SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 1-16, February 1995

# PERFORMANCE GUARANTEES FOR APPROXIMATION ALGORITHMS DEPENDING ON PARAMETRIZED TRIANGLE INEQUALITIES *

THOMAS ANDREAE† AND HANS-JÜRGEN BANDELT†

**Abstract.** The worst-case analyses of heuristics in combinatorial optimization are often far too pessimistic when confronted with performance on real-world problems. One approach to partially overcome this discrepancy is to resort to average-case analyses by stipulating realistic distributions of input data. Another way is to incorporate a priori information on the potential domain of the input data, for instance, assuming the triangle inequality for input matrices is in some cases instrumental for establishing approximation algorithms with fixed performance guarantee. Now, a parametrized form of the triangle inequality has a considerably larger range of applicability and allows the prediction of the heuristics performance, where otherwise no bound could be provided. For example, it is interesting to observe that two well-known approximation algorithms for the Traveling Salesman Problem (TSP), assuming the triangle inequality, behave differently when one relaxes the imposed triangle inequality. The double-spanning-tree heuristic can be adjusted (by suitably extracting a Hamilton circuit from a Eulerian walk) to yield an approximation algorithm with performance guarantee increasing quadratically with the parameter governing the relaxed triangle inequality. The Christofides algorithm cannot be modified in this way and hence does not tolerate a relaxation of the standard triangle inequality without loosing the bound on its relative performance.

**Key words.** approximation algorithm, performance guarantee, parametrized triangle inequality, Traveling Salesman Problem, minimum Steiner tree, anticlustering

**AMS subject classifications.** 90C27, 90C35, 68R99

**Introduction.** Hard combinatorial optimization problems are often approached in practice by heuristic procedures that produce near-optimal solutions, which for particular data, however, may be far away from optima. It is then interesting from a theoretical point of view to predict the worst-case behavior of such algorithms. A heuristic qualifies as an approximation algorithm if the returned solutions are within a fixed factor $r$ (performance guarantee) of the optimal value. Even this relaxation of the problem may be too demanding, viz., leaving it still NP-hard. One would then resort to a (possibly more consolatory) average-case analysis or restrict the range of feasible input data. The Traveling Salesman Problem (TSP) is quite typical in this respect: there is no hope (unless $P = NP$) for a polynomial approximation algorithm with performance guarantee, but if the input distance matrix $C = (c_{i,j})$ obeys the triangle inequality, then several approximation algorithms are available. It is conceivable that slight violations of the triangle inequality should not be too deleterious with respect to performance guarantees. The deviation from the triangle inequality is captured by a parameter $\tau$ in the following relaxation of the triangle inequality:

$$c_{i,j} \le \tau(c_{i,k} + c_{k,j}) \tag{1}$$

for all choices of three distinct points $i, j, k$. This parametrized form, suggested by Bandelt, Crama, and Spieksma (1991), allows us to tailor the worst-case analysis

* Received by the editors November 16, 1992; accepted for publication (in revised form) January 21, 1994.

† Mathematisches Seminar der Universität Hamburg, Bundesstr. 55, D-20146 Hamburg 13, Germany.

more faithfully to the expected instances of a particular real-world problem, as was demonstrated for several heuristics solving multidimensional assignment problems, where the performance guarantee was even bounded from above by a factor $r$ linearly dependent on the parameter $\tau$.

A parametrized triangle inequality naturally comes up when the input data are from a fixed range of values. Assume that all cost matrices $C = (c_{i,j})$ under consideration are bounded by real numbers $L$ and $U$:

$$L \leq c_{i,j} \leq U \quad \text{for all } i, j.$$

If $L > 0$, then $C$ certainly satisfies the inequality (1) with parameter $\tau = U/2L$. In case more a priori information is available, smaller choices of the parameter $\tau$ may be possible. For instance, if the median cost of the three edges in each triangle is bounded below by $M$, that is,

$$M \leq \max\{\min(c_{i,j}, c_{i,k}), \min(c_{i,j}, c_{j,k}), \min(c_{i,k}, c_{j,k})\},$$

then $\tau = U/(L + M)$ is a feasible choice. In this case one could even admit negative costs as long as $-L < M$.

In this paper we will study the parametrized triangle inequality in more detail. In the presence of the standard triangle inequality (that is, for $\tau = 1$) it is straightforward to estimate the distance between the end points of a path with more than one interior point: take the sum of distances along the path. When $\tau > 1$, however, a corresponding factor necessarily grows with the number of interior points. In particular, we will determine the exact values of the smallest possible factors $\tau_n$, given $\tau$, in the inequalities

$$(1') \qquad c_{i,j} \leq \tau_n(c_{i,k_1} + c_{k_1,k_2} + \cdots + c_{k_{n-1},k_n} + c_{k_n,j}).$$

Asymptotically, $\tau_n$ equals $\tau$ to the power $\log_2(n + 1)$. (Any application of this "iterated parametrized triangle inequality" for large $n$ will thus lead only to poor estimates of the performance of the approximation algorithm under study.)

Now, what can be said about the performance of some TSP heuristics when the relaxed triangle inequality is imposed? Two pertinent algorithms depart from a minimum-cost-spanning tree by expanding it to a Eulerian graph, either by simply doubling each edge or by adding a minimum-cost matching of the vertices of odd degree. Extracting a Hamilton circuit from an arbitrary Eulerian walk then results in an algorithm with performance guarantee 2 or 3/2, respectively, provided the distances satisfy the triangle inequality. It turns out that the two algorithms perform differently when applied to distance matrices $(c_{i,j})$ fulfilling the relaxed form of the triangle inequality with parameter $\tau > 1$. The simple double-spanning-tree heuristic can be modified (by carefully selecting the Hamilton circuit) in such a way that a performance guarantee of factor $3\tau^2/2 + \tau/2$ can be achieved. The other algorithm (due to Christofides) does not profit from such a selection strategy of tours: for $\tau > 1$, the worst-case ratio of heuristic tour length and optimal tour length goes to infinity when the number $n$ of cities (points) increases.

So far we have regarded the parametrized triangle inequality as a relaxation of the standard one, i.e., we have assumed $\tau > 1$. But when distances satisfy the standard triangle inequality strictly, we can more precisely describe this by an improved triangle inequality with parameter $\tau < 1$. Evidently, $\tau$ cannot be smaller than 1/2. This scenario applies to the minimum-spanning-tree heuristic for the Steiner tree problem: when $\tau$ approaches $1/2$, the performance guarantee factor 2 decreases and eventually reaches 1. As in the multidimensional assignment problem we treat the anticlustering problem of Feo and Khellaf (1990) for every choice of the parameter $\tau$ with $1/2 \leq \tau < \infty$. Our terminology is fairly standard; cf. Papadimitriou and Steiglitz (1982). Vertex set and edge set of a (finite, simple, undirected) graph $G$ are denoted by $V(G)$ and $E(G)$, respectively. For $k \geq 1$, the *kth power* $G^k$ of a graph $G$ is the graph with the same vertex set as $G$ and where two distinct vertices $a$ and $b$ of $G$ form an edge if and only if there exists an $a, b$-path in $G$ consisting of at most $k$ edges. A *trivial tree* is a tree consisting of just one vertex.

Let $X$ be a set and let $c$ be a mapping that assigns to every two-element subset $\{u, v\}$ of $X$ a nonnegative real number $c(u, v)$. Then $c$ is called a *cost function* on $X$, and $c(u, v)$ is the cost of the edge $\{u, v\}$. Whenever appropriate, we will also write $c_{u,v}$ instead of $c(u, v)$. In some instances, $c$ is referred to as a *distance function.* If $E$ is a finite subset of $\binom{X}{2}$, then we write $c(E)$ for the sum of all $c(u, v)$ for which $\{u, v\} \in E$. We call $c(E)$ the *cost of* $E$. If $E$ is the edge set of a graph $H$, then we also write $c(H)$ instead of $c(E)$; and in case $H$ is complete, that is, $E(H) = \binom{A}{2}$ for $A \subseteq X$, the cost $c(H)$ is written as $c(A)$. We say that a cost function $c$ on $X$ satisfies the *$\tau$-inequality* (for some $\tau$ with $1/2 \leq \tau < \infty$) if (1) holds for all choices of three distinct points $i, j, k$ in $X$.

**1. Iterated parametrized triangle inequalities.** Our goal is to determine the smallest possible factors $\tau_n$ in the inequalities (1′). To this end some technical prerequisites are necessary. We assume $\tau \geq 1$ (unless stated otherwise). For each positive integer $n$, let us define numbers $a_n$ and $b_n$ as follows: let $n = 2^q + r$ with integers $q, r$ such that $q \geq 0$ and $0 \geq r < 2^q$; then (for given $\tau \geq 1$)

$$a_n := r\tau^{q+1} + (\lfloor n/2 \rfloor - r)\tau^q,$$
$$b_n := r\tau^{q+1} + (\lceil n/2 \rceil - r)\tau^q.$$

Note that $a_n$ and $b_n$ equal $(n/2)\tau^{\log_2 n}$ asymptotically: indeed, the numbers $a_n$ and $b_n$ are relatively minor deviations from their arithmetic mean

$$r\tau^{q+1} + \left(\frac{n}{2} - r\right)\tau^q,$$

which constitutes the linear interpolation of the convex function

$$h(x) = \frac{x}{2}\tau^{\log_2 x} = \frac{1}{2}x^{1+\log_2 \tau}$$

along the points $x = 2^q$ for $q = 0, 1, \ldots$.

In the next lemma we collect some properties of the numbers $a_n$ and $b_n$ that will be useful in the following sections. The proof of this lemma is left to the reader.

LEMMA 1. *The numbers $a_n, b_n$ have the following properties* (2.1)–(2.4).

(2.1) *Let* $q = \lfloor \log_2 n \rfloor$. *Then*

$$b_n - a_n = \begin{cases} \tau^q & \text{if } n \text{ is odd,} \\ 0 & \text{otherwise,} \end{cases}$$

$$a_{n+1} - a_n = \begin{cases} \tau^{q+1} & \text{if } n \text{ is odd,} \\ \tau^{q+1} - \tau^q & \text{otherwise,} \end{cases}$$

$$b_{n+1} - b_n = \begin{cases} \tau^{q+1} - \tau^q & \text{if } n \text{ is odd,} \\ \tau^{q+1} & \text{otherwise.} \end{cases}$$

(2.2) $a_n \le b_n \le a_{n+1}$.

(2.3) *Let $n \ge m$. If $n - m$ is even, then*

$$\begin{aligned} a_{n+1} - a_n &\ge a_{m+1} - a_m, \\ b_{n+1} - b_n &\ge b_{m+1} - b_m, \\ b_n - a_n &\ge b_m - a_m; \end{aligned}$$

*if $n - m$ is odd, then*

$$\begin{aligned} a_{n+1} - a_n &\ge b_{m+1} - b_m, \\ b_{n+1} - b_n &\ge a_{m+1} - a_m. \end{aligned}$$

(2.4) *If $n \ge 3$ is odd, then*

$$a_n = \tau(a_{\lfloor n/2 \rfloor} + a_{\lceil n/2 \rceil}) \quad \textit{and} \quad b_n = \tau(b_{\lfloor n/2 \rfloor} + b_{\lceil n/2 \rceil});$$

*if $n$ is even, then*

$$a_n = b_n = \tau(a_{n/2} + b_{n/2}).$$

For $X = \{0, 1, \ldots, n\}$ with $n \ge 1$, we define a cost function $c$ on $X$ as follows: for $i, k \in X$ with $i < k$ let

$$c_{i,k} = c_{k,i} = \begin{cases} a_{k-i} & \text{for } i \text{ even and } k \text{ odd}, \\ b_{k-i} & \text{for } i \text{ odd and } k \text{ even}, \\ a_{k-i} (= b_{k-i}) & \text{otherwise}. \end{cases} \tag{3}$$

LEMMA 2. *The cost function $c$ defined on $X = \{0, 1, \ldots, n\}$ by* (3) *satisfies the $\tau$-inequality.*

*Proof.* Let $i, j, k \in X, i < j < k$. Then $\max\{c_{i,j}, c_{j,k}, c_{i,k}\} = c_{i,k}$ by (3) and (2.2). Hence, for the proof of the $\tau$-inequality, it suffices to show

$$c_{i,k} \le \tau(c_{i,j} + c_{j,k}), \qquad 0 \le i < j < k \le n. \tag{4}$$

Note that, by the definition of $c$, it is sufficient to prove (4) for $i \in \{0, 1\}$.

*Case* 1: *$i = 0$ and $k$ is odd.* Then (by (3)) the claimed inequality (4) is equivalent to

$$a_k \le \tau(a_j + a_{k-j}), \qquad j = 1, \ldots, k-1. \tag{4.1}$$

Clearly it suffices to show (4.1) for $1 \le j \le (k-1)/2$, which we assume henceforth. Then $k - j - 1 \ge j$. Note further that $k - j - 1 - j$ is an even number and, therefore, we can apply (2.3) to obtain $a_{k-j} - a_{k-j-1} \ge a_{j+1} - a_j$. Hence

$$a_j + a_{k-j} \ge a_{j+1} + a_{k-(j+1)}, \qquad j = 1, \ldots, (k-1)/2.$$

Moreover, we already know from (2.4) that statement (4.1) is true for $j = (k-1)/2$ and thus (by the inequalities we have just derived) it is true for all $j = 1, \ldots, (k-1)/2$. This settles Case 1.

*Case* 2: *$i = 1$ and $k$ is even.* In this case, (4) becomes

$$b_{k-1} \le \tau(b_{j-1} + b_{k-j}), \qquad j = 2, \ldots, k-1.$$

With $h = k - 1$ this is equivalent to

$$(4.2) \qquad b_h \leq \tau(b_j + b_{h-j}), \qquad j = 1, \ldots, h-1.$$

Since $h$ is odd, (4.2) can be verified analogously to the proof of (4.1) by making use of (2.3) and (2.4).

The remaining cases ($i = 0$ with $k$ even, and $i = 1$ with $k$ odd) can be settled by similar arguments. □

We now prove the main result of this section, which, in other words, states that the smallest possible factor $\tau_n$ in the inequality (1′) equals

$$\frac{a_{n+1}}{\lfloor (n+1)/2 \rfloor}.$$

THEOREM 1. *Let $X = \{0, 1, \ldots, n\}$ with $n = 2^q + r (0 \leq r < 2^q, q \geq 1)$ and let $c$ be a cost function on $X$ satisfying the $\tau$-inequality for $\tau \geq 1$. Then*

$$(5) \qquad c(0,n) \leq \frac{r\tau^{q+1} + (\lfloor n/2 \rfloor - r)\tau^q}{\lfloor n/2 \rfloor} \sum_{j=0}^{n-1} c(j, j+1),$$

*which is best possible, as for each $\tau \geq 1$ and $n \geq 2$ there exists an example yielding equality in* (5).

*Proof.* We first consider the case in which $n$ is a power of 2. In this case $r = 0$ and thus the claimed inequality (5) becomes

$$(5') \qquad c(0, 2^q) \leq \tau^q \sum_{j=0}^{2^q-1} c(j, j+1),$$

which can easily be verified by induction on $q$: indeed, for $q = 1$, (5′) is just the $\tau$-inequality, and for $q \geq 2$, by making use of the induction hypothesis, one obtains the following:

$$\begin{aligned} c(0, 2^q) &\leq \tau(c(0, 2^{q-1}) + c(2^{q-1}, 2^q)) \\ &\leq \tau \left( \tau^{q-1} \sum_{j=0}^{2^{q-1}-1} c(j, j+1) + \tau^{q-1} \sum_{j=2^{q-1}}^{2^q-1} c(j, j+1) \right) \\ &= \tau^q \sum_{j=0}^{2^q-1} c(j, j+1). \end{aligned}$$

Now let us assume that $n$ is not a power of 2. Then $1 \leq r \leq \lfloor n/2 \rfloor$. Let

$$T_n = (t_{i,j}), \qquad i, j = 0, \ldots, \lfloor n/2 \rfloor - 1$$

be an $\lfloor n/2 \rfloor \times \lfloor n/2 \rfloor$-matrix with entries 0 or 1 such that each row and each column contains exactly $r$ entries equal to 1; a matrix with these properties is easily seen to exist. For each $i \in \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$, denote by $j_{i,1}, \ldots, j_{i,r}$ the column indices $j$ for which $t_{i,j} = 1$, where we assume

$$j_{i,1} < j_{i,2} < \cdots < j_{i,r}, \qquad i = 0, \ldots, \lfloor n/2 \rfloor - 1.$$

Furthermore, for each $i \in \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$, let $J_i$ be the set of integers that are of the form $2j_{i,k} + 1$ for $k \in \{1, \ldots, r\}$. Then, clearly, $J_i \subseteq \{1, \ldots, n-1\}$ and $|J_i| = r$ and, consequently, $|\{0, 1, \ldots, n\} \backslash J_i| = 2^q + 1$. For $i = 0, \ldots, \lfloor n/2 \rfloor - 1$, denote the members of $\{0, 1, \ldots, n\} \backslash J_i$ by

$$m_{i,0} < m_{i,1} < \cdots < m_{i,2^q}.$$

Then, evidently, $m_{i,0} = 0$ and $m_{i,2^q} = n$ ($i = 0, \ldots, \lfloor n/2 \rfloor - 1$). Hence we conclude from (5′) that

$$(6) \qquad c(0,n) \le \tau^q \sum_{k=0}^{2^q-1} c(m_{i,k}, m_{i,k+1}), \qquad i = 0, \ldots, \lfloor n/2 \rfloor - 1.$$

For all $i \in \{0, \ldots, \lfloor n/2 \rfloor - 1\}$ and $k \in \{1, \ldots, r\}$, the $\tau$-inequality yields

$$(7) \qquad c(2j_{i,k}, 2j_{i,k} + 2) \le \tau(c(2j_{i,k}, 2j_{i,k} + 1) + c(2j_{i,k} + 1, 2j_{i,k} + 2)).$$

Let us define an $\lfloor n/2 \rfloor \times n$-matrix $S_n = (s_{i,j})$ as follows: for all $i \in \{0, \ldots, \lfloor n/2 \rfloor - 1\}$ and $j \in \{0, \ldots, n-1\}$ let

$$\begin{aligned} s_{i,j} &:= 1 \quad \text{if } j = 2j_{i,k} \text{ or } j = 2j_{i,k} + 1 \text{ for some } k \in \{1, \ldots, r\}, \text{ and} \\ s_{i,j} &:= 0 \quad \text{otherwise.} \end{aligned}$$

In other words, $S_n$ results from $T_n$ by (i) doubling each column (i.e., inserting to the right of each column an identical copy of it), and (ii) adding an all-zero column to the right of all columns if $n$ is odd. In particular,

(8) each column of $S_n$ contains at most $r$ entries equal to 1.

