin S$ and $j \in \{k+1, \ldots, 2k\}$. Thus $p$ and $f$ agree on a fraction $\frac{2k+k(d+1)}{2(d+2)k} = \frac{d+3}{2(d+2)} = \delta$ of the inputs from $H$, as required.

We now argue that if instance of the $\mathsf{FunctionalPolyAgree}_2$ problem produced by the reduction is satisfiable, then the subset sum instance is satisfiable. Fix a polynomial $p$ that has agreement $\delta$ with $f$; i.e., $p(i,j) = f(i,j)$ for at least $2k+k(d+1)$ inputs from $H$. We argue first that in such a case $p(i,j) = p'(i)$ for some polynomial $p'(i)$ and then the proof will be similar to that of Theorem 6.1. The following claim is crucial in this proof.

Claim 6.3. *For any $i \in [d+1]$, if $|\{j|p(i,j) = f(i,j)\}| \geq k$, then there exists $c_i \in \{0, y_i\}$ such that $p(i,j) = c_i$ for every $j \in [2k]$.*

*Proof.* Consider the function $p^{(i)}(j) \stackrel{\text{def}}{=} p(i,j)$. $p^{(i)}$ is a degree $d$ polynomial in $j$. By hypothesis (and the definition of $f(i,j)$) we have $p^{(i)}(j) \in \{0, y_i\}$ for $k$ values of $j \in [2k]$. Hence $p^{(i)}(j) = 0$ for $k/2$ values of $j$ or $p^{(i)}(j) = y_i$ for $k/2$ values of $j$. In either case we have that $p^{(i)}$, a degree $d$ polynomial, equals a constant polynomial for $k/2 = d+1$ points implying that $p^{(i)}$ is a constant. That $p^{(i)}(j) = c_i \in \{0, y_i\}$ follows from the hypothesis and definition of $f$. $\square$

From the claim above it follows immediately that for any $i \in [d+1]$, $|\{j|f(i,j) = p(i,j)\}| \leq k$. Now using the fact that $f$ and $p$ agree on $2k + k(d+1)$ inputs it follows that for every $i \in [d+1]$, $f(i,j) = p(i,j)$ for exactly $k$ values of $j$, and $f(0,j) = p(0,j) = B$ for all values of $j$. Using the above claim again we conclude that we can define a function $p'(i) \stackrel{\text{def}}{=} c_i \in \{0, y_i\}$ if $i \in [d+1]$ and $p'(0) = B$ such that $p(i,j) = p'(i)$ for every $(i,j) \in H$. Furthermore $p'(i)$ is a degree $d$ polynomial, since $p$ is a degree $d$ polynomial; and hence $\sum_{i=0}^{d+1} \alpha_i p'(i) = 0$. Letting $S = \{i \in [d+1] | y_i \neq 0\}$, we get $-B + \sum_{i \in S} \alpha_i y_i = 0$, which in turns implies $B = \sum_{i \in S} a_i$. Thus the instance of the subset sum problem is satisfiable. This concludes the proof. $\square$

## 7. An application to complexity theory.

In this section we use the reconstruction algorithm for linear-polynomials to prove the security of new, generic, hard-core *functions*. This generalizes the result of Goldreich and Levin [17] that provided hard-core *predicates*. We comment that an alternative construction of generic hard-core *functions* was also presented in [17], where its security was reduced to the security of a specific hard-core predicate via a "computational XOR lemma" due to [43]. For further details, see [16].

Loosely speaking, a function $h : \{0,1\}^* \to \{0,1\}^*$ is called a *hard-core* of a function $f : \{0,1\}^* \to \{0,1\}^*$ if $h$ is polynomial-time, but given $f(x)$ it is infeasible to distinguish $h(x)$ from a random $|h(x)|$-bit long string. Thus, not only is $h(x)$ hard to find given $f(x)$, it is even hard to recognize pairs of the form $(h(x), f(x))$. Intuitively, if $h$ is a hard-core of $f$, then it must be hard to invert $f$ (i.e., given $f(x)$ find $x$). We formulate all these notions below, assuming for simplicity that $f$ is length-preserving, i.e., $|f(x)| = |x|$ for all $x \in \{0,1\}^*$.

Definition 7.1 (one-way function). *A function $f : \{0,1\}^* \to \{0,1\}^*$ is called* one-way *if the following two conditions hold:*

1. *The function $f$ is computable in polynomial-time.*
2. *For every probabilistic polynomial-time algorithm $A$, every polynomial $p$, and*

*all sufficiently large $n$*

$$\mathbf{P}_{x\in\{0,1\}^n}[A(f(x)) \in f^{-1}(f(x))] < \frac{1}{p(n)}.$$

DEFINITION 7.2 (hard-core function). *A function $h : \{0,1\}^* \to \{0,1\}^*$ is called a* hard-core *of a function $f : \{0,1\}^* \to \{0,1\}^*$ if the following two conditions hold:*

1. *The function $h$ is computable in polynomial-time.*
2. *For every probabilistic polynomial-time algorithm $D$, every polynomial $p$, and all sufficiently large $n$*

$$\left|\mathbf{P}_{x\in\{0,1\}^n}[D(f(x),h(x))=1] - \mathbf{P}_{x\in\{0,1\}^n\,,\,r\in\{0,1\}^{|h(x)|}}[D(f(x),r)=1]\right| < \frac{1}{p(n)}.$$

THEOREM 7.3. *Let $f'$ be a one-way function and $\ell$ be a polynomial-time computable integer function satisfying $\ell(m) = O(\log m)$. For $x = (x_1,\ldots,x_m) \in \mathrm{GF}(2^{\ell(m)})^m$ and $y = (y_1,\ldots,y_m) \in \mathrm{GF}(2^{\ell(m)})^m$, let $f(x,y) \overset{\text{def}}{=} (f'(x), y)$ and $h(x,y) \overset{\text{def}}{=} \sum_{i=1}^m x_i y_i$. Then $h$ is a hard-core of $f$.*

This theorem generalizes the result of Goldreich and Levin [17] from $\ell \equiv 1$ to any logarithmically bounded $\ell$.

*Proof.* Assume for contradiction that there exists a probabilistic polynomial-time algorithm $D$ and a polynomial $p$ so that for infinitely many $m$'s

$$\left|\begin{array}{l}\mathbf{P}_{x,y\in\mathrm{GF}(2^{\ell(m)})^m}[D(f(x,y),h(x,y))=1] \\ \quad - \ \mathbf{P}_{x,y\in\mathrm{GF}(2^{\ell(m)})^m\,,\,r\in\mathrm{GF}(2^{\ell(m)})}[D(f(x,y),r)=1]\end{array}\right| \geq \frac{1}{p(m)}.$$

Let $M$ denote the set of integers $m$ for which the above inequality holds; and let $\epsilon(m)$ denote the difference (inside the absolute value), and assume, without loss of generality, that it is positive. Using the above $D$ we first prove the following.

*Claim.* There exists a probabilistic polynomial-time algorithm $A$ that satisfies

$$\mathbf{P}_{x,y\in\mathrm{GF}(2^{\ell(m)})^m}[A(f(x,y)) = h(x,y)] \geq 2^{-\ell(m)} + \frac{\epsilon(m)}{2^{\ell(m)}-1}$$

for every $m \in M$.

*Proof.* The claim may be established by analyzing the success probability of the following algorithm $A$: On input $z = f(x,y)$, uniformly select $r \in \mathrm{GF}(2^{\ell(m)})$, invoke $D(z,r)$, and return $r$ if $D(z,r) = 1$ and a uniformly selected value in $\mathrm{GF}(2^{\ell(m)}) \setminus \{r\}$ otherwise. Details follow.

$$\begin{aligned}\mathbf{P}_{x,y}[A(f(x,y)) = h(x,y)] &= \mathbf{P}_{x,y,r}[A(f(x,y)) = h(x,y) \ \&\ r = h(x,y)] \\ &\quad + \mathbf{P}_{x,y,r}[A(x,y) = h(x,y) \ \&\ r \neq h(x,y)] \\ &= \mathbf{P}_{x,y,r}[D(f(x,y),r) = 1 \ \&\ r = h(x,y)] \\ &\quad + \mathbf{P}_{x,y,r,r'}[D(f(x,y),r) = 0 \ \&\ r' = h(x,y) \neq r],\end{aligned}$$

where $r'$ denotes a uniformly selected value in $\mathrm{GF}(2^{\ell(m)}) \setminus \{r\}$. Letting $L = 2^{\ell(m)}$, we have

$$\begin{aligned}\mathbf{P}_{x,y,r}[D(f(x,y),r) = 1 \ \&\ r = h(x,y)] &= \frac{1}{L}\cdot\mathbf{P}_{x,y,r}[D(f(x,y),r) = 1 \,|\, r = h(x,y)] \\ &= \frac{1}{L}\cdot\mathbf{P}_{x,y,r}[D(f(x,y),h(x,y)) = 1].\end{aligned}$$

On the other hand

$$\begin{aligned}
&\mathbf{P}_{x,y,r,r'}[D(f(x,y),r)=0 \;\&\; r'=h(x,y)\neq r]\\
&= \frac{1}{L-1}\cdot \mathbf{P}_{x,y,r}[D(f(x,y),r)=0 \;\&\; r\neq h(x,y)]\\
&= \frac{1}{L-1}\cdot (\mathbf{P}_{x,y,r}[D(f(x,y),r)=0]-\mathbf{P}_{x,y,r}[D(f(x,y),r)=0 \;\&\; r=h(x,y)])\\
&= \frac{1}{L-1}\cdot \left(1-\mathbf{P}_{x,y,r}[D(f(x,y),r)=1]-\frac{1}{L}+\frac{1}{L}\cdot \mathbf{P}_{x,y}[D(f(x,y),h(x,y))=1]\right).
\end{aligned}$$

Combining the above, we get

$$\begin{aligned}
&\mathbf{P}_{x,y}[A(f(x,y))=h(x,y)]\\
&= \frac{1}{L}\cdot \mathbf{P}_{x,y,r}[D(f(x,y),h(x,y))=1]\\
&\quad +\frac{1}{L}+\frac{1}{L-1}\cdot\left(\frac{1}{L}\cdot \mathbf{P}_{x,y}[D(f(x,y),h(x,y))=1]-\mathbf{P}_{x,y,r}[D(f(x,y),r)=1]\right)\\
&= \frac{1}{L}+\frac{1}{L-1}\cdot(\mathbf{P}_{x,y}[D(f(x,y),h(x,y))=1]-\mathbf{P}_{x,y,r}[D(f(x,y),r)=1])\\
&\geq \frac{1}{L}+\frac{1}{L-1}\cdot\epsilon(m),
\end{aligned}$$

and the claim follows. ☐

Let $A$ be as in the above claim. Recalling that $f(x,y)=(f'(x),y)$, observe that $A$ gives rise to a function $F_{z'}:\mathrm{GF}(2^{\ell(m)})^m\to\mathrm{GF}(2^{\ell(m)})$ defined by $F_{z'}(y)\stackrel{\mathrm{def}}{=}A(z',y)$, and it holds that

$$\mathbf{P}_{x,y\in\mathrm{GF}(2^{\ell(m)})^m}\left[F_{f'(x)}(y)=\sum_{i=1}^m x_iy_i\right]\geq 2^{-\ell(m)}+\frac{\epsilon(m)}{2^{\ell(m)}-1}.$$

Thus, for at least an $\frac{\epsilon(m)}{2^{\ell(m)+1}}$ fraction of the possible $x=(x_1,\ldots,x_m)$'s it holds that

$$\mathbf{P}_{y\in\mathrm{GF}(2^{\ell(m)})^m}\left[F_{f'(x)}(y)=\sum_{i=1}^m x_iy_i\right]\geq 2^{-\ell(m)}+\frac{\epsilon(m)}{2^{\ell(m)+1}}.$$