Now, from (6) and (7), together with the definition of $S_n = (s_{i,j})$, one immediately obtains

$$(9) \qquad c(0,n) \le \sum_{j=0}^{n-1} \tau^{q+s_{i,j}} c(j, j+1), \qquad i = 0, \ldots, \lfloor n/2 \rfloor - 1,$$

since for each $i$ there exist exactly $2^q - r$ pairs $m_{i,k}, m_{i,k+1}$ of consecutive numbers, whereas $r$ pairs $m_{i,k}, m_{i,k+1}$ are of the form $2j_{i,k}, 2j_{i,k} + 2$ (so that (7) applies). Summing up these inequalities and dividing the result by $\lfloor n/2 \rfloor$ yields

$$\begin{aligned} c(0,n) &\le \frac{1}{\lfloor n/2 \rfloor} \sum_{i=0}^{\lfloor n/2 \rfloor - 1} \sum_{j=0}^{n-1} \tau^{q+s_{i,j}} c(j, j+1) \\ &= \frac{1}{\lfloor n/2 \rfloor} \sum_{j=0}^{n-1} \left( \sum_{i=0}^{\lfloor n/2 \rfloor - 1} \tau^{q+s_{i,j}} \right) c(j, j+1). \end{aligned}$$

In addition, (8) yields for all $j \in \{0, \ldots, n-1\}$

$$\sum_{i=0}^{\lfloor n/2 \rfloor - 1} \tau^{q+s_{i,j}} \le r\tau^{q+1} + (\lfloor n/2 \rfloor - r)\tau^q.$$

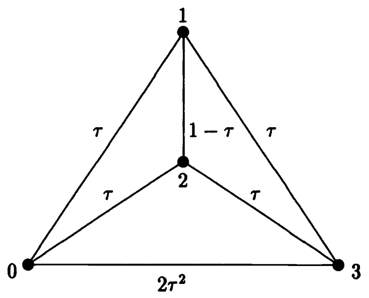


FIG. 1. *An example showing that the bound of* (13) *is best possible.*

This proves (5).

Now, let $X = \{0, 1, \dots, n\}$ and let $c$ be the cost function defined on $X$ by (3). Then by Lemma 2 the $\tau$-inequality is satisfied, and we have $c(0, n) = a_n$ and

$$\sum_{j=0}^{n-1} c(j, j+1) = \lfloor n/2 \rfloor.$$

Hence, for the so-defined example $(X, c)$, (5) holds with equality. □

To provide sharp bounds for the iterated $\tau$-inequality in case $\tau < 1$ does not seem easy. Here is the case of $n = 3$ edges: From $c_{i,j} \leq \tau c_{i,k} + \tau c_{k,j}$ and $\tau c_{k,j} \leq \tau^2 c_{i,j} + \tau^2 c_{i,k}$ it follows that $c_{i,j} \leq \tau c_{i,k} + \tau^2 c_{i,j} + \tau^2 c_{i,k}$ whence $(1 - \tau^2) c_{i,j} \leq \tau(1+\tau) c_{i,k}$, that is,

$$c_{i,j} \leq \frac{\tau}{1-\tau} c_{i,k} \quad \text{for any triple of pairwise distinct vertices } i, j, k. \tag{10}$$

Iterating the $\tau$-inequality for $c_{0,3}$ in two different ways gives

$$c_{0,3} \leq \tau^2 c_{0,1} + \tau^2 c_{1,2} + \tau c_{2,3},$$
$$c_{0,3} \leq \tau c_{0,1} + \tau^2 c_{1,2} + \tau^2 c_{2,3}.$$

Hence

$$2c_{0,3} \leq (\tau^2 + \tau) c_{0,1} + 2\tau^2 c_{1,2} + (\tau^2 + \tau) c_{2,3}. \tag{11}$$

Let $\lambda \geq 0$ be a real number (to be determined later); then by (10) we have $\lambda c_{0,1} \leq \lambda(\tau/(1-\tau)) c_{1,2}$ as well as $\lambda c_{2,3} \leq \lambda(\tau/(1-\tau)) c_{1,2}$, and thus we obtain from (11)

$$2c_{0,3} \leq (\tau^2 + \tau - \lambda) c_{0,1} + \left(2\tau^2 + 2\lambda \frac{\tau}{1-\tau}\right) c_{1,2} + (\tau^2 + \tau - \lambda) c_{2,3}. \tag{12}$$

Now, in order to have all three coefficients on the right-hand side of (12) equal, we choose

$$\lambda = \frac{\tau(1-\tau)^2}{1+\tau}.$$

With this choice of $\lambda$ the inequality (12) yields

$$c_{0,3} \leq \frac{2\tau^2}{1+\tau} (c_{0,1} + c_{1,2} + c_{2,3}). \tag{13}$$

This bound is best possible, as the (legitimate) example of Fig. 1 shows.

*Problem.* Provide an analogue to Theorem 1 for $\tau < 1$.

**2. The TSP.** By $\Delta_\tau$TSP we mean the Traveling Salesman Problem restricted to distance matrices satisfying the $\tau$-inequality for some fixed $\tau \geq 1/2$. Given an input for $\Delta_\tau$TSP, that is, a set $X$ of $n$ "cities" and a distance function $c$ on $X$ satisfying the $\tau$-inequality, we denote by $c_{\min}$ the cost of an optimal tour on $X$; moreover, given an approximation algorithm for $\Delta_\tau$TSP, we write $c_{\text{approx}}$ for the cost of an approximate solution. In the following, we restrict ourselves to the case $\tau \geq 1$. If $\tau = 1$, then we write $\Delta$TSP rather than $\Delta_1$TSP. We discuss two well-known heuristic procedures for the TSP, namely, the double-tree algorithm and Christofides' algorithm; cf. Papadimitriou and Steiglitz (1982).

Double-Tree algorithm
**Step 1.** Find a minimum spanning tree $T$ for $(X, c)$.
**Step 2.** Create the multigraph $T_d$ by doubling each edge of $T$.
**Step 3.** Find a Eulerian walk $E$ of $T_d$, extract a Hamilton circuit $H$ from $E$, and return the corresponding tour as a heuristic solution.

Christofides' algorithm
**Step 1.** Find a minimum spanning tree $T$ for $(X, c)$.
**Step 2.** Create the multigraph $T_m$ by adding to $T$ a minimum-cost matching of the vertices of $T$ having odd degree.
**Step 3.** Find a Eulerian walk $E$ of $T_m$, extract a Hamilton circuit $H$ from $E$, and return the corresponding tour as a heuristic solution.

In Step 3 of these algorithms, extraction of a Hamilton circuit $H$ from $E$ means that, for each vertex $v$ occurring more than once in $E, m$ times say, one arbitrarily deletes $m-1$ of $v$'s occurrences. If $H$ can be obtained from $E$ in this way, we also say that $H$ is an *embedded tour* of $E$.

It is well known that the double-tree algorithm and Christofides' algorithm are approximation algorithms for $\Delta$TSP with performance guarantees 2 and 3/2, respectively; cf. Papadimitriou and Steiglitz (1982). We now consider the $\Delta_\tau$TSP for some fixed $\tau > 1$ and study the worst-case behaviour of the above procedures. The following example is instructive. For $X = \{0, 1, \ldots, n-1\}$ with $n - 1 = 2^q (q \geq 1)$, define the distances $c_{i,k}$ as in (3). Recall that, by Lemma 2, the so-defined distance function $c$ on $X$ satisfies the $\tau$-inequality. Then

$$c_{\min} \leq (n-2)\tau + 1,$$

since a tour with cost $(n-2)\tau + 1$ can easily be specified: just take the tour $(0, 2, 4, \ldots, n-1, n-2, n-4, \ldots, 3, 1, 0)$. Note also that the path $T = (0, 1, \ldots, n-1)$ is the (uniquely determined) minimum-spanning tree of $(X, c)$. Hence Christofides' algorithm returns the tour $H = (0, 1, \ldots, n-1, 0)$, and one easily sees that the double-tree algorithm may also return this tour, since $H$ is an embedded tour of the Eulerian walk $(0, 1, \ldots, n-1, \ldots, 1, 0)$. Note that

$$c(H) = \frac{n-1}{2} + \frac{n-1}{2}\tau^q = \frac{n-1}{2}\left(1 + \tau^{\log_2(n-1)}\right)$$

and thus

$$\frac{c(H)}{c_{\min}} \geq \frac{(n-1)(1 + \tau^{\log_2(n-1)})}{2(n-2)\tau + 2},$$

which (as $\tau > 1$) goes to infinity as $n$ goes to infinity. In other words, the worst-case ratio of heuristic tour length and minimal tour length goes to infinity as the number $n$ of "cities" increases. Actually, in the case of Christofides' algorithm, things are even worse: note that, in the above example, the graph $T_m$ equals the circuit $(0, 1, \ldots, n-1, 0)$ and, therefore, no choice other than $E = H = T_m$ was possible. Hence, in the case of Christofides' algorithm, it is impossible to overcome the poor worst-case behaviour by specifying additional rules for the selection of $E$ and $H$. This negative result is in contrast to what we have for the double-tree algorithm: we will see that, by carefully selecting $H$ from the variety of Hamilton circuits embedded in some Eulerian walk of $T_d$, the double-tree algorithm can be refined to become an approximation algorithm with performance guarantee $3\tau^2/2 + \tau/2$.

Our refinement of the double-tree algorithm is based on a result of Sekanina (1960) stating that the cube $T^3$ of a tree $T$ with at least three vertices has a Hamilton circuit; see also Walther and Voss (1974). For our purpose, we need the following Theorem A, which is a slightly modified version of Sekanina's theorem. For a tree $T$ with $n \geq 3$ vertices, let $H = (x_0, x_1, \ldots, x_n = x_0)$ be a Hamilton circuit of $T^3$ and denote by $P_i$ the unique $x_{i-1}, x_i$-path of $T$ $(i = 1, \ldots, n)$. Then, obviously, each $P_i$ has at most three edges. If $P_i$ has exactly three edges, $P_i = (x_{i-1}, u, v, x_i)$ say, then the edge $e = \{u, v\}$ is called the *middle edge* of $P_i$, and $H$ is called *admissible* if

(14.1) each edge of $T$ is contained in exactly two of the paths $P_i$ $(i = 1, \ldots, n)$,

and

(14.2) each edge of $T$ is the middle edge of at most one $P_i$ $(i = 1, \ldots, n)$.

THEOREM A. *Let $T$ be a tree with $n \geq 3$ vertices and let $\{a, b\}$ be an edge of $T$. Then $T^3$ has a Hamilton $a, b$-path $P$ such that the corresponding Hamilton circuit $H = P + \{a, b\}$ is admissible.*

*Proof.* The proof of Theorem A runs completely along the lines of the proof of Sekanina's theorem; see, e.g., Sekanina (1960) or Walther and Voss (1974). For $n = 3$, Theorem A clearly holds. Let $n \geq 4$ and assume that the theorem is true for trees with fewer than $n$ vertices. Let $T - \{a, b\}$ be the graph that results when we delete the edge $\{a, b\}$ from $T$; let $T_a$ and $T_b$ be the components of $T - \{a, b\}$ containing $a$ and $b$, respectively. If $|V(T_a)| \geq 3$, pick a neighbour $a'$ of $a$ with $a' \in T_a$, and (inductively) find a Hamilton $a, a'$-path $P_a$ in $T_a^3$ with the property described in Theorem A. If $|V(T_a)| = 2$, let $P_a$ be the path of length one formed by $a$ and its unique neighbour $a' \in V(T_a)$. If $|V(T_a)| = 1$, let $P_a$ be the trivial path consisting of $a$ and let $a' = a$. Analogously, $b'$ and $P_b$ are defined. Then the desired Hamilton $a, b$-path $P$ is obtained by linking $P_a$ and $P_b$ by the edge $\{a', b'\}$. $\square$

The inductive argument of the preceding proof also shows that $P$ (and consequently $H$ as well) can be obtained in $O(n)$ time, since splitting $T$ into $T_a$ and $T_b$, picking $a'$ and $b'$, and forming $P$ from $P_a$ and $P_b$ are operations that can be done in constant time.

Now consider the following heuristic procedure for the TSP which we call the *$T^3$-algorithm.*

$T^3$-ALGORITHM

**Step 1.** Find a minimum spanning tree $T$ for $(X, c)$.

**Step 2.** Find an admissible Hamilton circuit $H$ of $T^3$ and return the corresponding tour as a heuristic solution.

Note that by (14.1), if we traverse the above-defined paths $P_1, P_2, \ldots, P_n$ in this order, then we obtain a Eulerian walk $E$ of $T_d$ having $H$ as an embedded tour and, therefore, the $T^3$-algorithm can be viewed as a refinement of the double-tree algorithm.

THEOREM 2. *For $\tau \geq 1$, the $T^3$-algorithm is an approximation algorithm for $\Delta_\tau$TSP with performance guarantee*

$$c_{\text{approx}} \leq \left(\frac{3}{2}\tau^2 + \frac{1}{2}\tau\right) c_{\min} \tag{15}$$

*Proof.* For $\tau \geq 1$, let $(X, c)$ be an input for $\Delta_\tau$TSP with $|X| = n$. For a spanning tree $T$ of $(X, c)$, let $H = (x_0, x_1, \dots, x_n = x_0)$ be an admissible Hamilton circuit of $T^3$ returned by the $T^3$-algorithm. As above let $P_i$ be the unique $x_{i-1}, x_i$-path of $T\,(i = 1, \dots, n)$. Further, put $p_i := |E(P_i)|$ and recall that $1 \leq p_i \leq 3\,(i = 1, \dots, n)$.

If $p_i = 3$, let $P_i = (v_{i,0}, v_{i,1}, v_{i,2}, v_{i,3})$ where $x_{i-1} = v_{i,0}, x_i = v_{i,3}$, and put $e_{i,j} = \{v_{i,j-1}, v_{i,j}\}$ for $j = 1, 2, 3$. Iterating the $\tau$-inequality in two different ways one obtains

$$c(x_{i-1}, x_i) \leq \tau c(e_{i,1}) + \tau^2 c(e_{i,2}) + \tau^2 c(e_{i,3}),$$
$$c(x_{i-1}, x_i) \leq \tau^2 c(e_{i,1}) + \tau^2 c(e_{i,2}) + \tau c(e_{i,3}).$$

Hence

$$c(x_{i-1}, x_i) \leq \frac{\tau^2 + \tau}{2} c(e_{i,1}) + \tau^2 c(e_{i,2}) + \frac{\tau^2 + \tau}{2} c(e_{i,3}) \quad \text{if } p_i = 3. \tag{16.1}$$

If $p_i = 2$ then (choosing the notations $e_{i,1}, e_{i,2}$ similar to the case $p_i = 3$) by making use of the $\tau$-inequality one obtains

$$c(x_{i-1}, x_i) \leq \tau c(e_{i,1}) + \tau c(e_{i,2}).$$

Hence (because $(\tau^2 + \tau)/2 \geq \tau$) we have

$$c(x_{i-1}, x_i) \leq \frac{\tau^2 + \tau}{2} c(e_{i,1}) + \frac{\tau^2 + \tau}{2} c(e_{i,2}) \quad \text{if } p_i = 2. \tag{16.2}$$

Finally, if $p_i = 1$, let $e_{i,1} = (x_{i-1}, x_i)$. Hence (because $(\tau^2 + \tau)/2 \geq 1$) we have

$$c(x_{i-1}, x_i) \leq \frac{\tau^2 + \tau}{2} c(e_{i,1}) \quad \text{if } p_i = 1. \tag{16.3}$$

For $i = 1, \dots, n$ and $j = 1, \dots, p_i$ let $\lambda_{i,j} = \tau^2$ if $p_i = 3$ and $j = 2$, and $\lambda_{i,j} = (\tau^2 + \tau)/2$, otherwise. Then we obtain from (16.1)–(16.3)

$$c_{\text{approx}} = \sum_{i=1}^{n} c(x_{i-1}, x_i) \leq \sum_{i=1}^{n} \sum_{j=1}^{p_i} \lambda_{i,j} c(e_{i,j}). \tag{17}$$

By (14.1) and (14.2) each $e \in E(T)$ occurs exactly twice in the right-hand sum of (17) and, for at most one of these occurrences, the corresponding coefficient $\lambda_{i,j}$ equals $\tau^2$. Hence the right-hand side of (17) is at most

$$\sum_{e \in E(T)} \left(\tau^2 + \frac{\tau^2 + \tau}{2}\right) c(e) = \left(\frac{3}{2}\tau^2 + \frac{1}{2}\tau\right) c(T).$$

If in an optimal tour we delete an edge of maximum cost, then we obtain a spanning tree of $(X, c)$ which has cost at most $(1 - 1/n)c_{\min}$ and thus $c(T) \leq (1 - 1/n)c_{\min}$. Summarizing, we have

$$c_{\text{approx}} \leq \left(\frac{3}{2}\tau^2 + \frac{1}{2}\tau\right)\left(1 - \frac{1}{n}\right) c_{\min},$$

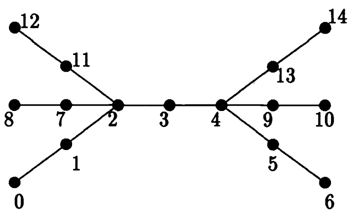


FIG. 2. *The tree $T$ for $k = 3$. The Hamilton circuit $H$ is indicated by the labels of $t$ vertices.*

which implies the assertion of our theorem. □

We do not believe that the performance guarantee obtained in (15) is really the best possible. We conjecture that a more careful analysis of the $T^3$-algorithm (possibly together with a different selection of the Hamilton circuit $H$ of $T^3$) would yield

$$c_{\text{approx}} \leq (\tau^2 + \tau)c_{\text{min}}. \tag{15'}$$

This would asymptotically be best possible as the following example shows. Let $\tau > 1$. For $k \geq 1$, define a tree $T$ on $4k + 3$ vertices as follows: Denote the vertices of $T$ by $x_1, \ldots, x_k, x'_1, \ldots, x'_k, y_1, \ldots, y_k, y'_1, \ldots, y'_k, z, z', w$, and let the edges of $T$ be $\{w, z\}, \{w, z'\}\{z, y_i\}, \{z', y'_i\}\{y_i, x_i\}, \{y'_i, x'_i\}(i = 1, \ldots, k)$. Let $G$ be the complete graph with vertex set $X = V(T)$, and let $H$ be the Hamilton circuit of $G$ defined by

$$H = (x_1, y_1, z, w, z', y'_1, x'_1, y_2, x_2, y'_2, x'_2, \ldots, y_k, x_k, y'_k, x'_k, x_1).$$

In Fig. 2, $T$ and $H$ are displayed for $k = 3$.

It is easy to check that the graph $T \cup H$ has the following property:

(18) $T \cup H$ contains no cycles with fewer than six edges.