Applying Theorem 2.1 to such $F_{f'(x)}$, with $\delta=2^{-\ell(m)}+\frac{\epsilon(m)}{2^{\ell(m)+1}}$, we obtain a list of all polynomials that agree with $F_{f'(x)}$ on at least a $2^{-\ell(m)}+\frac{\epsilon(m)}{2^{\ell(m)+1}}$ fraction of the inputs. This list includes $x$, and hence we have inverted the function $f'$ in time $\mathrm{poly}(m/\delta)=\mathrm{poly}(|x|)$. This happens on a polynomial fraction of the possible $x$'s, and thus we have reached a contradiction to the hypothesis that $f'$ is a one-way function. ☐

**Acknowledgments.** We are grateful to Mike Kearns and Dana Ron for discussion regarding agnostic learning. We thank Eyal Kushilevitz and Yishay Mansour for comments regarding an early version of this paper. We thank Salil Vadhan for clarifying discussions. We thank Felix Wu for pointing out an error in a previous version of the paper. We thank the anonymous referees for pointing out numerous errors and other helpful comments.

## REFERENCES

[1] D. Angluin and P. Laird, *Learning from noisy examples*, Machine Learning, 2 (1988), pp. 343–370.

[2] D. Angluin and M. Kriķis, *Learning with malicious membership queries and exceptions*, in Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory, New Brunswick, NJ, ACM Press, New York, 1994, pp. 57–66.

[3] S. Ar, R. J. Lipton, R. Rubinfeld, and M. Sudan, *Reconstructing algebraic functions from mixed data*, SIAM J. Comput., 28 (1999), pp. 487–510.

[4] S. Arora and M. Sudan, *Improved low degree testing and its applications*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, TX, 1997, ACM Press, New York, pp. 485–495.

[5] D. Beaver and J. Feigenbaum, *Hiding instances in multioracle queries*, in 7th Annual Symposium on Theoretical Aspects of Computer Science, Rouen, France, Lecture Notes in Comput. Sci., 415, Springer, New York, 1990, pp. 37–48.

[6] A. Blum and P. Chalasani, *Learning switching concepts*, in Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, ACM Press, Pittsburgh, PA, 1992, pp. 231–242.

[7] A. Blum, M. Furst, M. Kearns, and R. J. Lipton, *Cryptographic primitives based on hard learning problems*, in Advances in Cryptology—CRYPTO '93, D. R. Stinson, ed., Lecture Notes in Comput. Sci. 773, Springer-Verlag, New York, pp. 278–291.

[8] M. Blum, M. Luby, and R. Rubinfeld, *Self-testing/correcting with applications to numerical problems*, J. Comput. System Sci., 47 (1993), pp. 549–595.

[9] R. Cleve and M. Luby, *A Note on Self-Testing/Correcting Methods for Trigonometric Functions*, International Computer Science Institute Technical Report TR-90-032, ICSI, Berkeley, CA, 1990.

[10] R. A. DeMillo and R. J. Lipton, *A probabilistic remark on algebraic program testing*, Inform. Process. Lett., 7 (1978), pp. 193–195.

[11] P. Elias, *List decoding for noisy channels*, in 1957-IRE WESCON Convention Record, Part 2, IRE-IEEE Press, New York, 1957, pp. 94–104.

[12] Y. Freund and D. Ron, *Learning to model sequences generated by switching distributions*, in Proceedings of the Eighth Annual Conference on Computational Learning Theory, ACM Press, Santa Cruz, CA, 1995, pp. 41–50.

[13] S. A. Goldman, M. J. Kearns, and R. E. Schapire, *Exact identification of read-once formulas using fixed points of amplification functions*, SIAM J. Comput., 22 (1993), pp. 705–726.

[14] P. Gemmell, R. J. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson, *Self-testing/correcting for polynomials and for approximate functions*, in Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, New Orleans, LA, ACM Press, New York, 1991, pp. 32–42.

[15] P. Gemmell and M. Sudan, *Highly resilient correctors for polynomials*, Inform. Process. Lett., 43 (1992), pp. 169–174.

[16] O. Goldreich, *Foundations of Cryptography—Fragments of a Book*, Weizmann Institute of Science, Rehovot, Israel, 1995. Available from the ECCC, http://www.eccc.uni-trier.de/eccc/.

[17] O. Goldreich and L. A. Levin, *A hard-core predicate for all one-way functions*, in Proceedings of the 21st Annual ACM Symposium on Theory of Computing, Seattle, WA, ACM Press, New York, 1989, pp. 25–32.

[18] O. Goldreich, R. Rubinfeld, and M. Sudan, *Learning polynomials with queries: The highly noisy case*, in 36th Annual Symposium on Foundations of Computer Science, IEEE, Milwaukee, WI, IEEE, New York, 1995, pp. 294–303.

[19] V. Guruswami and M. Sudan, *Improved decoding of Reed-Solomon and algebraic-geometric codes*, IEEE Trans. Inform. Theory, 45 (1999), pp. 1757–1767.

[20] M. Kearns, *Efficient noise-tolerant learning from statistical queries*, J. ACM, 45 (1998), pp. 983–1006.

[21] M. Kearns and M. Li, *Learning in the presence of malicious errors*, SIAM J. Comput., 22 (1993), pp. 807–837.

[22] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie, *On the learnability of discrete distributions (extended abstract)*, in Proceedings of the 26th Annual ACM Symposium on the Theory of Computing, Montreal, Canada, ACM Press, New York, 1994, pp. 273–282.

[23] M. J. Kearns, R. E. Schapire, and L. M. Sellie, *Toward efficient agnostic learning (extended abstract)*, in Proceedings of the 5th Annual ACM Workshop on Computational Learning

Theory, Pittsburgh, PA, ACM Press, New York, 1992.

[24] P. Koiran, *Efficient learning of continuous neural networks*, in Proceedings of the 7th Annual ACM Conference on Computational Learning Theory, New Brunswick, NJ, ACM Press, New York, 1994, pp.348–355.

[25] E. Kushilevitz and Y. Mansour, *Learning decision trees using the Fourier spectrum*, SIAM J. Comput., 22 (1993), pp. 1331–1348.

[26] P. D. Laird, *Learning From Good and Bad Data*, Kluwer Academic Publishers, Boston, 1988.

[27] L. A. Levin, *Randomness and non-determinism*, J. Symbolic Logic, 58 (1993), pp. 1102–1103.

[28] R. J. Lipton, *New directions in testing*, in Distributed Computing and Cryptography: Proceedings of a DIMACS Workshop, J. Feigenbaum and M. Merritt, eds., AMS, Providence, RI, 1991, pp. 191–202.

[29] W. Maass, *Efficient agnostic PAC-learning with simple hypotheses*, in Proceedings of the 7th Annual ACM Conference on Computational Learning Theory, New Brunswick, NJ, ACM Press, New York, 1994, pp. 67–75.

[30] W. Maass, *Agnostic PAC-learning of functions on analog neural nets*, in Proceedings of the 6th Conference on Advances in Neural Information Processing Systems, J. D. Cowan, G. Tesauro, and J. Alspector, eds., Morgan Kaufmann, San Mateo, CA, 1994, pp. 311–318.

[31] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1981.

[32] Y. Mansour, *Randomized interpolation and approximation of sparse polynomials*, SIAM J. Comput., 24 (1995), pp. 357–368.

[33] D. Ron and R. Rubinfeld, *Learning fallible deterministic finite automata*, Machine Learning 18 (1995), pp. 149–185.

[34] R. Rubinfeld, *On the robustness of functional equations*, SIAM J. Comput., 28 (1999), pp. 1972–1997.

[35] R. Rubinfeld and M. Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM J. Comput., 25 (1996), pp. 252–271.

[36] Y. Sakakibara, *On learning from queries and counterexamples in the presence of noise*, Inform. Process. Lett., 37 (1991), pp. 279–284.

[37] Y. Sakakibara and R. Siromoney, *A noise model on learning sets of strings*, in Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, ACM Press, Pittsburgh, PA, ACM Press, New York, 1992, pp. 295–302.

[38] J. T. Schwartz, *Fast probabilistic algorithms for verification of polynomial identities*, J. ACM, 27 (1980), pp. 701–717.

[39] R. Sloan, *Types of noise in data for concept learning (extended abstract)*, in Proceedings of the 1988 Workshop on Computational Learning Theory, Cambridge, MA, Morgan Kaufmann, San Mateo, CA, 1988, pp. 91–96.

[40] M. Sudan, *Decoding of Reed Solomon codes beyond the error-correction bound*, J. Complexity, 13 (1997), pp. 180–193.

[41] M. Sudan, L. Trevisan, and S. Vadhan, *Pseudorandom generators without the XOR lemma*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, Atlanta, GA, ACM Press, New York, 1999, pp. 537–546.

[42] L. G. Valiant, *Learning disjunctions of conjunctions*, in Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, 1985, Morgan Kauffman Publishers, San Mateo, CA, 1985, pp. 560–566.

[43] U. V. Vazirani and V. V. Vazirani, *Efficient and secure pseudo-random number generation*, in Proceedings of the 25th IEEE Symposium on Foundations of Computer Science, Singer Island, FL, 1984, IEEE, New York, 1984, pp. 458–463.

[44] H. Wasserman, *Reconstructing randomly sampled multivariate polynomials from highly noisy data*, in Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, ACM Press, New York, SIAM, Philadelphia, 1998, pp. 59–67.

[45] L. R. Welch and E. R. Berlekamp, *Error Correction of Algebraic Block Codes*, US patent 4,633,470, December 30, 1986.

[46] R. E. Zippel, *Probabilistic algorithms for sparse polynomials*, in Proceedings of EUROSAM '79: International Symposium on Symbolic and Algebraic Manipulation, E. Ng, ed., Lecture Notes in Comput. Sci., 72, Springer, New York, 1979, pp. 216–226.

[47] R. E. Zippel, *Interpolating polynomials from their values*, J. Symbolic Comput., 9 (1990), pp. 375–403.

[48] R. E. Zippel, *Effective Polynomial Computation*, Kluwer Academic Publishers, Boston, 1993.

SIAM J. DISCRETE MATH.
Vol. 13, No. 4, pp. 571–575

# UNIQUENESS OF SOME RESOLUTION IV TWO-LEVEL REGULAR FRACTIONAL FACTORIAL DESIGNS[*]

HEGANG CHEN[†] AND CHING-SHUI CHENG[‡]

**Abstract.** It is well known that if there exists a resolution IV $2^{n-m}$ design, then $n \leq 2^{n-m-1}$. A resolution IV $2^{n-m}$ design with $n = 2^{n-m-1}$ is known to be unique in the sense that all such designs can be obtained from one another by relabeling the factors. By using a projective geometric approach, we show that resolution IV $2^{n-m}$ designs with $n = 2^{n-m-1} - 1$ and $n = 2^{n-m-1} - 2$ are also unique.