We define a cost function $c$ on $\binom{X}{2}$ as follows: if $e \in E(T \cup H)$, then let

$$c(e) = \begin{cases} 0 & \text{if } e \text{ is a pendant edge of } T, \\ 1 & \text{otherwise.} \end{cases}$$

Now, let $e \notin E(T \cup H)$, say $e = \{v, v'\}$. If there exists a vertex $u$ such that $\{v, u\}$, $\{v', u\} \in E(T \cup H)$, then (by (18)) there is just one vertex $u$ with this property and so we can define

$$c(v, v') = \begin{cases} \tau & \text{if } c(v, u) + c(v', u) = 1, \\ 2\tau & \text{otherwise.} \end{cases}$$

Finally, for the remaining edges $e = \{v, v'\}$, we define

$$c(v, v') = \begin{cases} \tau^2 & \text{if there exists a vertex } u \in X \\ & \text{such that one of the edges } \{v, u\}, \{v', u\} \\ & \text{has cost } 0 \text{ and the other cost } \tau, \\ \tau^2 + \tau & \text{otherwise.} \end{cases}$$

We claim that the so-defined example $(X, c)$ satisfies the $\tau$-inequality. In order to prove this, we first note that (trivially)

(19) no triangle of $(X, c)$ contains more than one edge $e$ with $c(e) = 0$.

We say that a triangle $\Delta$ of $(X, c)$ is of *type* $\langle \alpha, \beta, \gamma \rangle$ if $\alpha, \beta$, and $\gamma$ are the costs of the edges of $\Delta$, respectively. Now, let $\Delta$ be a (hypothetical) triangle of $(X, c)$ that violates the $\tau$-inequality. Then one easily concludes from the definition of $c$ (together with (19)) that $\Delta$ must be of one of the types $\langle 0, \tau, 2\tau\rangle, \langle 0, \tau^2, 2\tau\rangle, \langle 0, \tau^2, \tau^2+\tau\rangle$. On the other hand, one finds that none of these types can actually occur in $(X, c)$: Let $a, b, d \in X$ and assume that $c(a, b) = 0$. If $c(b, d) = \tau$ and $c(a, d) = 2\tau$, then (by the definition of $c$ together with (19)) one could find vertices $e, f$ such that $c(b, e) = 1, c(e, d) = 0, c(d, f) = c(f, a) = 1$, contradicting (18). If $c(b, d) = \tau^2$ and $c(a, d) \in \{2\tau, \tau^2+\tau\}$, then we could find a vertex $e$ such that $c(b, e) = \tau$ and $c(e, d) = 0$. Furthermore, $c(b, e) = \tau$ implies that there exists a vertex $f$ such that $c(b, f) + c(f, e) = 1$, which (by (19)) is impossible unless $f = a$. But $f = a$ would imply $c(a, d) = \tau$, which is a contradiction. Hence we have shown that $(X, c)$ satisfies the $\tau$-inequality.

Since $H$ is certainly an optimal tour of $(X, c)$, we have

$$c_{\min} = c(H) = 2k + 3. \tag{20}$$

We claim

$$c(H') \geq 2(k-2)(\tau^2+\tau) \text{ for each Hamilton circuit } H' \text{ of } T^3. \tag{21}$$

To verify this, let $E_i = \{\{x_i, v\} : v = w \text{ or } v = y_j \text{ for some } j \neq i\} (i = 1, \ldots, k)$. Clearly, $\{x_i, z\} \in E(H')$ can hold for at most two of the vertices $x_i$ and, therefore, $E_i \cap E(H') \neq \varnothing$ for at least $k-2$ of the sets $E_i$. Further, one easily infers from the definition of $c$ that $c(e) = \tau^2+\tau$ for all $e \in E_i$ $(i = 1, \ldots, k)$. A similar argument holds for $E'_i = \{\{x'_i, v\} : v = w \text{ or } v = y'_j \text{ for some } j \neq i\} (i = 1, \ldots, k)$. It follows that $H'$ contains at least $2(k-2)$ edges $e$ with $c(e) = \tau^2 + \tau$, thus establishing (21).

From (20) and (21) we obtain

$$\frac{c(H')}{c_{\min}} \geq \frac{2k-4}{2k+3}(\tau^2+\tau) \to \tau^2+\tau \text{ for } k \to \infty,$$

which shows that, if the conjectured inequality (15′) is true, then it is best possible.

**3. Steiner trees.** Let $(X, c)$ be a metric space and $U$ be a finite subset of $X$ with $|U| = n \geq 2$. A tree $S$ with $V(S) \subseteq X$ is called a *Steiner tree* for $U$ if $U \subseteq V(S)$ and if all end vertices of $S$ are members of $U$. Let $T$ be a minimum-spanning tree for $U$. It is well known that

$$c(T) \leq 2\left(1 - \frac{1}{n}\right) c(S)$$

for any Steiner tree $S$ for $U$; see, e.g., Lengauer (1990). In other words, if we choose a minimum-spanning tree as a heuristic solution for the Steiner tree problem (i.e., the problem of minimizing $c(S)$ over all Steiner trees $S$ for $U$), then 2 is the corresponding factor of performance guarantee. The next theorem shows how this factor decreases when we make the additional assumption that, for some $\tau$ with $0 < \tau \leq 1$, the set $U$ satisfies the following "$\tau$-inequality with respect to $X - U$":

$$c(u, v) \leq \tau(c(u, x) + c(x, v)) \quad \text{for all } u, v \in U, x \in X - U. \tag{22}$$

THEOREM 3. *Let $(X, c)$ be a metric space and let $U$ be a finite subset of $X$ with $|U| = n \geq 2$. Assume that, for some $\tau (0 < \tau \leq 1), U$ satisfies the $\tau$-inequality with respect to $X - U$. Let $c_{\inf} = \inf\{c(S) : S$ is a Steiner tree for $U\}$ and let $T$ be a minimum-spanning tree for $U$. Then*

$$c(T) \leq 2\tau\left(1 - \frac{1}{n}\right) c_{\inf} \quad \text{if } \tau \geq \frac{n}{2(n-1)}, \text{ and otherwise}$$
$$c(T) = c_{\inf} \quad \text{if } \tau \leq \frac{n}{2(n-1)}.$$

*Proof.* Let $S$ be a Steiner tree for $U$. Note that $S$ is the edge disjoint union of nontrivial trees $S_i\,(i = 1,\dots,q)$ such that $V(S_i) \cap U$ is exactly the set of end vertices of $S_i\,(i = 1,\dots,q)$. Let $U_i := V(S_i) \cap U$. Then, clearly, $S_i$ is a Steiner tree for $U_i$. Let $T_i$ be a minimum-spanning tree for $U_i$ and let $n_i = |U_i|$. Then $2 \le n_i \le n(i = 1,\dots,q)$. We may assume that $|V(S_i)| \ge 3$ for $i = 1,\dots,p$ and $|V(S_i)| = 2$ for $i = p+1,\dots,q(0 \le p \le q)$. Then, clearly, $S_i = T_i\,(i = p+1,\dots,q)$.

Now, let $1 \le i \le p$. Choose end vertices $a_i, b_i \in S_i$ such that the cost of the $a_i, b_i$-path in $S_i$ is maximum. Denote this $a_i, b_i$-path by $P_{i,1}\,(i = 1,\dots,p)$. Now, extend this path to any Eulerian walk of the multigraph obtained by doubling each edge of the tree $S_i$. Specifically, for each $i(1 \le i \le p)$, enumerate the vertices of $U_i$ by $x_{i,1} = a_i, x_{i,2},\dots,x_{i,n_i-1}, x_{i,n_i} = b_i$ in such a way that, for each edge $e$ of $S_i$, there are exactly two paths among $P_{i,1}$ and the $(x_{i,j-1}, x_{i,j})$-paths $P_{i,j}\,(j = 2,\dots,n_i)$ of $S_i$ that contain $e$. Hence

$$2c(S_i) = \sum_{j=1}^{n_i} c(P_{i,j}) \le n_i c(P_{i,1}) \qquad (1 \le i \le p). \tag{23}$$

Note that each path $P_{i,j}$ consists of at least two edges. Moreover, the end vertices of $P_{i,j}$ are members of $U$ while all inner vertices of $P_{i,j}$ are in $X - U$. From this, together with (22) and the fact that $(X, c)$ is a metric space, we conclude $c(x_{i,j-1}, x_{i,j}) \le \tau c(P_{i,j})(2 \le j \le n_i, 1 \le i \le p)$. Hence, for $i = 1,\dots,p$

$$\begin{aligned} c(T_i) &\le \sum_{j=2}^{n_i} c(x_{i,j-1}, x_{i,j}) \\ &\le \sum_{j=2}^{n_i} \tau c(P_{i,j}) = \tau \sum_{j=1}^{n_i} c(P_{i,j}) - \tau c(P_{i,1}) \\ &\le 2\tau c(S_i) - \tau \frac{2}{n_i} c(S_i) = 2\tau\left(1 - \frac{1}{n_i}\right) c(S_i) \quad \text{by (23)} \\ &\le 2\tau\left(1 - \frac{1}{n}\right) c(S_i). \end{aligned}$$

The last inequality holds because $n_i \le n\,(i = 1,\dots,p)$. Hence

$$c(T) \le \sum_{i=1}^{q} c(T_i) \le 2\tau\left(1 - \frac{1}{n}\right) \sum_{i=1}^{p} c(S_i) + \sum_{i=p+1}^{q} c(S_i).$$

Consequently, if $\tau \le n/2(n-1)$, then $2\tau(1 - 1/n) \le 1$, and thus $c(T) \le \sum_{i=1}^{q} c(S_i) = c(S)$. Because this holds for all Steiner trees $S$ for $U$ and since $T$ itself is a Steiner tree for $U$, we conclude $c(T) = c_{\text{inf}}$. If, on the other hand, $\tau \ge n/2(n-1)$ then $c(T) \le 2\tau(1 - 1/n)\sum_{i=1}^{q} c(S_i) = 2\tau(1 - 1/n)c(S)$, and thus $c(T) \le 2\tau(1 - 1/n)c_{\text{inf}}$. □

Let $n \ge 2$ and $1 \ge \tau \ge n/2(n-1)$. The following examples show that the bound of Theorem 3 is best possible. Let $U = \{u_1,\dots,u_n\}$ and $X = U \cup \{x\}$. Define distances by $c(u_i, u_j) = 2\tau(i \ne j)$ and $c(x, u_i) = 1(i = 1,\dots,n)$. This meets the requirement of Theorem 3, and we have $c(T) = 2\tau(n-1)$ for a minimum-spanning tree $T$ for $U$ and $c(S) = n$ for a minimum-Steiner tree $S$ for $U$. Hence $c(T) = 2\tau(1 - 1/n)c_{\text{inf}}$.

## 4. Anticlustering.

Let $X = \{1,\dots,n\}$ with $n = k \cdot m(k, m \in \mathbb{N}, k, m \ge 2)$ and let $G = (X, \binom{X}{2})$ be the complete graph with vertex set $X$. Given a distance function $c$ on $X$, the *anticlustering problem* asks for a partition of $X$ into $k$ disjoint sets $A_1, \dots, A_k$ of size $m$ (called *anticlusters*) such that

$$\sum_{r=1}^{k} c(A_r) \to \max .$$

By $c_{\max}$ we denote the value $\sum_{r=1}^{k} c(A_r)$ for an optimal solution $A_1, \dots, A_k$ of the anticlustering problem. For $m = 2$, the anticlustering problem is the well-known maximum weighted matching problem, which can be solved in $0(n^3)$ time; cf. Papadimitriou and Steiglitz (1982). For fixed $m \geq 3$, the anticlustering problem is known to be $NP$-complete, since it contains "partition into complete subgraphs of order $m$" as a special case; see Garey and Johnson (1979). For information on the complexity of related problems, cf. Johnson et al. (1984) and Khellaf (1987). Henceforth, we assume $m \geq 3$. The following approximation algorithm for the anticlustering problem (with fixed size $m$ of anticlusters) is due to Feo and Khellaf (1990). If $m$ is even, find a perfect matching $M$ of $G$ such that $c(M)$ is maximum. Thereafter, partition $M$ arbitrarily into $k$ sets $M_1, \dots, M_k$ of $m/2$ edges each, and choose the corresponding vertex sets $A_h = \{i \in X : i \text{ is incident to some edge of } M_h\}$ $(h = 1, \dots, k)$ as anticlusters of an approximative solution. For odd $m$, a similar procedure is used: among all matchings of $G$ with $(n-k)/2 = (m-1)k/2$ edges, find a matching $M$ such that $c(M)$ is maximum; partition $M$ arbitrary into $k$ sets $M_1, \dots, M_k$ of $(m-1)/2$ edges each, and add to each $M_h$ an arbitrary unmatched vertex, thus yielding vertex sets $A_h$ of the required size $m$. In either case, we denote the value $\sum_{r=1}^{k} c(A_r)$ for the anticlusters $A_1, \dots, A_k$ obtained by this approximation algorithm by $c_{\text{approx}}$.

Now we extend the result of Feo and Khellaf (1990) from the case $\tau = 1$ (the standard triangle inequality) to arbitrary $\tau \geq 1/2$.

THEOREM 4. *If the distances $c_{i,j}$ satisfy the $\tau$-inequality (for some $\tau \geq 1/2$), then*

$$\frac{c_{\max}}{c_{\text{approx}}} \leq \begin{cases} \dfrac{2\tau(m-1)}{m-2+2r} & \textit{if } m \textit{ is even,} \\[2ex] \dfrac{2\tau m}{m-1+2\tau} & \textit{if } m \textit{ is odd.} \end{cases}$$

For the proof of this theorem, we need the following lemma.

LEMMA 3. *Let $G$ be a complete graph with vertex set $\{1, \dots, n\}$ and let $c$ be a function assigning a nonnegative real number $c_{i,j}$ to every edge $\{i, j\}$ of $G$. Let $M$ be an arbitrary matching of $G$ with $\lfloor n/2 \rfloor$ edges. Then the following inequalities hold:*

(24) *If $M$ is a maximum matching, then*

$$\begin{array}{ll} c(G) \leq (n-1)c(M) & \textit{if } n \textit{ is even, and} \\ c(G) \leq nc(M) & \textit{if } n \textit{ is odd.} \end{array}$$

(25) *If the $c_{i,j}$ satisfy the $\tau$-inequality for some $\tau \geq 1/2$, then*

$$\begin{array}{ll} \dfrac{n-2+2\tau}{2\tau}\, c(M) \leq c(G) & \textit{if } n \textit{ is even, and} \\[2ex] \dfrac{n-1+2\tau}{2\tau}\, c(M) \leq c(G) & \textit{if } n \textit{ is odd.} \end{array}$$

*Proof.* The assertion (24) immediately follows from the fact that $G$ can be decomposed into $n-1$ (or $n$) matchings if $n$ is even (or odd, respectively).

As to (25), assume $M = \{\{1,2\},\{3,4\},\dots,\{n-1,n\}\}$ if $n$ is even and $M = \{\{1,2\},\{3,4\},\dots,\{n-2,n-1\}\}$ if $n$ is odd. By the $\tau$-inequality,

$$c_{i,i+1} \le \tau(c_{i,j} + c_{j,i+1}) \quad \text{for all } \{i,i+1\} \in M \text{ and } j \ne i, i+1. \tag{26}$$

Hence

$$(n-2)c_{i,i+1} \le \sum_{j \ne i,i+1} \tau(c_{i,j} + c_{j,i+1}) \quad \text{for all } \{i,i+1\} \in M.$$

If $n$ is even, summing up all these inequalities for $\{i,i+1\} \in M$ yields

$$(n-2)c(M) \le 2\tau(c(G) - c(M)).$$

This establishes (25) for even $n$.

If $n$ is odd, let $G'$ result from $G$ by deletion of the vertex $n$. Since the number of vertices of $G'$ is even, we have

$$\frac{n-3+2\tau}{2\tau} c(M) \le c(G').$$

Consider (26) for $j = n$. Then summation over all $\{i,i+1\} \in M$ yields

$$c(M) \le \tau \sum_{i=1}^{n-1} c_{i,n}.$$

Hence

$$\frac{n-3+2\tau}{2\tau} c(M) + \frac{c(M)}{\tau} \le c(G') + \sum_{i=1}^{n-1} c_{i,n} = c(G),$$

which settles (25) for odd $n$. □

*Proof of Theorem 4.* Let $A_i\,(i = 1,\dots,k)$ be the anticlusters found by the approximation algorithm. Let $M$ denote the maximum matching of $G$ that is used by the approximation algorithm for the construction of $A_1,\dots,A_k$. Denote by $M_i$ the edges of $M$ placed by the algorithm into $A_i$. Let $A_h^*\,(h = 1,\dots,k)$ be the clusters of an optimal solution and let $M_h^*$ be a maximum matching of the complete graph with vertex set $A_h^*$ (with respect to the distances $c_{i,j}$). Let $m' = m-1$ if $m$ is even and $m' = m$ otherwise. By (24), $c(A_i^*) \le m'c(M_i^*)$ and thus

$$c_{\max} = \sum_{i=1}^{k} c(A_i^*) \le m' \sum_{i=1}^{k} c(M_i^*) \le m'c(M).$$

The last inequality holds since $\cup_{i=1}^{k} M_i^*$ is a matching of $G$. By (25)

$$c(M_i) \le \frac{2\tau}{m'-1+2\tau} c(A_i), \qquad i = 1,\dots,k$$

and thus

$$\begin{aligned} c_{\max} &\le m'c(M) = m' \sum_{i=1}^{k} c(M_i) \\ &\le m' \frac{2\tau}{m'-1+2\tau} \sum_{i=1}^{k} c(A_i) = \frac{2\tau m'}{m'-1+2\tau} c_{\text{approx}}. \end{aligned}$$ □

The following example shows that the upper bound on $c_{\max}/c_{\text{approx}}$ in Theorem 4 is best possible for each $\tau \geq 1/2$. We just treat the case when $m$ is even; for odd $m$, a similar example can be found. Let $k \geq m/2$ and let $X = \{1, \ldots, n\}$ with $n = k \cdot m$. Further, let $M = \{\{1,2\}, \{3,4\}, \ldots, \{n-1, n\}\}$ and $A_h = \{j \in \mathbb{N} : (h-1)m < j \leq hm\}(h = 1, \ldots, k)$. For $i \neq j$, distances $c_{i,j}$ are defined by

$$c_{i,j} = \begin{cases} 1 & \text{if } \{i,j\} \in M \text{ or if } i \text{ and } j \text{ are in distinct sets } A_h, \\ \dfrac{1}{2\tau} & \text{otherwise.} \end{cases}$$

It follows that the $\tau$-inequality is satisfied. Since $M$ is a maximum matching, application of the approximation algorithm may result in the anticlusters $A_1, \ldots, A_k$. Then

$$\begin{aligned} c_{\text{approx}} &= \sum_{i=1}^{k} c(A_i) = k\left(\frac{m}{2} + \frac{1}{2\tau}\left(\frac{m(m-1)}{2} - \frac{m}{2}\right)\right) \\ &= \frac{km(m-2+2\tau)}{4\tau}. \end{aligned}$$

On the other hand, as long as $k \geq m/2$, one can choose a partition into anticlusters $A_1^*, \ldots, A_k^*$ of size $m$ such that each $A_i^*$ intersects any $A_j$ in an edge of $M$ or in at most one point, whence

$$c_{\max} = k \cdot \frac{m(m-1)}{2},$$

yielding

$$\frac{c_{\max}}{c_{\text{approx}}} = \frac{2\tau(m-1)}{m-2+2\tau}.$$

## REFERENCES

H.-J. Bandelt, Y. Crama, and F. C. R. Spieksma (1991), *Approximation algorithms for multidimensional assignment problems with decomposable costs*, Discrete Appl. Math., to appear.