**Key words.** experimental design, projective geometry, resolution, saturated design

**AMS subject classifications.** 62K15, 05B25

**PII.** S0895480199363544

## 1. Introduction.

Two-level regular fractional factorial designs are among the most widely used experimental designs. Suppose there are $n$ two-level factors. Denote the two levels by 0 and 1, and identify them with the elements of GF(2), the finite field with two elements. Then the set of all the $2^n$ level combinations is conveniently represented by the $n$-dimensional linear space $\{\boldsymbol{x} : \boldsymbol{x} = (x_1, \dots, x_n), x_i = 0, 1\}$ over GF(2), denoted by $V^n$. Without loss of generality, a $2^{n-m}$ regular fractional factorial design can be defined as an $(n-m)$-dimensional subspace of $V^n$, or equivalently, the solution set of $m$ simultaneous linear equations

$$\boldsymbol{x}\boldsymbol{b}_i = \boldsymbol{0}, \quad i = 1, \dots, m, \tag{1}$$

where $\boldsymbol{b}_1, \dots, \boldsymbol{b}_m$ are linearly independent vectors in $V^n$, written as column vectors. Let $\boldsymbol{W}$ be the $n \times (2^m - 1)$ matrix whose columns are all the nonnull linear combinations of $\boldsymbol{b}_1, \dots, \boldsymbol{b}_m$. Then (1) is equivalent to

$$\boldsymbol{x}\boldsymbol{W} = \boldsymbol{0}. \tag{2}$$

In factorial design literature, (2) is called the defining relation of the design, and each column of $\boldsymbol{W}$ is referred to as a defining word. The number of nonzero components that a defining word contains is called its length (or weight in coding theory literature), and the *resolution* of a design is defined as the length of a shortest defining word. This important concept, introduced by Box and Hunter (1961a, 1961b), is useful for classifying fractional factorial designs.

It is well known that if there exists a resolution III $2^{n-m}$ design, then $n \leq 2^{n-m} - 1$, and if a resolution IV $2^{n-m}$ design exists, then $n \leq 2^{n-m-1}$; these follow, for example, from the two Rao's inequalities on p. 13 of Hedayat, Sloane, and Stufken

[*]Received by the editors October 27, 1999; accepted for publication (in revised form) August 14, 2000; published electronically November 27, 2000. This research was supported by the National Science Foundation. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allows others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.
http://www.siam.org/journals/sidma/13-4/36354.html

[†]Division of Biostatistics, University of Minnesota, MMC 303, 420 Delaware St. S.E., Minneapolis, MN 55455-0378 (hegangc@biostat.umn.edu).
[‡]Department of Statistics, University of California, Berkeley, CA 94720-3860 (cheng@stat.berkeley.edu).

(1999). A resolution III $2^{n-m}$ design with $n = 2^{n-m} - 1$ is called *saturated*. Saturated resolution III designs are unique in the sense that they can be obtained from one another by relabeling the factors. We shall also call a resolution IV $2^{n-m}$ design with $n = 2^{n-m-1}$ saturated. (In this case, the number of factors is exactly half of the run size.) It follows from Margolin (1969) that a saturated resolution IV $2^{n-m}$ design must be the *foldover* of a saturated resolution III $2^{(n-1)-m}$ design. Thus saturated resolution IV $2^{n-m}$ designs are also unique. In this note, we shall show the uniqueness of some nearly saturated resolution IV $2^{n-m}$ designs. Specifically, we show that resolution IV $2^{n-m}$ designs with $n = 2^{n-m-1} - 1$ (respectively, $n = 2^{n-m-1} - 2$) must be obtained by deleting one factor (respectively, two factors) from a saturated resolution IV design, when $n - m \geq 3$ (respectively, $n - m \geq 4$).

In general, for $n = 2^{n-m-1} - q$, a design obtained by deleting $q$ factors from a saturated resolution IV design with $2^{n-m}$ runs is of resolution at least IV. However, when $q > 2$, this is not the only design of resolution at least IV. Clearly, for $q > 2$, there is more than one way to delete $q$ factors from a saturated resolution IV design. A more fundamental question is whether a design of resolution at least IV can always be obtained by deleting $q$ factors from a saturated resolution IV design. The answer to this general question is no, but it holds when $q$ is small relative to $2^{n-m-1}$.

Connections between fractional factorial designs and finite projective geometry is well known (Bose (1947)). In section 2, we briefly review this connection and show that our uniqueness results are equivalent to the following theorem which is proved in section 3.

THEOREM. *Let $G$ be a $PG(k-1, 2)$, a $(k-1)$-dimensional projective geometry based on $GF(2)$, and $t = 2^{k-1} - 1 + q$, $q \geq 1$.*

(a) *For $k \geq 3$ and $q = 1$, a $t$-subset of $G$ contains the maximum number of lines if and only if it is obtained by adding an arbitrary point to a $(k-2)$-flat in $G$.*

(b) *For $k \geq 4$ and $q = 2$, a $t$-subset of $G$ contains the maximum number of lines if and only if it is obtained by adding two arbitrary points to a $(k-2)$-flat in $G$.*

Recall that an $f$-flat consists of all the $2^{f+1}-1$ nonnull linear combinations of $f+1$ linearly independent points of a projective geometry. Some recent papers exploiting the connection between fractional factorial designs and projective geometry include Chen and Hedayat (1996), Chen and Cheng (1999), Cheng and Mukerjee (1997, 1998), Mukerjee and Wu (1999). Chen and Hedayat (1996) is particularly relevant since it also considered the problem of maximizing the number of lines.

Throughout this note, $|A|$ denotes the cardinality of a set $A$.

## 2. A projective-geometric formulation.

Let $k = n - m$ and $G$ be a $PG(k-1, 2)$. Each of the $g$ points of $G$, where $g = 2^k - 1$, can be represented by a nonnull $k \times 1$ vector. Let $T$ be an $n$-subset of $G$ and $V_T$ be the $k \times n$ matrix with the points of $T$ as its columns. If $V_T$ has full row rank, then its row space gives a regular $2^{n-m}$ design. Thus each regular $2^{n-m}$ design corresponds to an $n$-subset of $PG(k-1, 2)$, where $k = n - m$. A defining word of length $t$ corresponds to $t$ of the $n$ chosen points, say, $\boldsymbol{\alpha}_{i_1}, \ldots, \boldsymbol{\alpha}_{i_t}$, with $\boldsymbol{\alpha}_{i_1} + \cdots + \boldsymbol{\alpha}_{i_t} = \mathbf{0}$. Thus, for example, a defining word of length three corresponds to three collinear points and constructing a $2^{n-m}$ design with resolution at least four amounts to choosing a set of $n$ points from $PG(k-1, 2)$ which contains no lines, where $k = n - m$.

When all the $2^k - 1$ points of a $PG(k-1, 2)$ are chosen, one obtains a saturated resolution III design. A saturated resolution IV design can also be constructed easily. Let $A$ be a $(k-2)$-flat in $G$. Then $G = A \cup \{\boldsymbol{a}\} \cup (\boldsymbol{a} + A)$ for a point $\boldsymbol{a} \notin A$. It is clear that $\{\boldsymbol{a}\} \cup (\boldsymbol{a} + A)$ contains no lines; consequently choosing all the $2^{k-1}$ points in $\{\boldsymbol{a}\} \cup (\boldsymbol{a} + A)$ leads to a resolution IV design with $2^{k-1}$ factors, which is saturated. In other words, a saturated resolution IV $2^{n-m}$ design can be obtained by choosing all the points in the *complement of a* $(k-2)$*-flat*.

Now suppose $n = 2^{k-1} - q$, $q \geq 1$. Let $A$ be obtained by adding $q$ points to a $(k-2)$-flat. Then $\overline{A}$, the complement of $A$, has $2^{k-1} - q$ points and obviously contains no lines. Hence $\overline{A}$ defines a $2^{n-m}$ design for $n = 2^{k-1} - q$ factors with resolution at least IV. The question raised in the introduction is essentially whether this is the only way to choose an $n$-subset which contains no lines. As remarked there and will be shown in the following, the answer is yes for $q = 1$ and 2 when $k \geq 3$ and 4, respectively.

For any subset $A$ of $G$, let $N(G)$ and $N(A)$ be the numbers of lines contained in $G$ and $A$, respectively, and let $N(A, \overline{A})$ be the number of lines with at least one point in each of $A$ and $\overline{A}$. Then

$$N(G) = N(A) + N(\overline{A}) + N(A, \overline{A}). \tag{3}$$

Since $N(A, \overline{A}) = \frac{1}{2}|A||\overline{A}|$ and $N(G)$ is a constant, when $|\overline{A}| \leq 2^{k-1}$, $N(\overline{A}) = 0$ if and only if $A$ maximizes $N(A)$. Thus an equivalent problem is, for $q = 1$ and 2, whether the only way to maximize $N(A)$ over all $(2^{k-1} - 1 + q)$-subsets of $G$ is to let $A$ be a $(k-2)$-flat supplemented by $q$ points. This is what the theorem in the introduction is about. Note that for $q = 0$ (the saturated case), $N(A)$ is obviously maximized by a $(k-2)$-flat, thus providing a geometric proof of the uniqueness of saturated resolution IV designs; see Bose (1947) and Chen (1993).

In the rest of the paper, $\langle H \rangle$ denotes the flat generated by a subset $H$ of $G$. A subset of $G$ is said to be of rank $p$ if it contains $p$, but not more than $p$, linearly independent points.

## 3. Proof of the theorem.

We first prove several lemmas.

LEMMA 1. *Suppose $G$ is a $PG(k-1, 2)$, $t = 2^{k-1} - 1 + q$, $q \geq 1$, and $A$ is a $t$-subset of $G$. Then there exists an $H \subset A$ such that $H$ has rank $k-1$, $H = A \cap \langle H \rangle$ and $A = H \cup \{\boldsymbol{a}, \boldsymbol{a} + \boldsymbol{b}_1, \ldots, \boldsymbol{a} + \boldsymbol{b}_s\}$, where $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s \in \langle H \rangle$, and $\boldsymbol{a} \notin \langle H \rangle$.*

*Proof.* First of all, $A$ must contain $k$ linearly independent points, say, $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$; for otherwise, $t = |A| \leq 2^{k-1} - 1$, a contradiction. Let $H = A \cap \langle \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k-1} \rangle$ and $\boldsymbol{a} = \boldsymbol{a}_k$. Then $H$ has rank $k-1$, and $\langle H \rangle = \langle \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k-1} \rangle$ is a $(k-2)$-flat. It follows that each $\boldsymbol{x} \in A \backslash H$ with $\boldsymbol{x} \neq \boldsymbol{a}$ has the form $\boldsymbol{a} + \boldsymbol{b}$ for some $\boldsymbol{b} \in \langle H \rangle$. □

LEMMA 2. *In Lemma 1, suppose $A$ contains the maximum number of lines among all $t$-subsets of $G$. Then the complement $\overline{A}$ of $A$ in $G$ contains no lines.*

*Proof.* This has been shown in the sentence following (3).

LEMMA 3. *In Lemma 1, suppose $A$ contains the maximum number of lines among all $t$-subsets of $G$. Then the set $H$ has the following properties:*

(i) *$\langle H \rangle \backslash H$ contains no lines,*

(ii) *$|H| \geq 2^{k-2} - 1$ (so $s \leq 2^{k-2} + q - 1$), and*

(iii) *if $|H| \neq 2^{k-1} - 1$ (i.e., $H$ is not a $(k-2)$-flat), then $|H| \leq 2^{k-2} + q - 1$ (i.e., $s \geq 2^{k-2} - 1$).*

*Proof.* Since $\langle H \rangle \backslash H \subset \overline{A}$, by Lemma 2, it contains no lines. This proves (i). Then we must have $\boldsymbol{y}$, $\boldsymbol{z} \in \langle H \rangle \backslash H \Rightarrow \boldsymbol{y} + \boldsymbol{z} \in H$. Therefore $|\langle H \rangle \backslash H| \leq |H| + 1$. Thus (ii) follows from $|\langle H \rangle| = 2^{k-1} - 1$.