T. A. Feo, and M. Khellaf (1990), *A class of bounded approximation algorithms for graph partitioning*, Networks, 20, 181–195.

M. R. Garey and D. S. Johnson (1979), *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

D. S. Johnson, C. H. Papadimitriou, P. Seymour, and M. Yannakakis (1984), *The complexity of multiway cuts*, AT & T Bell Laboratories, Murray Hill, NJ, manuscript.

M. Khellaf (1987), *On the partitioning of graphs and hypergraphs*, Ph.D. thesis, University of California, Berkeley, CA.

T. Lengauer (1990), *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley–Teubner, Chichester.

C. H. Papadimitriou and K. Steiglitz (1982), *Combinatorial Optimization*, Prentice-Hall, Englewood Cliffs, NJ.

M. Sekanina (1960), *On an ordering of the set of vertices of a connected graph*, Spisy Přirod. Fak. Univ. Brno, 42, 137–141.

H. Walther and H.-J. Voss (1974), *Über Kreise in Graphen*, Deutscher Verlag der Wisenschaften, Berlin.

SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 17–32, February 1995

# DELTA-MATROIDS, JUMP SYSTEMS, AND BISUBMODULAR POLYHEDRA [*]

ANDRÉ BOUCHET[†] AND WILLIAM H. CUNNINGHAM[‡]

**Abstract.** This paper relates an axiomatic generalization of matroids, called a jump system, to polyhedra arising from bisubmodular functions. Unlike the case for usual submodularity, the points of interest are not all the integral points in the relevant polyhedron but form a subset of them. However, it is shown that the convex hull of the set of points of a jump system is a bisubmodular polyhedron, and that the integral points of an integral bisubmodular polyhedron determine a (special) jump system. The authors prove addition and composition theorems for jump systems, which have several applications for delta-matroids and matroids.

**Key words.** delta-matroid, jump system, bisubmodular polyhedron, composition theorem

**AMS subject classifications.** primary 05B35, secondary 90C99

## 1. Introduction.

Matroids are important as a unifying structure in pure combinatorics, as well as a useful model in the theory of algorithms and in combinatorial optimization. (See Bixby and Cunningham [1] for a survey of the latter aspects.) Delta-matroids constitute an interesting generalization and have been introduced only recently. Many of the nice properties associated with matroids (greedy algorithm, polyhedral description, interesting examples) extend to delta-matroids. In the present paper we begin by reviewing some of this work. Then we prove a new composition theorem for delta-matroids. It has several applications, including constructions for matroids. An important theme is to identify in which of the applications the composition is algorithmically constructible.

The polyhedral aspects of matroids, developed more than 20 years ago by Edmonds [11], led him to a different generalization, integral polymatroids. In a certain sense there are two views of an integral polymatroid; first, it is a polyhedron $P$, and second, it is a set $\mathcal{F}$ of integral points. There is a simple relation between the two views—$P$ is the convex hull of $\mathcal{F}$, and $\mathcal{F}$ is the set of integral points in $P$. In this paper we introduce jump systems, a common generalization of delta-matroids and (the second view of) integral polymatroids. A jump system is defined by a set $\mathcal{F}$ of integral points, but it is not generally true that it is the set of integral points in its convex hull. We present some examples of jump systems and prove an addition theorem, that implies the composition theorem for delta-matroids.

Although jump systems cannot be defined via polyhedra, there is an important subclass that can. These arise from (integral) bisubmodular polyhedra, introduced by Dunstan and Welsh [10] in 1973 in a paper that seems to have been fully appreciated only recently. We prove that the integral points in such a polyhedron determine a jump system. Moreover, there is a partial converse—if $\mathcal{F}$ is a set of integral points determining a jump system, then the convex hull of $\mathcal{F}$ is an integral bisubmodular

[*] Received by the editors December 2, 1991; accepted for publication (in revised form) January 3, 1994.

[†] Département de Mathématiques et Informatique, Université du Maine, 72017 Le Mans, Cedex, France. Partially supported by a National Sciences and Engineering Research Council of Canada International Scientific Exchange Award at Carleton University.

[‡] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. Partially supported by a National Sciences and Engineering Research Council of Canada operating grant.

polyhedron. So it is true that a polyhedron with integral vertices is bisubmodular if and only if the integral points in it form a jump system.

Throughout this paper $S$, with or without subscripts, is a finite set. We use $\mathbf{R}$, $\mathbf{R}_+$, $\mathbf{Z}$, and $\mathbf{Z}_+$ to denote the sets of real numbers, non-negative real numbers, integers, and non-negative integers, respectively. For $x \in \mathbf{R}^S$ and $A \subseteq S$, we often use $x(A)$ as an abbreviation for $\sum(x_j : j \in A)$. For $c, x \in \mathbf{R}^S$ we write $cx$ to mean $\sum(c_j x_j : j \in S)$. For $x \in \mathbf{R}^S$ and $A \subseteq S$, we use $x_{|A}$ to denote the restriction of $x$ to $A$, that is, the vector $x' \in \mathbf{R}^A$ such that $x'_j = x_j$ for all $j \in A$. Finally, we use the symbol $A$ also to denote the incidence vector of $A$, that is, the vector $x \in \mathbf{R}^S$ such that $x_j = 1$ if $j \in A$ and $x_j = 0$ if $j \notin A$.

**2. Delta-matroids.** Let $\mathcal{F}$ be a family of subsets of a finite set $S$. Then $(S, \mathcal{F})$ is a *delta-matroid* if the following *symmetric exchange axiom* is satisfied:

(SEA) If $F_1, F_2 \in \mathcal{F}$ and $j \in F_1 \Delta F_2$ then there is $k \in F_1 \Delta F_2$ such that $F_1 \Delta \{j, k\} \in \mathcal{F}$.

(Here and elsewhere $\Delta$ denotes symmetric difference.) These structures have been introduced by Bouchet [2]. Essentially equivalent structures were independently considered by Dress and Havel [8] and by Chandrasekaran and Kabadi [5]. A main motivation for their study is that, if $\mathcal{F}$ is the family of bases of a matroid on $S$, then $(S, \mathcal{F})$ is a delta-matroid. In fact, matroids are precisely the delta-matroids for which all members of $\mathcal{F}$ have the same cardinality. (We remark that throughout the paper we use "matroid" to mean a matroid defined by its family of bases.) In addition to these examples, we mention a few others.

**Matching delta-matroids.** Let $G = (V, E)$ be a graph, let $S = V$, and let $F \in \mathcal{F}$ if and only if there is a matching of $G$ covering precisely the elements of $F$. Then $(S, \mathcal{F})$ is a delta-matroid. This can be proved using augmenting path arguments.

**Twisting.** Let $(S, \mathcal{F})$ be a delta-matroid, and let $N \subseteq S$. Let $\mathcal{F} \Delta N$ denote $\{F \Delta N : F \in \mathcal{F}\}$. Then $(S, \mathcal{F} \Delta N)$ is a delta-matroid. For example, we can get delta-matroids by applying twisting to a matroid. In one case we get again a matroid; namely, when $N = S$, we get the dual matroid.

**Linear delta-matroids.** Let $M = (m_{ij} : i \in S, j \in S)$ be a skew-symmetric matrix over a field. Define $\mathcal{F}$ by $F \in \mathcal{F}$ if and only if the principal submatrix $(m_{ij} : i \in F, j \in F)$ is non-singular. Then $(S, \mathcal{F})$ is a delta-matroid. The proof of this result is not trivial; see Bouchet [3], where it is also generalized. (For example, a symmetric matrix can also be used.)

Another basic fact is that, if $(S, \mathcal{F})$ is a delta-matroid and $\mathcal{F}'$ is the family of maximal members of $\mathcal{F}$, then $(S, \mathcal{F}')$ is a matroid. This and twisting can be used to justify a greedy algorithm for optimizing any linear function over $\mathcal{F}$. Namely, $|c|(F \Delta N) = c(F) - c(N)$, where $N = \{j : c_j < 0\}$. Therefore, we can apply the matroid greedy algorithm to the maximal members of $\mathcal{F} \Delta N$ with weight function $|c|$. Translating that algorithm into one operating directly on $(S, \mathcal{F})$ and $c$, we get the following procedure. It appears in [2] and [5], but a similar kind of greedy algorithm can be found in Dunstan and Welsh [10].

**Greedy Algorithm for Delta-Matroids**

**Input:** Delta-matroid $(S, \mathcal{F})$ and weight vector $c \in \mathbf{R}^S$.
**Objective:** To find $F \in \mathcal{F}$ such that $c(F)$ is maximum.

```
begin
    order $S$ as $\{e_1, e_2, \dots, e_n\}$ so that $|c_{e_1}| \geq |c_{e_2}| \geq \cdots \geq |c_{e_n}|$;
    for $i = 1$ to $n + 1$ let $T_i = \{e_i, \ldots, e_n\}$;
    $J \leftarrow \emptyset$;
    for $i = 1$ to $n$
        if $c_{e_i} \geq 0$ and there exists $F \in \mathcal{F}$ with $J \cup \{e_i\} \subseteq F \subseteq J \cup T_i$
            then $J \leftarrow J \cup \{e_i\}$;
        if $c_{e_i} < 0$ and there does not exist $F \in \mathcal{F}$ with $J \subseteq F \subseteq J \cup T_{i+1}$
            then $J \leftarrow J \cup \{e_i\}$;
end.
```

Notice that to implement this algorithm we need to be able to answer the question, given disjoint subsets $A$, $B$ of $S$,

(2.1) Does there exist $F \in \mathcal{F}$ with $A \subseteq F \subseteq S \setminus B$ ?

A more general question is to ask for the value $f(A, B)$, defined to be $\max_{F\in\mathcal{F}}(|F \cap A| - |F \cap B|)$, since the answer to (2.1) is "yes" exactly when $f(A, B) = |A|$. However, the two questions are algorithmically equivalent because $f(A, B)$ can be computed by the greedy algorithm with $c_j = 1$ for $j \in A$, $-1$ for $j \in B$, and 0 otherwise. We consider the existence of an efficient subroutine to evaluate the function $f$ (or answer the question (2.1)) to be the measure of algorithmic tractability of the delta-matroid. (If $(S, \mathcal{F})$ is a matroid with rank function $r$, a simple argument shows that $f(A, B) = r(A) + r(S \setminus B) - r(S)$. Since $r(A) = f(A, \emptyset)$, it follows that this oracle is available for a matroid exactly when the usual one is available.)

**Composition of delta-matroids.** Our main result on delta-matroids is a composition theorem. We define the *composition* of delta-matroids $(S_0, \mathcal{F}_0)$, $(S_1, \mathcal{F}_1)$ to be $(S, \mathcal{F})$ where $S = S_0 \Delta S_1$ and $\mathcal{F} = \{F_0 \Delta F_1 : F_0 \in \mathcal{F}_0,\ F_1 \in \mathcal{F}_1,\ F_0 \cap S_1 = F_1 \cap S_0\}$. That is, each feasible set is a symmetric difference of two feasible sets, one from each of the initial delta-matroids, that agree on $S_0 \cap S_1$. The proof that this construction gives a delta-matroid is our original one, which we include because of its algorithmic flavour. However, the reader is warned that the next section contains an easier proof of a more general result, so he may want to skip this proof on a first reading.

THEOREM 2.1. *The composition of delta-matroids is a delta-matroid.*

*Proof.* We consider $F, G \in \mathcal{F}$ and $j \in F \Delta G$, and we search for $k \in F \Delta G$ such that $F \Delta \{j, k\} \in \mathcal{F}$. There exist $F_0, G_0 \in \mathcal{F}_0$ and $F_1, G_1 \in \mathcal{F}_1$ such that $F = F_0 \Delta F_1$ and $G = G_0 \Delta G_1$. We also consider $S' = S_0 \cap S_1$, $F' = F_0 \cap S' = F_1 \cap S'$ and $G' = G_0 \cap S' = G_1 \cap S'$. For any integer $i$ we let $F_i$, $G_i$, $\mathcal{F}_i$ be respectively equal to $F_0$, $G_0$, $\mathcal{F}_0$ if $i$ is even or to $F_1$, $G_1$, $\mathcal{F}_1$ if $i$ is odd.

The element $j$ belongs to $F_0 \Delta G_0 \Delta F_1 \Delta G_1$. By symmetry we may assume that $j \in F_1 \Delta G_1$. Applying (SEA) to $F_1, G_1 \in \mathcal{F}_1$ and $j \in F_1 \Delta G_1$ we can find $z \in F_1 \Delta G_1$ such that $F_1 \Delta \{j, z\} \in \mathcal{F}_1$. If $z \notin S'$ we have $F_1 \Delta \{j, z\} \Delta F_0 = F \Delta \{j, z\} \subseteq S$, and the property is proved with $k = z$. From now on we assume that $z \in S'$, so that $z \in F' \Delta G'$.

We consider a sequence $U = (j_1, j_2, \ldots, j_r)$ of pairwise distinct elements belonging to $F' \Delta G'$ with $j_1 = z$. For $0 \leq i \leq r$ we let $U_i = \{j, j_1, j_2, \ldots, j_i\}$ if $i$ is odd or $U_i = \{j_1, j_2, \ldots, j_i\}$ if $i$ is even and suppose that $\Phi_i = F_i \Delta U_i \in \mathcal{F}_i$. The conditions are satisfied if $U = (j_1)$ because $\Phi_0 = F_0$ and $\Phi_1 = F_1 \Delta \{j, z\}$. From now on we suppose that the length of $U$ is maximal.

We have $(\Phi_{r-1} \Delta G_{r-1}) \cap S' = (F_{r-1} \Delta U_{r-1} \Delta G_{r-1}) \cap S' = F' \Delta G' \Delta U_{r-1}$. The element $j_r$ belongs to $F' \Delta G'$ but does not belong to $U_{r-1}$. Therefore $j_r \in (\Phi_{r-1} \Delta G_{r-1}) \cap S' \subseteq \Phi_{r-1} \Delta G_{r-1}$. We apply (SEA) to $\Phi_{r-1}, G_{r-1} \in \mathcal{F}_{r-1}$ and $j_r \in \Phi_{r-1} \Delta G_{r-1}$.

This yields an element $j_{r+1} \in \Phi_{r-1}\Delta G_{r-1}$ such that $\Phi_{r-1}\Delta\{j_r, j_{r+1}\} \in \mathcal{F}_{r-1}$. We let $U_{r+1} = U_{r-1}\Delta\{j_r, j_{r+1}\}$ and $\Phi_{r+1} = \Phi_{r-1}\Delta\{j_r, j_{r+1}\}$.

We claim that either $j_{r+1} \notin S'$ or $j_r = j_{r+1}$. If this is not true, we have $j_{r+1} \in (\Phi_{r-1}\Delta G_{r-1}) \cap S' = F'\Delta G'\Delta U_{r-1}$. Since $U_{r-1} \subseteq F'\Delta G'$, this implies that $j_{r+1}$ is distinct from $j_1$, $j_2, \ldots$, $j_r$. Therefore $(j_1, j_2, \ldots, j_{r+1})$ satisfies the same properties as $U$, which contradicts the maximality of $U$.

If either $j_{r+1} \notin S'$ or $j_{r+1} = j_r$, we have $\Phi_{r+1} \cap S' = \Phi_r \cap S' = U_r\Delta F'$. Since $\Phi_{r+1} \in \mathcal{F}_{r+1}$ and $\Phi_r \in \mathcal{F}_r$, we have $\Phi_r\Delta\Phi_{r+1} \in \mathcal{F}$. If $j_{r+1} \notin S'$ we verify that $\Phi_r\Delta\Phi_{r+1} = F\Delta\{j, j_{r+1}\}$ and $j_{r+1} \in F\Delta G$, which proves the property with $k = j_{r+1}$. If $j_{r+1} = j_r$ we have $\Phi_r\Delta\Phi_{r+1} = F\Delta\{j\}$, which proves the property with $k = j$. □

Given a set $l$ of disjoint pairs of $S$ and a subset $F \subseteq S$ we abuse the notation $F\Delta l$ to represent the symmetric difference of $F$ with the union of the pairs that belong to $l$. Let $(S, \mathcal{F})$ be a delta-matroid. For $F, F' \in \mathcal{F}$, a *linking* $L$ of $(F, F')$ is a partition of $F\Delta F'$ into pairs such that $F\Delta l \in \mathcal{F}$ for all $l \subseteq L$. We say that $(S, \mathcal{F})$ is *linkable* if there exists a linking of $(F, F')$ for all $F, F' \in \mathcal{F}$. This generalizes the notion of strong base orderability (see Welsh [23]) for matroids.

THEOREM 2.2. *The composition of linkable delta-matroids is a linkable delta-matroid.*

*Proof.* The notation is the same as in the proof of Theorem 2.1. For $i = 0, 1$, let $L_i$ be a linking of $(F_i, G_i)$. Let $H$ be the graph defined over the vertex-set $S_0 \cup S_1$ and the edge-set $L_0 \cup L_1$. Each vertex of $H$ has degree 0, 1, or 2, and no vertex in $S_0\Delta S_1$ has degree 2. Hence the components of $H$ are paths and circuits, and each path ends in $S_0\Delta S_1$. Let $\mathcal{P}$ be the set of the components of $H$ that are paths. Let $L = \{\{s, t\} : s \text{ and } t \text{ are the ends of a path in } \mathcal{P}\}$. We prove that $L$ is a linking of $F\Delta G$. Let $l = \{(s^1, t^1), (s^2, t^2), \ldots, (s^k, t^k)\} \subseteq L$. Let $P^j$ be the path in $\mathcal{P}$ that ends at $s^j$ and $t^j$, for $1 \le j \le k$. Let $l_i^j = L_i \cap P^j$, for $i = 0, 1$. Since $L_i$ is a linking of $(F_i, G_i)$, we have

$$\text{(i)} \qquad F_i' = F_i\Delta(l_i^1\Delta l_i^2 \ldots l_i^k) \in \mathcal{F}_i.$$

Notice that $X\Delta l_0^j\Delta l_1^j = X\Delta\{s^j, t^j\}$ holds for all $X \subseteq S$ and $1 \le j \le k$. Hence it follows from (i) that $F\Delta l = F_0'\Delta F_1'$, so $F\Delta l \in \mathcal{F}$. □

*Remark.* Matching delta-matroids are examples of linkable delta-matroids. But for matching delta-matroids an even stronger property holds. For $F, F' \in \mathcal{F}$, there is a partition of $F\Delta F'$ that is a linking of both $(F, F')$ and $(F', F)$.

**Composition of matroids.** If we apply the composition to two matroids, it is clear that the composed delta-matroid is not necessarily a matroid. However, composition can be combined with twisting to provide a matroid construction.

THEOREM 2.3. *If $(S_0, \mathcal{F}_0)$, $(S_1, \mathcal{F}_1)$ are matroids, then the composition $(S, \mathcal{F})$ of $(S_0, \mathcal{F}_0)$ with $(S_1, \mathcal{F}_1\Delta(S_0 \cap S_1))$ is a matroid (provided $\mathcal{F}$ is non-empty).*

*Proof.* $(S, \mathcal{F})$ is a delta-matroid by Theorem 2.1, so we need only show that the members of $\mathcal{F}$ all have the same cardinality. But

$$\mathcal{F} = \{(F_0 \cup F_1) \setminus (S_0 \cap S_1) : F_0 \in \mathcal{F}_0,\ F_1 \in \mathcal{F}_1,\ F_0 \cap F_1 = \emptyset,\ F_0 \cup F_1 \supseteq S_0 \cap S_1\}.$$

Thus $F \in \mathcal{F}$ implies $|F| = |F_0| + |F_1| - |S_0 \cap S_1|$, and we are done. □

In fact, this matroid composition can be obtained from standard constructions: $(S, \mathcal{F}) = \big((S, \mathcal{F}_0) + (S, \mathcal{F}_1)\big)/(S_0 \cap S_1)$, where $+$ denotes matroid union [23] and $/$ denotes contraction. This composition was investigated in [6] and [22]. It is easy to derive a formula for its rank function $r$ in terms of the rank functions $r_0, r_1$ of $(S, \mathcal{F}_0)$, $(S, \mathcal{F}_1)$, namely,

$$r(A) = \min_{X \subseteq S_0 \cap S_1} \big( r_0(X \cup (A \cap S_0)) + r_1(X \cup (A \cap S_1)) - |X| \big).$$

The research in [6], [22] concentrated on cases where $|S| > |S_0|, |S_1|$ and treated the resulting decomposition, which has some nice properties based on connectivity. But the composition also yields constructions for smaller matroids, as follows.

COROLLARY 2.4. *Let $M_0 = (S_0, \mathcal{B}_0)$ and $M_1 = (S_1, \mathcal{B}_1)$ be matroids with $S_1 \subseteq S_0$. Then $\{B \setminus S_1 : B \in \mathcal{B}_0,\ B \cap S_1 \in \mathcal{B}_1\}$, if non-empty, is the family of bases of a matroid on $S = S_0 \setminus S_1$. Its rank function is given by*

$$r(A) = \min_{X \subseteq S_1} \big( r_0(A \cup X) + r_1(X) - |X| \big).$$

COROLLARY 2.5. *Let $M_0 = (S_0, \mathcal{B}_0)$ be a matroid, let $S \subseteq S_0$, let $S_1 = S_0 \setminus S$, and let $k$ be an integer. Then $\{B \cap S : B \in \mathcal{B}_0,\ |B \cap S_1| = k\}$, if non-empty, is the basis family of a matroid on $S$. Its rank function is given by*

$$r(A) = \min(r_0(A),\ r_0(A \cup S_1) - |S_1| + k).$$

*Proof.* We apply Corollary 2.4, taking $M_1$ to be the uniform matroid of rank $k$ on $S_1$. This matroid has rank function $r_1$ defined by $r_1(X) = \min(|X|,\ k)$. In the expression for $r(A)$, we see that if $|X| \geq k$, then we may as well take $X = S_1$, and if $|X| < k$, we may as well take $X = \emptyset$. This leads to the required expression for the rank function. □

We observe that the last construction contains as special cases both contraction and deletion.

**Efficient realization of composed delta-matroids.** Another application of Theorem 2.1 is the following result of Bouchet [4]. We use it and its additional corollary to make an important point about the availability of the oracle for a composition of delta-matroids.

COROLLARY 2.6. *Let $G$ be a bipartite graph with bipartition $\{S, S'\}$, let $(S, \mathcal{F})$ be a delta-matroid, and let $\mathcal{F}' = \{F' \subseteq S' : F'$ is matched in $G$ to a member of $\mathcal{F}\}$. Then $(S', \mathcal{F}')$ is a delta-matroid.*

*Proof.* $(S', \mathcal{F}')$ is the composition of $(S, \mathcal{F})$ with the matching delta-matroid of $G$. □

In the special case of Corollary 2.6 in which $(S, \mathcal{F})$ is a matroid, we get that $(S', \mathcal{F}')$ is also a matroid; this is a classical result (see Welsh [23]). A further specialization gives a "partition" construction for delta-matroids. This is also from [4].

COROLLARY 2.7. *Let $(S, \mathcal{F}_0)$, $(S, \mathcal{F}_1)$ be delta-matroids, and let $\mathcal{F} = \{F_0 \cup F_1 : F_0 \in \mathcal{F}_0,\ F_1 \in \mathcal{F}_1,\ F_0 \cap F_1 = \emptyset\}$. Then $(S, \mathcal{F})$ is a delta-matroid.*

We refer to this construction as the "union" of delta-matroids. Corollary 2.7 can be used to show that Theorem 2.1 is not necessarily algorithmically realizable, in the sense that an oracle for $(S, \mathcal{F})$ may not be available from oracles for $(S_0, \mathcal{F}_0)$, $(S_1, \mathcal{F}_1)$. In Corollaries 2.4 and 2.5 oracles can be constructed efficiently, essentially by means of the matroid partition algorithm, and of course Theorem 2.3 is even easier. We show that in Corollary 2.7 (and hence in Corollary 2.6 and Theorem 2.1), in general, they cannot.

Suppose we are given a graph $G = (V, E)$ and a matroid $M = (V, \mathcal{B})$. Consider the union $(V, \mathcal{F})$ (as in Corollary 2.7) of the matching delta-matroid of $G$ with the dual matroid $M^*$ of $M$. Suppose that we have an oracle for $(V, \mathcal{F})$. Then we can apply the greedy algorithm to find a largest member of $\mathcal{F}$, and in particular to decide whether $V \in \mathcal{F}$. But $V \in \mathcal{F}$ if and only if it is partitionable into a matchable set and a basis of $M^*$, that is, if and only if there is a basis of $M$ that is matchable in $G$. It is well known that deciding whether this is true ("the matroid matching problem" [17]) is not generally solvable in polynomial time. Hence an oracle for the union of $(S, \mathcal{F}_0)$, $(S, \mathcal{F}_1)$ is not constructible in polynomial time from oracles for $(S, \mathcal{F}_0)$, $(S, \mathcal{F}_1)$. The composition is a useful construction, but it is important to distinguish the cases where it is efficiently constructible from those where it is not.

We conclude the section by deriving a new class of delta-matroids from the composition theorem and constructing the relevant oracle. A *red-blue graph* is a graph with each edge coloured either red or blue. A vertex $v$ of a red-blue graph is *bichromatic* or *monochromatic* according to whether $v$ is incident to edges of both colours or not. An *alternating path* of a red-blue graph is a path of length at least one whose edges alternate in colour. Here is a class of delta-matroids arising from red-blue graphs. Notice that the matching delta-matroids form a subclass, arising from the case where there are no blue edges.

PROPOSITION 2.8. *Let $G = (V, E)$ be a red-blue graph, $S$ be the set of monochromatic vertices, and $\mathcal{F} = \{F \subseteq S : F$ is the set of end vertices of a set of vertex-disjoint simple alternating paths$\}$. Then $(S, \mathcal{F})$ is a delta-matroid.*

*Proof.* Let $(S_0, \mathcal{F}_0)$ be the matching delta-matroid of the graph $(S_0, E_0)$, where $S_0 = \{v \in V : v$ is incident to a red edge$\}$ and $E_0$ is the set of red edges. Similarly define $(S_1, \mathcal{F}_1)$ with red replaced by blue. It is easy to see that $(S, \mathcal{F})$ is the composition of $(S_0, \mathcal{F}_0)$ and $(S_1, \mathcal{F}_1)$. □

We describe an efficient construction of the oracle for this class of delta-matroids, due to John Vande Vate. Given disjoint subsets $A, B$ of $S$, delete the vertices of $B$ from $G$. For each bichromatic vertex $w$, split $w$ into two vertices $w_1, w_2$ such that $w_1$ is incident to the red edges previously incident to $w$, and $w_2$ is incident to the blue edges previously incident to $w$. Also join $w_1, w_2$ by a new "white" edge. Let $G'$ be the new graph. Let $P$ be the set of edges of a set of alternating paths determining a feasible set $F$, $A \subseteq F \subseteq S \backslash B$, and let $M$ be $P$ together with the set of white edges corresponding to bichromatic vertices not in any of the paths. Then $M$ is a matching of $G'$ covering all vertices not in $S \backslash (A \cup B)$. Conversely, any such matching of $G'$ determines such a set of alternating paths. Hence the oracle is provided by a matching algorithm. In the next section we will see another example that is based on red-blue graphs but allows the alternating paths to be nonsimple.

**3. Two-step axiom and jump systems.** For vectors $x, y \in \mathbf{Z}^S$, we use the norm $\|x\| = \sum(|x_j| : j \in S)$ and the distance $d(x, y) = \|x - y\|$. For $x, y \in \mathbf{Z}^S$ a *step from $x$ to $y$* is a vector $s \in \mathbf{Z}^S$ such that $\|s\| = 1$ and $d(x + s, y) = d(x, y) - 1$. We denote the set of steps from $x$ to $y$ by $St(x, y)$. A *jump system* is a pair $(S, \mathcal{F})$ where $\mathcal{F} \subseteq \mathbf{Z}^S$ satisfies the following 2-*step axiom*:

(2-SA) If $x, y \in \mathcal{F}$, $s \in \mathrm{St}(x, y)$, and $x + s \notin \mathcal{F}$, then there exists $t \in \mathrm{St}(x + s, y)$with $x + s + t \in \mathcal{F}$.

We begin by considering some simple examples of jump systems.

**Low-dimensional jump systems.** In Fig. 1 we illustrate two choices of $\mathcal{F}$ for the case where $|S| = 2$. In both we denote members of $\mathcal{F}$ with solid dots and nonmembers by hollow or non-existent dots. It is easy to see that in the first case, we have a jump system, whereas in the second case a pair $x, y$ violating (2-SA) is

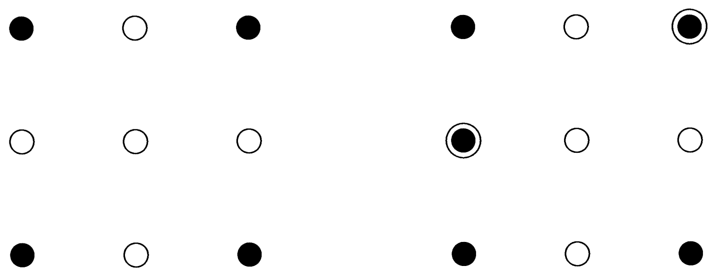


FIG. 1. *The* 2*-step axiom.*

indicated. It is interesting also to consider the case $|S| = 1$, that is, to ask which subsets of the integers satisfy (2-SA). These are the sets that have no gap of size bigger than one; that is, there do not exist two consecutive integers not in $\mathcal{F}$, unless either all elements of $\mathcal{F}$ are bigger than both or all are smaller than both.

**Hyperplanes.** Let $a \in \{0, 1, -1\}^S$, let $b \in \mathbf{Z}$, and let $\mathcal{F} = \{x \in \mathbf{Z}^S : ax = b\}$. It is easy to check that $(S, \mathcal{F})$ is a jump system.

**Delta-matroids.** It is an easy exercise to prove that a pair $(S, \mathcal{F})$ such that $\mathcal{F} \subseteq \{0, 1\}^S$ is a jump system if and only if it is a delta-matroid. (Here, of course, we are identifying subsets of $S$ with their characteristic vectors.)

**Simple operations on jump systems.** Here we mention a few elementary operations that preserve (2-SA).

**Translation.** Let $(S, \mathcal{F})$ be a jump system and let $a \in \mathbf{Z}^S$. Then the translation $(S, \mathcal{F}')$ of $(S, \mathcal{F})$ by $a$ is defined by $\mathcal{F}' = \{x + a : x \in \mathcal{F}\}$, and is clearly a jump system.

**Cartesian product.** Let $S_0, S_1$ be disjoint sets, and let $(S_i, \mathcal{F}_i)$ be a jump system for $i = 0, 1$. Define $S = S_0 \cup S_1$ and $\mathcal{F} = \{F_0 \cup F_1 : F_0 \in \mathcal{F}_0, F_1 \in \mathcal{F}_1\}$. Then $(S, \mathcal{F})$ is a jump system.

**Reflection.** Let $(S, \mathcal{F})$ be a jump system and let $N \subseteq S$. For each $x \in \mathbf{R}^S$, let $x'$ be the vector obtained by reflecting $x$ in the coordinates indexed by $N$, that is, $x'_j = x_j$ if $j \notin N$ and $x'_j = -x_j$ otherwise. Then, where $\mathcal{F}' = \{x' : x \in \mathcal{F}\}$ , it is easy to see that $(S, \mathcal{F}')$ is a jump system. We observe that the twisting operation on delta-matroids is a combination of reflection and translation; more precisely, twisting by $N$ is equivalent to reflecting in the coordinates indexed by $N$ followed by translating by the characteristic vector of $N$.

**Minors.** Let $(S, \mathcal{F})$ be a jump system, let $S' \subseteq S$, let $x \in \mathbf{Z}^{S \setminus S'}$, and let $\mathcal{F}' = \{x' \in \mathbf{Z}^{S'} : (x', x) \in \mathcal{F}\}$. Then $(S', \mathcal{F}')$ is a jump system.

**Intersection with a box.** A *box* is a set of the form $\{x \in \mathbf{R}^S : l \le x \le u\}$, where $l \in (\mathbf{R} \cup \{-\infty\})^S$ and $u \in (\mathbf{R} \cup \{\infty\})^S$. It is easy to see that the intersection of a jump system with a box is again a jump system.

**Restriction or projection.** Let $(S, \mathcal{F})$ be a jump system and let $S' \subseteq S$. Then $(S', \mathcal{F}')$ is a jump system, where $\mathcal{F}' = \{x_{|S'} : x \in \mathcal{F}\}$. We remark that this is not completely obvious, but we leave the (easy) proof to the reader. Also, we point out that the minor operation is now redundant, in the sense that it can be obtained as an intersection with a box followed by a projection. (Namely, intersect with the box defined by $l_j = -\infty$, $u_j = \infty$, $j \in S'$, and $l_j = u_j = x_j$ otherwise, and then restrict to $S'$.)

**Integral polymatroids.** Now we introduce a less trivial example. An *integral polymatroid* is a polyhedron $P = \{x \in \mathbf{R}^S_+ : \ x(A) \leq f(A)$ for all $A \subseteq S\}$, where $f : \{0,1\}^S \to \mathbf{Z}_+$ is normalized ($f(\emptyset) = 0$) and submodular ($f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for all $A, B \subseteq S$).

PROPOSITION 3.1. *If $P$ is an integral polymatroid in $\mathbf{R}^S$, then $P \cap \mathbf{Z}^S$ satisfies* (2-SA).

The proof uses a well-known result, from [11]. Given $x \in P$, where $P$ is determined by $f$, we say that a set $A \subseteq S$ is *$x$-tight* or just *tight* if $x(A) = f(A)$.

LEMMA 3.2. *The union and intersection of tight sets are tight.*

*Proof of Proposition* 3.1 Let $x, y$ be integral points of $P$ and $s$ a step from $x$ to $y$ such that $x + s \notin P$. Then it is easy to see that $s$ must be non-negative, so $s = \{e\}$ for some $e \in S$ such that $x_e < y_e$. It follows that there is an $x$-tight set $A$ such that $e \in A$. Now if $y_j \geq x_j$ for all $j \in A$, then $y(A) > x(A) = f(A)$, a contradiction. So there exists $j \in A$ with $x_j > y_j$. If $x + \{e\} - \{j\} \in P$, we are done, so we may assume that for every such $j$ there is an $x$-tight set $A_j$ with $e \in A_j$ and $j \notin A_j$. The intersection of all these $A_j$ with $A$ is, by Lemma 3.2, an $x$-tight set $B$ such that $e \in B$ and $x_j \leq y_j$ for all $j \in B$. But then $y(B) > x(B) = f(B)$, a contradiction. □

**Sum of jump systems.** The *sum* of jump systems $(S, \mathcal{F}_1)$ and $(S, \mathcal{F}_2)$, defined on the same set $S$, is the pair $(S, \mathcal{F})$ where $\mathcal{F} = \mathcal{F}_1 + \mathcal{F}_2 = \{x + y : \ x \in \mathcal{F}_1, \ y \in \mathcal{F}_2\}$. The simple proof of the following theorem was suggested to us by András Sebő.

THEOREM 3.3. *The sum of two jump systems is a jump system.*

*Proof.* We use the above notation. Let $x, y \in \mathcal{F}_1 + \mathcal{F}_2$ and let $s$ be a step from $x$ to $y$. We have to prove that $x + s \in \mathcal{F}_1 + \mathcal{F}_2$ or there exists a step $t$ from $x + s$ to $y$ such that $x+s+t \in \mathcal{F}_1+\mathcal{F}_2$. We assume that $x+s \notin \mathcal{F}_1+\mathcal{F}_2$ and we search for $t$. Let $x = x_1 + x_2$ and $y = y_1 + y_2$ with $x_1, y_1 \in \mathcal{F}_1$ and $x_2, y_2 \in \mathcal{F}_2$. We have $x_1 + s \notin \mathcal{F}_1$ and $x_2 + s \notin \mathcal{F}_2$ (for example if $x_1 + s \in \mathcal{F}_1$ then $(x_1 + s) + x_2 = x + s \in \mathcal{F}_1 + \mathcal{F}_2$, a contradiction).

We claim that we can find $x'_1 \in \mathcal{F}_1$, $x'_2 \in \mathcal{F}_2$ and a step $t$ satisfying $x + s + t = x'_1 + x'_2$. Since $s$ is a step from $x_1 + x_2$ to $y_1 + y_2$, $s$ is a step from $x_1$ to $y_1$ or a step from $x_2$ to $y_2$. By symmetry we may assume the former. Apply (2-SA) to $x_1, y_1 \in \mathcal{F}_1$ and the step $s$ from $x_1$ to $y_1$. Since $x_1 + s \notin \mathcal{F}_1$ there exists a step $t$ from $x_1 + s$ to $y_1$ such that $x_1 + s + t \in \mathcal{F}_1$. Then $(x_1 + s + t) + x_2 = x + s + t \in \mathcal{F}_1 + \mathcal{F}_2$, which implies the existence of $x'_1$ and $x'_2$.