Suppose $|H| \neq 2^{k-1} - 1$. Then $\langle H \rangle$ contains at least one point $\boldsymbol{x}$ such that $\boldsymbol{x} \notin A$. Let $C = \{\boldsymbol{a}, \boldsymbol{a} + \boldsymbol{b}_1, \ldots, \boldsymbol{a} + \boldsymbol{b}_s\}$ and $D$ be the complement of $C$ in $\{\boldsymbol{a}\} \cup (\boldsymbol{a} + \langle H \rangle)$. Then since $\overline{A}$ contains no line, we must have $\boldsymbol{x} + \boldsymbol{y} \in C$ for all $\boldsymbol{y} \in D$.

Therefore $|C| \geq |D|$. Since $|C \cup D| = 2^{k-1}$, $|C| \geq 2^{k-2}$; therefore $s \geq 2^{k-2} - 1$, proving (iii). $\square$

LEMMA 4. *In Lemma* 1*, suppose $A$ contains the maximum number of lines among all $t$-subsets of $G$. Let $E$ be a $(k-3)$-flat in $\langle H\rangle$. If neither $E$ nor $\langle H\rangle\backslash E$ is a subset of $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$, then $|A \cap (\langle H\rangle\backslash E)| \geq 2^{k-2} - \frac{1}{2}(s+1)$.*

*Proof.* Since $G\backslash\langle H\rangle = \{\boldsymbol{a}\} \cup (\boldsymbol{a} + \langle H\rangle)$, we can decompose $\overline{A} \cap (G\backslash\langle H\rangle)$ as $P \cup Q$, where $P = \overline{A} \cap (\boldsymbol{a} + E)$ and $Q = \overline{A} \cap (\boldsymbol{a} + (\langle H\rangle\backslash E))$. By the assumption that neither $E$ nor $\langle H\rangle\backslash E$ is a subset of $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$ and noting that $P \cup Q$ is the complement of $\{\boldsymbol{a}, \boldsymbol{a}+\boldsymbol{b}_1, \ldots, \boldsymbol{a}+\boldsymbol{b}_s\}$ in $G\backslash\langle H\rangle$, we see that both $P$ and $Q$ are nonempty. Then since $\overline{A}$ contains no lines, and $\boldsymbol{y} + \boldsymbol{z} \in \langle H\rangle\backslash E$ for all $\boldsymbol{y} \in \boldsymbol{a} + E$ and $\boldsymbol{z} \in \boldsymbol{a} + (\langle H\rangle\backslash E)$, we must have $\boldsymbol{y} + \boldsymbol{z} \in A \cap (\langle H\rangle\backslash E)$ for all $\boldsymbol{y} \in P$ and $\boldsymbol{z} \in Q$. It follows that $|A \cap (\langle H\rangle\backslash E)| \geq \max\{|P|, |Q|\} \geq \frac{1}{2}|P \cup Q| = \frac{1}{2}|\overline{A} \cap (G\backslash\langle H\rangle)| = \frac{1}{2}(2^{k-1} - s - 1)$. $\square$

We shall prove the theorem by induction, first establishing the validity of part (a) for $k = 3$ and part (b) for $k = 4$. Also, it is enough to prove the necessity part since the sufficiency has already been established by the construction in section 2.

Part (a) of the theorem holds for $k = 3$ trivially. We now prove that (b) holds for $k = 4$. In this case, $A$ has nine points. It suffices to show that $A$ contains a 2-flat.

Let $H$ be a set of rank three as in Lemma 1. If $H$ is itself a 2-flat, then there is nothing to prove. Otherwise, by Lemma 3, $3 \leq |H| \leq 5$. On the other hand, by Lemma 3(i), $\langle H\rangle\backslash H$ contains no lines; therefore $H$ must contain at least one line. Then the case $|H| = 3$ is ruled out since $H$ also needs to contain three linearly independent points, which cannot be collinear.

*Case* 1. $|H| = 4$. In this case, $s = 4$, and $H$ must be a line supplemented by a point, say, $H = \{\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{u}+\boldsymbol{v}, \boldsymbol{w}\}$. Let $E$ be the line $\{\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{u}+\boldsymbol{v}\}$. Then since $|A \cap (\langle H\rangle\backslash E)| = |\{\boldsymbol{w}\}| = 1$, which is less than $2^{k-2} - \frac{1}{2}(s+1) = 1.5$, by Lemma 4, either $E$ or $\langle H\rangle\backslash E$ is a subset of $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$. In both cases, it is easy to see that $A$ contains a 2-flat.

*Case* 2. $|H| = 5$. In this case, $H$ is obtained by deleting two points from a 2-flat. Without loss of generality, we may assume that $H = \{\boldsymbol{w}, \boldsymbol{u}+\boldsymbol{v}, \boldsymbol{u}+\boldsymbol{w}, \boldsymbol{v}+\boldsymbol{w}, \boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w}\}$, where $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{w}$ are linearly independent. Let $E$ be the line consisting of the three points $\boldsymbol{u}+\boldsymbol{v}, \boldsymbol{u}+\boldsymbol{w}$, and $\boldsymbol{v}+\boldsymbol{w}$. Then $\langle H\rangle\backslash E = \{\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w}\}$ and $A \cap (\langle H\rangle\backslash E) = \{\boldsymbol{w}, \boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w}\}$. Let $P$ and $Q$ be as defined in the proof of Lemma 4. In this case, $\langle H\rangle\backslash E$ cannot be a subset of $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$, since $|\langle H\rangle\backslash E| = 4$ and $s = 3$. We may also assume that $E \neq \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$, for otherwise $A$ would contain the 2-flat generated by $\boldsymbol{a}, \boldsymbol{u}+\boldsymbol{w}$ and $\boldsymbol{v}+\boldsymbol{w}$, and there would be nothing to prove. So as in the proof of Lemma 4, we have $\max\{|P|, |Q|\} \leq |A \cap (\langle H\rangle\backslash E)| = 2$. On the other hand, $|P \cup Q| = 4$. Therefore $|P| = |Q| = 2$. Let $P = \{\boldsymbol{a}+\boldsymbol{c}_1, \boldsymbol{a}+\boldsymbol{c}_2\}$ and $Q = \{\boldsymbol{a}+\boldsymbol{c}_3, \boldsymbol{a}+\boldsymbol{c}_4\}$, where $\boldsymbol{a} \notin \langle H\rangle$, $\boldsymbol{c}_1, \boldsymbol{c}_2 \in E$, and $\boldsymbol{c}_3, \boldsymbol{c}_4 \in \langle H\rangle\backslash E$. Since $\boldsymbol{y} \in P$ and $\boldsymbol{z} \in Q \Rightarrow \boldsymbol{y}+\boldsymbol{z} \in A \cap (\langle H\rangle\backslash E)$ (see the proof of Lemma 4), we must have $\{\boldsymbol{c}_1+\boldsymbol{c}_3, \boldsymbol{c}_1+\boldsymbol{c}_4\} = \{\boldsymbol{c}_2+\boldsymbol{c}_3, \boldsymbol{c}_2+\boldsymbol{c}_4\} = A \cap (\langle H\rangle\backslash E)$, which is $\{\boldsymbol{w}, \boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w}\}$. In particular, we must have $\{\boldsymbol{c}_1+\boldsymbol{w}, \boldsymbol{c}_1+\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w}\} = \{\boldsymbol{c}_2+\boldsymbol{w}, \boldsymbol{c}_2+\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w}\}$. By checking all three ways of choosing two points from $E$, we conclude that this equality holds only for $\{\boldsymbol{c}_1, \boldsymbol{c}_2\} = \{\boldsymbol{u}+\boldsymbol{w}, \boldsymbol{v}+\boldsymbol{w}\}$. Therefore $\{\boldsymbol{c}_3, \boldsymbol{c}_4\} = \{\boldsymbol{u}, \boldsymbol{v}\}$, and so $A = \{\boldsymbol{w}, \boldsymbol{u}+\boldsymbol{v}, \boldsymbol{u}+\boldsymbol{w}, \boldsymbol{v}+\boldsymbol{w}, \boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{a}+\boldsymbol{u}+\boldsymbol{v}, \boldsymbol{a}+\boldsymbol{w}, \boldsymbol{a}+\boldsymbol{u}+\boldsymbol{v}+\boldsymbol{w}\}$, which contains the 2-flat generated by $\boldsymbol{a}, \boldsymbol{w}$, and $\boldsymbol{u}+\boldsymbol{v}$. $\square$

Now we are ready to finish the proof of the theorem by induction. Assume that the necessity part in the theorem holds for all $k' < k$, where $k > 4$ for $q = 2$ and $k > 3$ for $q = 1$. We shall show that $A$ contains a $(k-2)$-flat.

Let $H$ be as in Lemma 1. If $H$ is itself a $(k-2)$-flat, then there is nothing to prove. Otherwise, by Lemma 3, $2^{k-2} - 1 \leq |H| \leq 2^{k-2} + q - 1$, where $q = 1$ or 2. Suppose $|H| = 2^{k-2} - 1 + u$, where $0 \leq u \leq q$. By Lemma 3(i), $\langle H \rangle \backslash H$ contains no lines. Hence $H$ maximizes the number of lines among all the $(2^{k-2} - 1 + u)$-subsets of $\langle H \rangle$. By the induction hypothesis, $H$ is obtained by adding $q$ points to a $(k-3)$-flat in $\langle H \rangle$. Let $E$ be this $(k-3)$-flat. Then $|A \cap (\langle H \rangle \backslash E)| = u$. On the other hand, $s = 2^{k-2} + q - 1 - u$. We have $|A \cap (\langle H \rangle \backslash E)| < 2^{k-2} - \frac{1}{2}(s+1)$ for $k > 3$, $q = 1$ and $k > 4$, $q = 2$. It follows from Lemma 4 that either $E$ or $\langle H \rangle \backslash E$ is a subset of $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$. In both cases, $A$ clearly contains a $(k-2)$-flat. $\square$

## REFERENCES

R. C. Bose (1947), *Mathematical theory of the symmetrical factorial design*, Sankhyā, 8, pp. 107–166.

G. E. P. Box and J. S. Hunter (1961a), *The* $2^{k-p}$ *fractional factorial designs*, I, Technometrics, 3, pp. 311–351.

G. E. P. Box and J. S. Hunter (1961b), *The* $2^{k-p}$ *fractional factorial designs*, II, Technometrics, 3, pp. 449–458.

H. Chen (1993), *Contributions to Experimental Designs*, Ph.D. dissertation, University of Illinois at Chicago, Chicago, IL.

H. Chen and C. S. Cheng (1999), *Theory of optimal blocking of* $2^{n-m}$ *designs*, Ann. Statist., 27, pp. 1948–1973.

H. Chen and A. S. Hedayat (1996), $2^{n-m}$ *fractional factorial designs with weak minimum aberration*, Ann. Statist., 24, pp. 2536–2548.

C. S. Cheng and R. Mukerjee (1998), *Regular fractional factorial designs with minimum aberration and maximum estimation capacity*, Ann. Statist., 26, pp. 2289–2300.

C. S. Cheng and R. Mukerjee (1997), *Blocked regular fractional factorial designs with maximum estimation capacity*, revised for Ann. Statist.

A. S. Hedayat, N. J. A. Sloane, and J. Stufken (1999), *Orthogonal Arrays: Theory and Applications*, Springer-Verlag, New York.

B. H. Margolin (1969), *Resolution* IV *fractional factorial designs*, J. Roy. Statist. Soc. Ser. B, 31, pp. 514–523.

R. Mukerjee and C. F. J. Wu (1999), *Blocking in regular fractional factorials: A projective geometric approach*, Ann. Statist., 27, pp. 1256–1271.