Choose a triple $(x'_1, x'_2, t)$ that minimizes $d(x'_1, y_1) + d(x'_2, y_2)$. We show that, under this assumption, $t$ is a step from $x + s$ to $y$, proving the theorem. Assume not; we will derive a contradiction. Then $-t$ is a step from $x'_1 + x'_2 = x+s+t$ to $y_1 + y_2 = y$. This implies that $-t$ is a step from $x'_1$ to $y_1$ or a step from $x'_2$ to $y_2$. By symmetry we assume the former. Apply (2-SA) to $x'_1, y_1 \in \mathcal{F}_1$ and the step $-t$. The point $x'_1 - t$ does not belong to $\mathcal{F}_1$ because, if so, we should have $(x'_1 - t) + x'_2 = x + s \in \mathcal{F}_1 + \mathcal{F}_2$, which contradicts the initial assumption. Thus we can find a step $r$ from $x'_1 - t$ to $y_1$ such that $x'_1 - t + r \in \mathcal{F}_1$. This implies $x + s + r = (x'_1 - t + r) + x'_2 \in \mathcal{F}_1 + \mathcal{F}_2$, where $x'_1 - t + r$ is closer to $y_1$ than $x'_1$. So the triple $(x'_1 - t + r, x'_2, r)$ contradicts the choice of $(x'_1, x'_2, t)$. □

**Bidirected graphs.** We consider finite graphs that may have loops and multiple edges. In order to define bidirections, it is convenient to let each edge be incident to two half-edges. Formally a graph $G$ is defined by three pairwise disjoint finite sets: a set of vertices $V$, a set of edges $E$, and a set of half-edges $H$. There is an incidence relation between $H$ and $V$, as well as between $H$ and $E$. These incidence relations are such that each half-edge is incident to precisely one vertex and one edge. Further, an edge is incident to precisely two half-edges. We denote by $\mathrm{h}v$ the set of the half-edges incident to a vertex $v$. The degree of $v$ is $\mathrm{d}(v) = |\mathrm{h}v|$.

A *biorientation*, or *bidirection*, over $G$ is a function $\epsilon : H \to \{-1, +1\}$. For $f \in \mathbf{Z}^E$ and $v \in V$, the *excess* of $f$ at $v$ is $\mathrm{ex}(f)_v = \sum(\epsilon(h)f(e) : h \in hv, e$ is the edge incident to $h)$, and the *excess* of $f$ is the vector $\mathrm{ex}(f) = (\mathrm{ex}(f)_v : v \in V)$. Given $c_1, c_2 \in \mathbf{Z}^E$, with $c_1 \leq c_2$, we denote by $[c_1, c_2]$ the set $\{f \in \mathbf{Z}^E : \ c_1 \leq f \leq c_2\}$.

PROPOSITION 3.4. *Let $c_1, c_2 \in \mathbf{Z}^E$, such that $c_1 \leq c_2$. Then $(V, \{\mathrm{ex}(f) : f \in [c_1, c_2]\})$ is a jump system.*

*Proof.* For $h \in H$ let $x(h) \in \mathbf{Z}^V$ be defined by $x(h)_v = \epsilon(h)$ if the vertex $v$ is incident to $h$, $x(h)_v = 0$ otherwise. For $e \in E$ let $\mathcal{F}_e = \{\lambda(x(h') + x(h'')) : \lambda \in [c_1(e), c_2(e)]\}$, where $h'$ and $h''$ are the half-edges of $G$ incident to $e$. We easily verify that $(V, \mathcal{F}_e)$ is a jump system. (One way is to observe that it is a hyperplane jump system intersected with a box and then extended by zeroes, but it is perhaps as easy to check directly.) We have $(V, \{\mathrm{ex}(f) : f \in [c_1, c_2]\}) = \sum((V, \mathcal{F}_e) : e \in E)$, where the summation stands for the sum operation considered in Theorem 3.3. The result follows from that theorem. □

The special case in which we take the bidirection to be trivial, that is, all the values of $\epsilon$ to be $+1$, is already quite interesting. If we also define $c_1(e) = 0$ and $c_2(e) = 1$ for each edge $e$, then $\{\mathrm{ex}(f) : f \in [c_1, c_2]\}$ is the set of degree sequences of subgraphs of $G$. If we now intersect this set with the unit cube, we get the matching delta-matroid of $G$. More general sets of this type are investigated in [7].

Suppose that we consider again the red-blue graph example of Proposition 2.8, but this time we allow the alternating paths to repeat vertices, but not edges. We show that we obtain another delta-matroid. We form a bidirected graph, by assigning to each red edge two positive half-edges, and assigning to each blue edge two negative half-edges; we define again $c_1(e) = 0$ and $c_2(e) = 1$ for each edge $e$. Now consider the resulting jump system, and reflect it in the coordinates corresponding to the vertices incident only to blue edges. Next, intersect it with the box determined by $0, u$, where $u_j = 1$ if $j$ is monochromatic, and $u_j = 0$ otherwise. Finally, project the jump system to the coordinates corresponding to the monochromatic vertices. The resulting jump system is a delta-matroid, and it is easy to see that it is precisely the desired one. Moreover, an oracle for this delta-matroid can be realized in polynomial time by methods of bidirected matching; see [12].

**Composition of jump systems.** Let $(S_0, \mathcal{F}_0)$ and $(S_1, \mathcal{F}_1)$ be two jump systems. The *composition* of $(S_0, \mathcal{F}_0)$ and $(S_1, \mathcal{F}_1)$ is the pair $(S, \mathcal{F})$, where $S = S_0 \Delta S_1$ and $\mathcal{F} \subseteq \mathbf{Z}^S$ is defined by $x \in \mathcal{F}$ if and only if there exists $x_0 \in \mathcal{F}_0$ and $x_1 \in \mathcal{F}_1$ satisfying $x_{0|S_0 \cap S_1} = x_{1|S_0 \cap S_1}$, $x_{|S_0 \setminus S_1} = x_{0|S_0 \setminus S_1}$, $x_{|S_1 \setminus S_0} = x_{1|S_1 \setminus S_0}$. (We may also speak of the composition $\mathcal{F}$ of $\mathcal{F}_0$ and $\mathcal{F}_1$.) Notice that this definition, in the case of $\{0, 1\}$-valued vectors, corresponds to the composition of delta-matroids introduced in §2. Hence the next result generalizes Theorem 2.1.

PROPOSITION 3.5. *The composition of two jump systems is a jump system.*

*Proof.* For $i = 1, 2$ we extend each vector in $\mathcal{F}_i$ to an element of $\mathbf{Z}^{S_0 \cup S_1}$ by filling it out with zeroes. Then we reflect $\mathcal{F}_1$ in the components corresponding to $S_0 \cap S_1$, then we take the sum, and then we take the minor associated with the vector $0 \in \mathbf{Z}^{S_0 \cap S_1}$. □

Conversely, Theorem 3.3 can be easily derived from the preceding proposition. (In fact, the original version of this paper proved the proposition directly and used it to prove Theorem 3.3.) Consider two sets $\mathcal{F}', \mathcal{F}'' \subseteq \mathbf{Z}^S$ that satisfy (2-SA). We first notice that $\Phi = \{(x, y, x+y) : x, y \in \mathbf{Z}\}$ is a subset of $\mathbf{Z}^3$, which satisfies (2-SA). (For example, it is an instance of the hyperplane systems introduced earlier.) Let us consider a family $(T_v = \{v', v'', v\} : v \in S)$ of pairwise disjoint 3-element sets. For each $v \in S$ let $\Phi_v = \{(x_{v'}, x_{v''}, x_v) : x_{v'}, x_{v''} \in \mathbf{Z}, x_v = x_{v'} + x_{v''}\} \subseteq \mathbf{Z}^{T_v}$, and consider the cartesian product $\Phi = \times(\Phi_v : v \in S) \subseteq \times(\mathbf{Z}^{T_v} : v \in S) = \mathbf{Z}^{S' \cup S'' \cup S}$, with $S' = \{v' : v \in S\}$ and $S'' = \{v'' : v \in S\}$. Then $\Phi$ satisfies (2-SA). We make a copy $\mathcal{G}' \subseteq \mathbf{Z}^{S'}$ of $\mathcal{F}'$ and a copy $\mathcal{G}'' \subseteq \mathbf{Z}^{S''}$ of $\mathcal{F}''$. The cartesian product $\mathcal{G} = \mathcal{G}' \times \mathcal{G}'' \subseteq \mathbf{Z}^{S' \cup S''}$ satisfies (2-SA). Finally, we notice that the composition of $\Phi \subseteq \mathbf{Z}^{S' \cup S'' \cup S}$ with $\mathcal{G} \subseteq \mathbf{Z}^{S' \cup S''}$ is equal to $\mathcal{F}' + \mathcal{F}''$.

**4. Bisubmodular polyhedra and jump systems.** Here we describe a generalization of (integral) polymatroids, called (integral) bisubmodular polyhedra. We show that the integral points of an integral bisubmodular polyhedron satisfy (2-SA). In the next section, we show a partial converse: The convex hull of a set satisfying (2-SA) is an integral bisubmodular polyhedron.

A function $f$ from pairs $(A, B)$ of disjoint subsets of $S$ to $\mathbf{R} \cup \{\infty\}$ is called *bisubmodular* if it satisfies, for all such pairs $(A, B)$, $(A', B')$,

$$f(A, B) + f(A', B') \geq f((A, B) \wedge (A', B')) + f((A, B) \vee (A', B')).$$

Here $(A, B) \wedge (A', B')$ denotes $(A \cap A', B \cap B')$, and we call it the *intersection* of $(A, B)$ and $(A', B')$; $(A, B) \vee (A', B')$ denotes $((A \cup A') \setminus (B \cup B'), (B \cup B') \setminus (A \cup A'))$, and we call it the *reduced union* of $(A, B)$ and $(A', B')$. (Notice that the operation $\vee$ is not associative.) It is convenient to assume throughout that $f(\emptyset, \emptyset) = 0$. The bisubmodular inequality (on real-valued functions) has been introduced by Kabadi and Chandrasekaran [5], [16], by Nakamura [18], [19], and by Qi [21].

The *bisubmodular polyhedron* associated with $f$ is $P(f) = \{x \in \mathbf{R}^S : x(A) - x(B) \leq f(A, B),\ A, B \subseteq S,\ A \cap B = \emptyset\}$. These polyhedra, again with the exception that the function values are finite, were introduced by Dunstan and Welsh [10] and studied in [16], [18], [21]. Nakamura showed the equivalence of the Dunstan–Welsh definition and the bisubmodular one. The function $f$ associated in §2 with a delta-matroid $(S, \mathcal{F})$ is bisubmodular and the associated bisubmodular polyhedron is the convex hull of the elements of $\mathcal{F}$. This result appears in [5] and [2]. We say that $f$ is *integral* if its finite values are integral, and that $P(f)$ is *integral* if $f$ is integral.

A number of more familiar classes of polyhedra fall into this class. If $f'$ is submodular on subsets of $S$, and $f'(\emptyset) = 0$, then $f$ defined by $f(A, \emptyset) = f'(A)$ for $A \subseteq S$ and $f(A, B) = \infty$ for $B \neq \emptyset$ is bisubmodular. The associated $P(f)$ is $\{x \in \mathbf{R}^S : x(A) \leq f'(A) \text{ for all } A \subseteq S\}$, the *submodular polyhedron* associated with $f'$. If we take $f(A, B) = f'(A)$ for all pairs $A, B$ of disjoint subsets of $S$, then it is easy to check that $f$ is bisubmodular if and only if $f'$ is also *monotone*: if $A_1 \subseteq A_2$, then $f'(A_1) \leq f'(A_2)$. In this case $P(f)$ is $\{x \in \mathbf{R}^S_+ : x(A) \leq f'(A) \text{ for all } A \subseteq S\}$, the *polymatroid* associated with $f'$. (Although $P(f')$ is a polymatroid even without the assumption of monotonicity, it is known that every polymatroid is determined by a monotone submodular function, so every polymatroid is a bisubmodular polyhedron.) Finally, the *base polyhedron* $\{x \in \mathbf{R}^S : x(A) \leq f'(A) \text{ for all } A \subseteq S,\ x(S) = f'(S)\}$ associated with $f'$ is obtained by taking $f(A, B) = f'(A) + f'(S \setminus B) - f'(S)$, and $f$ is bisubmodular.

Another more general class of bisubmodular polyhedra consists of Frank's generalized polymatroids. Here we suppose that $g$, $h$ are submodular functions on $S$, which are allowed to take the value $\infty$, and that they also satisfy

$$g(A) + h(B) \geq g(A \setminus B) + h(B \setminus A)$$

for all pairs of subsets $A, B$ of $S$. Then $Q(g,h) = \{x \in \mathbf{R}^S : -h(A) \leq x(A) \leq g(A)$ for all $A \subseteq S\}$ is the *generalized polymatroid* determined by $g$ and $h$. If we define $f(A,B)$ to be $g(A)+h(B)$ for disjoint pairs $A, B$ of subsets of $S$, then $P(f) = Q(g,h)$, and $f$ is bisubmodular. The class of generalized polymatroids contains all the special classes mentioned earlier, but the class of bisubmodular polyhedra is even larger.

PROPOSITION 4.1. *Let $f$ be bisubmodular and let $C \subseteq S$. Then reflecting $P(f)$ in $C$ gives a bisubmodular polyhedron $P(f')$, where $f'$ is defined by*

$$f'(A,B) = f((A \setminus C) \cup (B \cap C), (B \setminus C) \cup (A \cap C)).$$

*Proof.* It is clear that $x' \in P(f)$ if and only if $x'(A') - x'(B') \leq f(A', B')$ for all pairs $(A', B')$, where $x'$ is obtained by reflecting $x$ in $C$. But this is equivalent to

$$x(A' \setminus C) - x(A' \cap C) - x(B' \setminus C) + x(B' \cap C) \leq f(A', B')$$

or, taking $A = (A' \setminus C) \cup (B' \cap C)$, $B = (B' \setminus C) \cup (A' \cap C)$,

$$x(A) - x(B) \leq f((A \setminus C) \cup (B \cap C), (B \setminus C) \cup (A \cap C)).$$

Hence, the reflection of $P(f)$ is $P(f')$, and it remains only to prove that $f'$ is bisubmodular. This can be done by a straightforward computation, which we omit. □

PROPOSITION 4.2. *Let $f$ be bisubmodular and $S' \subseteq S$. Then the projection of $P(f)$ onto the coordinates indexed by $S'$ is a bisubmodular polyhedron $P(f')$, where $f'$ is the restriction of $f$ to $S'$.*

*Proof.* It is obvious that $f'$ is bisubmodular, so we need only show that $P(f')$ is the projection. It is enough to prove this in the case in which $S' = S \setminus \{e\}$ for some $e \in S$. By Fourier elimination of $x_e$, the projection is determined by two classes of inequalities. The first consists of inequalities

(i) $$x(A) - x(B) \leq f(A,B), \quad \text{where } e \notin A \cup B.$$

The second consists of inequalities that are all obtained by adding an inequality for $P(f)$ in which $x_e$ has coefficient 1, to one in which $x_e$ has coefficient $-1$. So each such inequality has the form

(ii) $$x(A') - x(B') + x(A'') - x(B'') \leq f(A', B') + f(A'', B''), \quad \text{where } e \in A' \cap B''.$$

We need to show that each inequality of type (ii) is implied by those of type (i). In fact, we add the inequality for $(A', B') \wedge (A'', B'')$ to the one for $(A', B') \vee (A'', B'')$. These inequalities are both of type (i). Their sum has the same left-hand side as (ii) and its right-hand side is no larger than the right-hand side of (ii), by bisubmodularity. □

We remark that it follows that every bisubmodular polyhedron is non-empty: since $f(\emptyset, \emptyset) = 0$, this is true by induction. Besides projection, several other operations that preserve (2-SA) also preserve bisubmodular polyhedra, and the corresponding bisubmodular function can be explicitly constructed. For cartesian product, translation, and minors this is easy to show, and we do not bother to state the results. On the other hand, for intersection with a box, it is not obvious, and the formula for the defining function is not easy to establish. This result will be discussed elsewhere.

The proof that integral bisubmodular functions yield jump systems uses a basic lemma, the analogue for bisubmodular polyhedra of Lemma 3.2. Given $x \in P(f)$, we say that a pair $(A, B)$ is *$x$-tight* (or just *tight*) if $x(A) - x(B) = f(A, B)$.

LEMMA 4.3. *The intersection and the reduced union of $x$-tight pairs are $x$-tight.*

*Proof.* Suppose that $(A, B)$ and $(A', B')$ are $x$-tight.

$$\begin{aligned} x((A \cup A') \setminus (B \cup B')) - x((B \cup B') \setminus (A \cup A')) + x(A \cap A') - x(B \cap B') \\ = x(A) - x(B) + x(A') - x(B') \\ = f(A, B) + f(A', B') \\ \geq f((A, B) \wedge (A', B')) + f((A, B) \vee (A', B')). \end{aligned}$$

Since $x \in P(f)$, the inequality also holds in the other direction, so we have equality throughout. □

We remark that the above lemma, or similar arguments as in the proof, can be used to obtain further results. For example, if $(A, B)$ and $(A', B')$ are $x$-tight, then so is $(A \setminus B', B \setminus A')$, by applying the lemma again to $(A, B)$ and $(A, B) \vee (A', B')$. This was pointed out in [16]. Also if $f$ and $x$ are integral, $(A, B)$ is $x$-tight, and $x(A') - x(B') = f(A', B') - 1$, then one of the intersection and reduced union is $x$-tight.

THEOREM 4.4. *Let $f$ be bisubmodular and integral. Then $\mathcal{F} = \mathbf{Z}^S \cap P(f)$ satisfies* (2-SA).

*Proof.* Let $x, y \in \mathcal{F}$ and $s \in \mathrm{St}(x, y)$ and suppose (2-SA) fails. By reflecting $P(f)$ in $\{j : x_j > y_j\}$, we may assume that $x \leq y$. Suppose that $s = \{e\}$. Let $Q = \{j \in S \setminus \{e\} : x_j < y_j\}$, and let $Q' = \{j \in Q :$ there exists $x$-tight $(A, B)$ with $e \in A$, $j \notin B\}$.

*Claim.* There exists $x$-tight $(A, B)$ with $e \in A$ and $Q \setminus B = Q'$.

Suppose first that $Q' = \emptyset$. Choose $x$-tight $(A', B')$ with $e \in A'$. (Such exists, because $x + \{e\} \notin P(f)$.) Then by the definition of $Q'$, $Q \setminus B' = \emptyset = Q'$, as required. So suppose that $Q' \neq \emptyset$. Now take the intersection of all the $x$-tight pairs $(A'', B'')$ with $e \in A''$ and $Q \setminus B'' \neq \emptyset$. This gives an $x$-tight pair $(A, B)$ with $B \cap Q' = \emptyset$ and so, by the definition of $Q'$, $Q \setminus B = Q'$. This completes the proof of the claim.

Suppose there exists $j \in Q \setminus Q'$. Notice that $j \in B$. Since $x + \{e\} + \{j\} \notin P(f)$ and $j \notin Q'$, there exists $(A', B')$ such that

(i) $$x(A') - x(B') = f(A', B'),\ j \in A',\ e \notin B';$$

or

(ii) $$x(A') - x(B') \geq f(A', B') - 1,\ j, e \in A'.$$

In case (i) the reduced union of $(A, B)$ and $(A', B')$ is an $x$-tight pair $(A'', B'')$ with $e \in A''$, $j \notin B''$, so $j \in Q'$, a contradiction. In case (ii) both the reduced union and the intersection of $(A, B)$, $(A', B')$ are pairs $(A'', B'')$ with $e \in A''$, $j \notin B''$. Moreover, at least one of the two pairs is $x$-tight, so $j \in Q'$, a contradiction. It follows that $Q \setminus Q' = \emptyset$, so $y(A) - y(B) > x(A) - x(B) = f(A, B)$, again a contradiction. □

We use Lemma 4.3 to prove another basic fact about bisubmodular polyhedra, that each such polyhedron has a unique defining function. For the case where all function values are finite, this result is proved in [14, p. 94].

THEOREM 4.5. *If $f$ is bisubmodular on pairs of disjoint subsets of $S$, then for each such pair $A, B$ we have $f(A,B) = \max_{x \in P(f)}(x(A) - x(B))$. Moreover, if $f$ is integral, then the maximum is achieved by an integral $x$.*

*Proof.* By applying reflection and projection, using Propositions 4.1 and 4.2, we can assume that $A = S$ and $B = \emptyset$. Obviously, the maximum is at most $f(S, \emptyset)$, so if it is $\infty$, we are done. We choose $\bar{x} \in P(f)$ maximizing $x(A) - x(B)$. Fix $e \in S$. Since $\bar{x}_e$ cannot be increased, there is a tight pair $(A', B')$ with $e \in A'$.

*Claim.* For each $j \notin A'$ there is a tight pair $(A_j, B_j)$ such that $e \in A_j$ and $j \notin B_j$.

*Proof of claim.* If not, then there is a tight pair $(A'', B'')$ with $j \in A''$ and $e \notin B''$. (Otherwise we could increase both $\bar{x}_e$ and $\bar{x}_j$). Now take $(A_j, B_j) = (A', B') \vee (A'', B'')$. This pair is tight by Lemma 4.3, and it is easy to see that it satisfies the conditions of the claim.

Now take the intersection over all $j \notin A'$ of the pairs $(A_j, B_j)$. We get a tight pair $(A_e, B_e)$ with $e \in A_e$ and $B_e \subset A'$. The intersection of $(A_e, B_e)$ with $(A', B')$ is a tight pair $(A'_e, \emptyset)$ with $e \in A'_e$. Finally, the union over all $e$ of these tight pairs is the tight pair $(S, \emptyset)$, so $\bar{x}(S) = f(S, \emptyset)$, as required. It is straightforward to check that if $f$ is integral, then the whole argument applies to integral points, so the second part is proved also. □

**5. Jump systems and bisubmodular polyhedra.** We say that $x \in \mathcal{F}$ is *$(A, B)$-maximal* in $\mathcal{F}$ if $y \in \mathcal{F}$, $y_j \geq x_j$ for all $j \in A$, $y_j \leq x_j$ for all $j \in B$ imply $y_{|A \cup B} = x_{|A \cup B}$.

LEMMA 5.1. *If $\mathcal{F}$ satisfies* (2-SA) *and $y, x \in \mathcal{F}$ with $y(A) - y(B) > x(A) - x(B)$, then $x$ is not $(A, B)$-maximal.*

*Proof.* By twisting at B, we may assume that $B = \emptyset$. Suppose that $x$ is $(A, \emptyset)$-maximal and there exists $y \in \mathcal{F}$ and $y(A) > x(A)$. Subject to this, choose $y$ so that $\sum_{j \in A} |x_j - y_j|$ is minimum. By the maximality of $x$, there exists $e \in A$ with $x_e > y_e$. Either $y' = y + \{e\} \in \mathcal{F}$, or $y'' = y + \{e\} + s \in \mathcal{F}$ for some step $s$ from $y + \{e\}$ to $x$. But this contradicts the choice of $y$. □

Given $\mathcal{F}$, define $f$ by $f(A, B) = \max_{x \in \mathcal{F}}(x(A) - x(B))$ if the maximum exists, and to be $\infty$ otherwise.

LEMMA 5.2. *If $\mathcal{F}$ satisfies* (2-SA), *then $f$ is bisubmodular.*

*Proof.* Let $(A, B)$, $(A', B')$ be pairs of disjoint sets. First, we consider the case where the right-hand side of the bisubmodular inequality is not $\infty$. We may assume that $f(A, B) \neq \infty$, since otherwise the inequality holds trivially. Choose $x \in \mathcal{F}$ to be $(A \cap A', B \cap B')$-maximal. Now by increasing $x_j$ for $j \in (B \Delta B') \setminus (A \cup A')$, we can find an $x'$ that is $(A \vee A', B \vee B')$-maximal. Then by Lemma 5.1, we get

$$\begin{aligned}
&f(A,B) + f(A',B') \\
&\qquad \geq x'(A) - x'(B) + x'(A') - x'(B') \\
&\qquad = x(A \cap A') - x(B \cap B') + x'((A \cup A') \setminus (B \cup B')) - x'((B \cup B') \setminus (A \cup A')) \\
&\qquad = f((A,B) \wedge (A',B')) + f((A,B) \vee (A',B')),
\end{aligned}$$

as required.

Now suppose that the right-hand side of the bisubmodular inequality is infinity. Let $\mathcal{F}^k$ be $\mathcal{F} \cap \{x \in \mathbf{Z}^S : -k \leq x_j \leq k \text{ for all } j \in S\}$. Then $\mathcal{F}^k$ satisfies (2-SA); let $f^k$ be the corresponding (bisubmodular!) function. Then the right-hand side of the bisubmodular inequality for $f^k$ goes to $\infty$ with $k$, and so the left-hand side must also, and we are finished. □

THEOREM 5.3. *If $\mathcal{F}$ satisfies* (2-SA), *then* $\operatorname{conv}(\mathcal{F})$ *is an integral bisubmodular polyhedron.*

*Proof.* Define $f$ by $f(A,B) = \max_{x\in\mathcal{F}}(x(A) - x(B))$. Then $f$ is integral and bisubmodular, by Lemma 5.2, and $\mathcal{F} \subseteq P(f)$. If $P(f) \neq \operatorname{conv}(\mathcal{F})$, then there exists $c \in \mathbf{R}^S$ and $y \in P(f)$ such that $cy > cx$ for every $x \in \mathcal{F}$. By a straightforward perturbation argument we can choose $c$ so that there does not exist $j \in S$ with $c_j = 0$ and there do not exist distinct elements $j$, $k$ of $S$ with $|c_j| = |c_k|$. By reflection in $N = \{j : c_j < 0\}$, we may assume that $c_j > 0$ for all $j \in S$. Now $\max_{x\in\mathcal{F}} cx$ exists, so $\max_{x\in\mathcal{F}} x(S)$ exists, by Lemma 5.1 with $A = S$, $B = \emptyset$. Therefore, we can form the polyhedron $B(f) = \{x \in P(f) : x(S) = f(S,\emptyset)\}$. The set $\mathcal{B}$ of maximal members of $\mathcal{F}$ is contained in $B(f)$. Since $c_j > 0$, $j \in S$, there exists $y \in B(f)$ with $cy > cx$ for every $x \in \mathcal{B}$. This again follows from Lemma 5.1. We need the following claim.

*Claim.* If $y, x \in \mathcal{B}$, $A \subseteq S$, and $y(A) > x(A)$, then $x$ is not $A$-maximal over $\mathcal{B}$.

*Proof of Claim.* If possible, choose $y$ and $x$ violating the statement with $\sum_{j\in A}|x_j - y_j|$ as small as possible. Clearly there exists $e \in A$ with $x_e > y_e$. Apply (2-SA) to get $j \in S$ with $y + \{e\}$ or $y + \{e\} + \{j\}$ or $y + \{e\} - \{j\} \in \mathcal{F}$. By the definition of $\mathcal{B}$, the only possibility is the last one. But then $y'' = y + \{e\} - \{j\} \in \mathcal{B}$, and this contradicts the choice of $y$, and the claim is proved.

Let us relabel the elements of $S$ as $e_1, e_2, \ldots, e_n$ so that $c_{e_1} > c_{e_2} > \cdots > c_{e_n}$, and let $T_i$ denote $\{e_1, e_2, \ldots, e_i\}$ for $0 \le i \le n$. For $T \subseteq S$, $z \in \mathbf{R}^T$, and $Q \subseteq \mathbf{R}^S$ we write $z \hat{\in} Q$ to mean that there exists $\bar{z} \in Q$ such that $\bar{z}_{|T} = z$. Choose $\bar{x} \in \mathcal{B}$ as follows:

```
For $i = 1$ to $n$
    Choose $\bar{x}_{e_i}$ to be $\max(\alpha : (\bar{x}_{e_1}, \ldots, \bar{x}_{e_{i-1}}, \alpha) \hat{\in} \mathcal{B}$, if the maximum exists.
    Otherwise, stop.
```

Now suppose the procedure runs to completion and delivers $\bar{x} \in \mathcal{B}$. By the claim, $\bar{x}(T_i) = f(T_i,\emptyset) < \infty$, $1 \le i \le n$. Now for any $x \in B(f)$, we have

$$\begin{aligned} cx &= \sum_{i=1}^{n} c_{e_i}(x(T_i) - x(T_{i-1})) \\ &= \sum_{i=1}^{n-1}(c_{e_{i+1}} - c_{e_i})x(T_i) + c_{e_n}x(T_n) \\ &\le \sum_{i=1}^{n-1}(c_{e_{i+1}} - c_{e_i})\bar{x}(T_i) + c_{e_n}\bar{x}(T_n) = c\bar{x}. \end{aligned}$$

This shows that a point of $\mathcal{B}$ maximizes $cx$ over $B(f)$, a contradiction. Now suppose that the procedure stops early; say that $j$ is the first index for which the maximum does not exist and $\bar{x}$ is the point constructed after step $j-1$. Then by the claim, we have $\bar{x}(T_i) = f(T_i,\emptyset)$ for $1 \le i \le j-1$. Moreover, $\bar{x}$ is not $A$-maximal in $\mathcal{B}$, so there exists $x' \in \mathcal{B}$ with $x'_{e_i} = \bar{x}_{e_i}$, $1 \le i \le j-1$ and $x'_j = \bar{x}_j + \alpha$ for some positive integer $\alpha$. But then, $x'(S \setminus T_j) = \bar{x}(S \setminus T_j) - \alpha$, so $cx' \ge c\bar{x} + \alpha\epsilon$, where $\epsilon = \min(c_{e_i} - c_{e_{i+1}} : 1 \le i \le n-1)$. But the same argument can be applied to $x'$, and so on, so $cx$ is unbounded on $\mathcal{B}$, a contradiction. □

We remark that the proof of Theorem 5.3 contains the basis of a greedy algorithm for optimizing a linear function over a set satisfying (2-SA). However, we have managed to avoid many of the awkward parts of such an algorithm (such as those dealing with unboundedness and equal cost coefficients). These difficulties are handled for some classes of polyhedra in [13]–[15].

It is a consequence of Theorem 5.3 that the convex hull of a set satisfying (2-SA) is given by inequalities having coefficients $0, 1, -1$. This result can be applied to the bidirected-graph example of §3 to conclude that the resulting polyhedra can be described in this way. For the case of trivial bidirections, these results, and somewhat more, were proved in [7].

We can also prove that an integral bisubmodular polyhedron, that is, the polyhedron determined by a bisubmodular polyhedron that is integral, is indeed an integral polyhedron. This result, in a slightly less general setting, appears in [16], [18], and [21].

COROLLARY 5.4. *Every integral bisubmodular polyhedron is the convex hull of its integral points.*

*Proof.* Let $f$ be an integral bisubmodular function defined on pairs of disjoint subsets of $S$. By Theorem 4.4, $\mathcal{F} = P(f) \cap \mathbf{Z}^S$ satisfies (2-SA), and so by Theorem 5.3, $\text{conv}(\mathcal{F})$ is a bisubmodular polyhedron $P(f')$, where $f'$ is defined by $f'(A, B) = \max_{x \in \mathcal{F}}(x(A) - x(B))$. Now by Theorem 4.5, we have that $f = f'$, and we are done. □

A *gap* of a set $\mathcal{F} \subseteq \mathbf{Z}^S$ is an integral point in $\text{conv}(\mathcal{F}) \setminus \mathcal{F}$. The examples in Fig. 1 show that adding a gap to a set satisfying (2-SA) can create a set violating (2-SA). On the other hand, we have the following result.

COROLLARY 5.5. *Suppose that $\mathcal{F}$ satisfies* (2-SA) *and $\mathcal{F}'$ is obtained from $\mathcal{F}$ by adding all of the gaps of $\mathcal{F}$. Then $\mathcal{F}'$ satisfies* (2-SA).

*Proof.* By Theorem 5.3, $\text{conv}(\mathcal{F})$ is an integral bisubmodular polyhedron $P$. By Theorem 4.4, the integral points in $P$ satisfy (2-SA). □

The following related results have been obtained recently. Duchamp [21] has proved that the "delta-sum" $(S, \mathcal{F})$, of delta-matroids $(S, \mathcal{F}_0)$ and $(S, \mathcal{F}_1)$, is again a delta-matroid. Here $\mathcal{F} = \{F_0 \Delta F_1 : F_0 \in \mathcal{F}_0, F_1 \in \mathcal{F}_1\}$. This result implies the composition theorem for delta-matroids. Payan [20] has proved that the mod 2 reduction of a jump system is a delta-matroid. (That is, each component of a vector in $\mathcal{F}$ is replaced by 0 or 1 according to its parity.) This result, together with Theorem 3.3 implies the result of Duchamp. Although adding an arbitrary gap can violate (2-SA), there is a notion intermediate between this and adding all gaps. If $j \in S$, a *gap in direction $j$* is a point $x \notin \mathcal{F}$ such that $x + \{j\}$ and $x - \{j\}$ are both in $\mathcal{F}$. We originally conjectured that adding all gaps in the same direction preserves (2-SA). Payan [20] has proved this conjecture. This result implies Corollary 5.5.

**Acknowledgment.** We are grateful to András Sebő for his help and encouragement.

## REFERENCES

[1] R. E. BIXBY AND W. H. CUNNINGHAM, *Matroid optimization and algorithms*, in M. Grötschel, and L. Lovász, eds., Handbook of Combinatorics, R. L. Graham, North-Holland, Amsterdam, to appear.

[2] A. BOUCHET, *Greedy algorithm and symmetric matroids*, Math. Programming, **38** (1987), pp. 147–159.

[3] ———, *Representability of Δ-matroids*, Colloq. Soc. Janos Bolyai, **52** (1988), pp. 167–182.

[4] ———, *Matchings and Δ-matroids*, Discrete Appl. Math., **24** (1989), pp. 55–62.

[5] R. CHANDRASEKARAN AND S. N. KABADI, *Pseudomatroids*, Discrete Math., **71** (1988), pp. 205–217.

[6] W. H. Cunningham, *A Combinatorial Decomposition Theory*, Thesis, University of Waterloo, 1973.

[7] W. H. Cunningham and J. Green-Krótki, *b-matching degree-sequence polyhedra*, Combinatorica, **11** (1991), pp. 219-230.

[8] A. Dress and T. Havel, *Some combinatorial properties of discriminants in metric vector spaces*, Adv. Math., **62** (1986), pp. 285–312.

[9] A. Duchamp, Tech. report, Université du Maine, France, 1993.

[10] F. D. J. Dunstan and D. J. A. Welsh, *A greedy algorithm solving a certain class of linear programmes*, Math. Programming, **5** (1973), pp. 338–353.

[11] J. Edmonds, *Submodular functions, matroids, and certain polyhedra*, in Combinatorial Structures and their Applications, R. K. Guy, H. Hanani, N. Sauer, and J. Schönheim, eds., Gordon and Breach, New York, 1970, pp. 69-87.

[12] J. Edmonds and E. L. Johnson, *Matching: A well-solved class of integer linear programs*, in Combinatorial Structures and their Applications, R. K. Guy, H. Hanani, N. Sauer, and J. Schönheim, eds., Gordon and Breach, New York, 1970, pp. 88-92.

[13] A. Frank and E. Tardos, *Generalized polymatroids and submodular flows*, Math. Programming, **42** (1988), pp. 489–563.

[14] S. Fujishige, *Submodular Functions and Optimization*, North-Holland, Amsterdam, 1991.

[15] S. Fujishige and N. Tomizawa, *A note on submodular functions on distributive lattices*, J. Oper. Res. Soc. Japan, **38** (1988), pp. 155–167.

[16] S. N. Kabadi and R. Chandrasekaran, *On totally dual integral systems*, Discrete Appl. Math., **26** (1990), pp. 87–104.

[17] L. Lovász, *Matroid matching and some applications*, J. Combin. Theory Series B, **28** (1980), pp. 208–236.

[18] M. Nakamura, *A characterization of greedy sets: Universal polymatroids* (I), Sci. Papers College Arts Sci. Univ. Tokyo, **38** (1988), pp. 155–167.

[19] M. Nakamura, *A characterization of those polytopes in which the greedy algorithm works*, abstract, 13th International Symposium on Mathematical Programming, Tokyo, Japan, 1988.

[20] C. Payan, to appear.

[21] Liqun Qi, *Directed submodularity, ditroids and directed submodular flows*, Math. Programming, **42** (1988), pp. 579–599.

[22] W. R. Richardson, *Connectivity, Decomposition, and Complexity in Matroids*, Thesis, University of Waterloo, 1973.

[23] D. J. A. Welsh, *Matroid Theory*, Academic Press, New York, 1976.

SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 33–50, February 1995

# DUAL EULERIAN PROPERTIES OF PLANE MULTIGRAPHS*

BRADLEY S. CARLSON†, C.Y. ROGER CHEN‡, AND DIKRAN S. MELIKSETIAN§

**Abstract.** A plane multigraph is said to be dual Eulerian if both it and its dual contain an Euler path or circuit and the Euler paths have corresponding edge sequences. In this paper several properties of plane multigraphs are derived, and a necessary and sufficient condition for a plane multigraph to be dual Eulerian is given. Although the necessary and sufficient condition for a multigraph to be Eulerian is somewhat trivial, the necessary and sufficient condition for a plane multigraph to be dual Eulerian is not. Nevertheless, the question of whether or not a plane multigraph is dual Eulerian can be answered in time proportional to a linear function of the number of edges of the graph, and an algorithm that answers this question is presented in this paper. This theory can be applied to the layout synthesis of functional cells for Complementary Metal-Oxide Semiconductor Very Large-Scale Integrated circuits.

**Key words.** Eulerian graphs, planar graphs, VLSI layout

**AMS subject classifications.** 05C45, 68R10, 94C15

## 1. Introduction.

The problem of identifying plane multigraphs that are dual Eulerian is investigated in this paper. A *dual Eulerian plane multigraph* is one in which both itself and its dual contain a Euler path or circuit, and the Euler paths have corresponding edge sequences. The problem is restricted to planar multigraphs, since these are the only graphs for which a dual is defined. Furthermore, it is assumed that a specific embedding of a planar graph, called a *plane graph*, is given [3]. The problem of whether or not a planar graph admits an embedding that is dual Eulerian is a different problem and is not addressed in this paper. For example, the graphs shown in Figs. 1(a) and (b) are different plane embeddings of the same planar graph; and the graph in Fig. 1(b) is dual Eulerian but the graph in Fig. 1(a) is not.

This problem is of interest in the design of Complementary Metal-Oxide Semiconductor (CMOS) Very Large-Scale Integrated (VLSI) circuits [8]. In VLSI design it is desirable to design the physical implementations of circuits such that they require a minimum amount of silicon area. When circuits are represented by undirected multigraphs, dual paths correspond to linear placements of transistors that require a minimum amount of silicon area [8].

In general, the related VLSI layout problem requires one to find a minimum number of disjoint dual Euler paths that cover the entire graph. This is exactly the problem addressed in [8]; however, they restrict the problem to series/parallel graphs. Uehara and VanCleemput [8] propose a heuristic technique for the solution of this problem. Maziasz and Hayes [5] also address the problem for series/parallel graphs and show that an exact polynomial-time solution exists. Nair et al. [6] present an exact polynomial-time method for series/parallel graphs as well. Wimer et al. [10] present a technique to solve this problem for arbitrary circuit topologies; i.e., it is not necessary that the graph representations be dual to one another. Hence, their

* Received by the editors October 8, 1992; accepted for publication (in revised form) December 27, 1993. This research was supported in part by National Science Foundation grant MIP-8909792.

† Department of Electrical Engineering, State University of New York at Stony Brook, Stony Brook, New York 11794-2350 (bcarlson@sbee.sunysb.edu).

‡ Department of Electrical and Computer Engineering, Syracuse University, Syracuse, New York 13244-1240.

§ Department of Electrical Engineering, South Dakota School of Mines and Technology, Rapid City, South Dakota 57701.

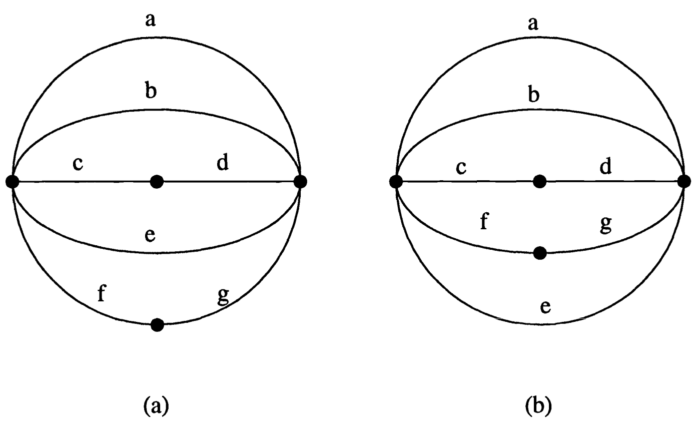


FIG. 1. *Two plane embeddings of a planar graph that are* (a) *not dual Eulerian and* (b) *dual Eulerian.*

method is difficult to formalize mathematically, and is heuristic in nature. A parallel algorithm for this problem on series/parallel graphs is presented in [4].

We address the problem for a more general class of graphs than [8], [5], [6], [4] but for a less general class of circuit topologies than [10]. We cannot handle arbitrary circuit topologies, since our technique is very formal; nevertheless, we solve the problem for the most general class of plane graphs (i.e., biconnected graphs) whose topological duals satisfy the graph definition used in this paper. The general problem (i.e., determining a minimum number of disjoint dual Euler paths) for arbitrary plane graphs is NP-hard [9]. We show that the problem of determining the existence of a dual Euler path is polynomial-time solvable. Moreover, we give an algorithm for computing a dual Euler path when one exists. Our method can easily be extended to solve the VLSI layout problem either heuristically or exactly using branch-and-bound method [1].

This paper is organized as follows. The necessary mathematical definitions and some preliminary results are presented in the next section. The derivation of the algorithm and the main results are presented in §3. The algorithm is specified in pseudocode and analyzed in §4. An example illustrating the application of the algorithm is presented in §5, and the paper is concluded in §6.

**2. Definitions and preliminary results.** $\Gamma(V, f, E)$ is an undirected multigraph with a set $V$ of vertices, a set $E$ of edges, and a function $f : E \to P_2(V)$, where $P_2(V)$ is a set that contains the subsets of $V$ of size two [2]. Furthermore, the edges are labeled for convenience in representation and presentation. A *path* $p = v_0e_0v_1e_1v_2\dots v_{n-1}e_{n-1}v_n$ in $\Gamma$ is a sequence of alternating vertices and edges that begins and ends at a vertex, $f(e_i) = \{v_i, v_{i+1}\}$ $(0 \le i \le n-1)$ and $e_i \ne e_j$ for $i \ne j$. The *length* of a path is the number of edges contained in it. A path is a *circuit* if $v_0 = v_n$. A multigraph $\Gamma$ is said to be *Eulerian* if and only if it contains a path that contains every edge of $\Gamma$. A circuit is *elementary* if each vertex is distinct with the exception of the first coinciding with the last. The edge set of a circuit in $\Gamma$ is a *cycle* [2]. An *elementary cycle* is the cycle of an elementary circuit. $\Gamma$ is said to be

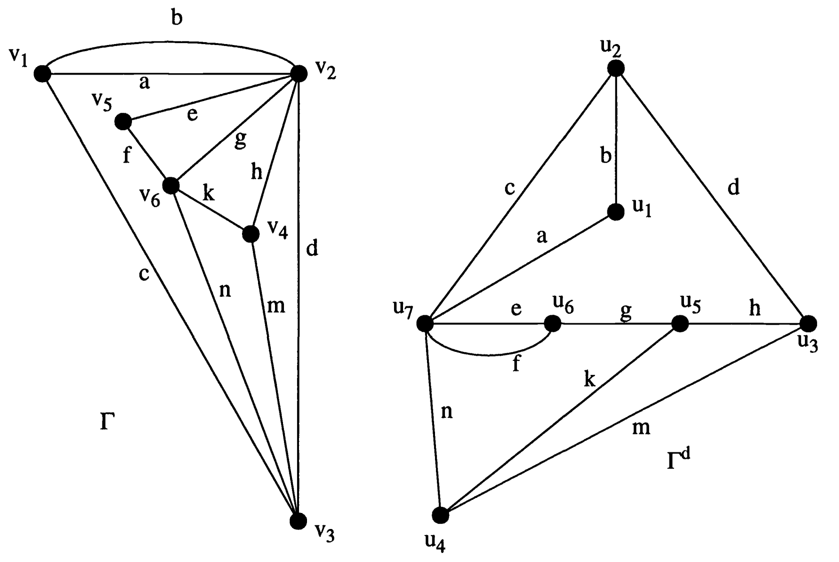


FIG. 2. *A plane undirected multigraph* $\Gamma$ *and its dual* $\Gamma^d$.

*planar* if it can be drawn on the plane with edges meeting only at the vertices [2]. An embedding of a planar multigraph on the plane is called a *plane* multigraph [3]. A *face* of a plane graph $\Gamma$ is a domain of the plane surrounded by edges of $\Gamma$ such that any two points in it can be joined by a curve not crossing any edge [7]. A *facial circuit* is a circuit that forms the boundary of a face in $\Gamma$. A *facial cycle* is the cycle of a facial circuit. The dual multigraph $\Gamma^d$ of a plane multigraph $\Gamma$ can be constructed by placing a vertex in each face of $\Gamma$ and connecting two vertices, say $v_1$ and $v_2$, in $\Gamma^d$ by an edge labeled $e$ if and only if the edge labeled $e$ in $\Gamma$ is on the boundary of the faces of $\Gamma$ corresponding to $v_1$ and $v_2$ of $\Gamma^d$. The edge sets of $\Gamma$ and $\Gamma^d$ are the same, and the vertices of $\Gamma^d$ correspond to the faces of $\Gamma$ and vice versa. A plane multigraph $\Gamma$ and its dual $\Gamma^d$ are shown in Fig. 2.

Henceforth it is assumed that $\Gamma$ is a plane undirected multigraph. We restrict ourselves to plane multigraphs, since their duals are unique. A path in $\Gamma$ is said to be a *dual path* if there exists a path in $\Gamma^d$ with the same edge sequence. For example, $v_2\,d\,v_3\,m\,v_4\,k\,v_6\,g\,v_2\,e\,v_5\,f\,v_6$ is a path in $\Gamma$ of Fig. 2, and $u_2\,d\,u_3\,m\,u_4\,k\,u_5\,g\,u_6\,e\,u_7\,f\,u_6$ is a path in $\Gamma^d$. Since these paths have corresponding edge sequences, each one is a dual path. A dual path is a *dual circuit* if it is a circuit in both $\Gamma$ and $\Gamma^d$. A path can be identified by its sequence of edges, and this alternative means of identification is used throughout the sequel whenever it is unambiguous or the ambiguity is unimportant. In $\Gamma$ of Fig. 2 $a\,e\,f\,g$ is a path, but it is not a dual path since $a\,e\,f\,g$ is not a path in $\Gamma^d$. Similarly, $e\,a\,b\,c$ is a path in $\Gamma^d$ but is not a dual path since it is not a path in $\Gamma$. $\Gamma$ is said to be *dual Eulerian* if and only if it contains a dual path that contains every edge. Of course if $\Gamma$ is dual Eulerian, then so is $\Gamma^d$. An obvious necessary condition for

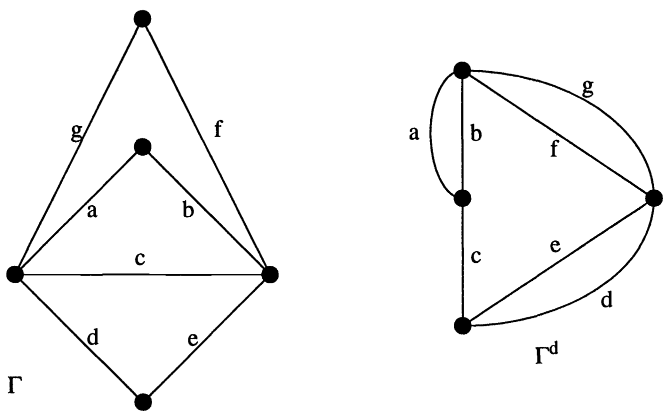


FIG. 3. *Example illustrating that both* Γ *and* $\Gamma^d$ *being Eulerian is not a sufficient condition for them to be dual Eulerian.*

Γ to be dual Eulerian is that both Γ and $\Gamma^d$ be Eulerian.[1] A simple counterexample shown in Fig. 3 illustrates that this is not a sufficient condition.

A *submultigraph* of a plane undirected multigraph $\Gamma(V, f, E)$ is a plane undirected multigraph $\Gamma'(V', f_{|E'}, E')$, where $V' \subseteq V$, $E' \subseteq E$, and $f_{|E'}$ is the function $f$ restricted to the set $E'$. A submultigraph $\Gamma'(V', f_{|E'}, E')$ of $\Gamma(V, f, E)$ is a *component* of Γ if there does not exist $v \in V'$ and $e \in E \setminus E'$ such that $v \in f(e)$. A *cocycle* in Γ is a set of edges of Γ such that the removal of these edges from Γ increases the number of components of Γ. The set of edges incident on a vertex is a cocycle and is referred to as a *vertex cocycle.* If the vertex cocycle of $v \in V$ is denoted by $f^*(v)$, then

$$f^*(v) = \{e \in E | v \in f(e)\} \quad [2].$$

An *isthmus* is an edge that when removed from Γ increases the number of components of Γ (i.e., it is a cocycle of size one). It is assumed that Γ does not contain any isthmuses, since if it did, its dual would not satisfy the definition of a graph [2]. That is, the dual of a graph with isthmuses contains loops (i.e., an edge that is incident on the same vertex at both of its ends), and loops contradict the definition of the function $f : E \to P_2(V)$.

Two edges $e_i$ and $e_j$ are said to be in *series* if they are incident on the same vertex of degree two or they are connected by a sequence of series connected edges. Two edges $e_i$ and $e_j$ are said to be in *parallel* if $f(e_i) = f(e_j)$.

LEMMA 2.1. *The series and parallel relations are equivalence relations.*

[1] A well-known necessary and sufficient condition for a multigraph to be Eulerian is that it have no more than two vertices with an odd degree [2].

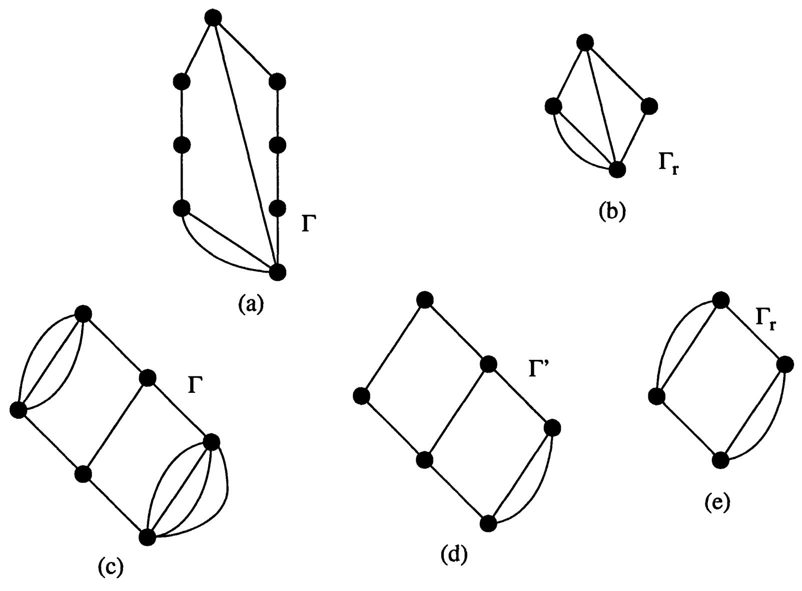


FIG. 4. *Examples illustrating graph reduction.*

Let the equivalence classes induced by the series and parallel relations be known as the series and parallel sets of edges, respectively. A series (respectively, parallel) set of edges in $\Gamma$ corresponds to a parallel (respectively, series) set of edges in $\Gamma^d$, and vice versa. Define the *series* (respectively, *parallel*) *reduction operation* on a graph $\Gamma$ as the operation of replacing an odd series (respectively, parallel) set of edges in $\Gamma$ by one edge. Define the *reduced graph* $\Gamma_r$ of a graph $\Gamma$ as the graph obtained by recursively applying the reduction operations to $\Gamma$ until they can no longer be applied. For example, the reduced graph $\Gamma_r$ in Fig. 4(b) is obtained from $\Gamma$ in Fig. 4(a) by applying the series reduction operation, and the graph $\Gamma'$ in Fig. 4(d) is obtained from $\Gamma$ in Fig. 4(c) by applying the parallel reduction operation. The series reduction operation can be applied to $\Gamma'$ in Fig. 4(d) to obtain the reduced graph $\Gamma_r$ in Fig. 4(e).

LEMMA 2.2. *$\Gamma_r$ is unique.*

*Proof.* Suppose that $\Gamma_r$ is not unique. This implies that there is a precedence in the order of application of the reduction operations. However, since an edge cannot be in both a series and parallel set simultaneously, the order of application does not make any difference. □

A reduction procedure is suggested in [8] for series/parallel graphs; and they show that if there is a Euler path in $\Gamma_r$, then there is a Euler path in $\Gamma$. On the other hand it is not obvious that a reduction method can be applied to general plane graphs. We present the following result for the reduction of plane multigraphs.

THEOREM 2.3. *A graph $\Gamma$ is dual Eulerian if and only if its reduced graph $\Gamma_r$ is dual Eulerian.*

**3. Identifying dual Eulerian graphs.** It is assumed throughout this section that $\Gamma$ is a reduced graph. The determination of dual paths in a plane undirected multigraph $\Gamma$ requires the traversal of edges in both itself and its dual in corresponding sequences. Suppose $v_1\, e_1\, v_2\, e_2\, v_3\, e_3\, v_4$ is a dual path in $\Gamma$ and $u_1\, e_1\, u_2\, e_2\, u_3\, e_3\, u_4$ is its corresponding dual path in $\Gamma^d$. These dual paths can be extended in length if and only if there exists an edge $e_i \in E$ such that $e_i \in (f^*(v_4) \cap f^*(u_4)) \cup (f^*(v_1) \cap f^*(u_1))$ and $e_i \notin \{e_1, e_2, e_3\}$. Therefore, the intersections of vertex cocycles of $\Gamma$ and $\Gamma^d$ indicate which edges can be adjacent in a dual path of $\Gamma$.

Let $R$ denote the set of vertex cocycles of a graph $\Gamma(V, f, E)$ (i.e., $R = \{f^*(v) | v \in V\}$). Similarly, let $R^d$ denote the set of vertex cocycles of $\Gamma^d$. Define the set $T$ as follows:

$$T = \{x \cap y | x \in R; y \in R^d\}.$$

That is, take the intersection of all possible pairs of elements of $R$ and $R^d$ and let these be the elements of the set $T$.

LEMMA 3.1. *All nonempty elements of $T$ have cardinality two.*

*Proof.* The set of vertex cocycles $R^d$ of $\Gamma^d$ corresponds to the set of facial cycles of $\Gamma$. Therefore, $T$ is formed by taking the pairwise intersections of vertex cocycles and facial cycles in $\Gamma$. Choose some vertex $v \in V$ and facial cycle $\zeta$ in $\Gamma$. If $v$ is not in the facial circuit that corresponds to the facial cycle $\zeta$, then the intersection is empty. If $v$ is contained in the facial circuit corresponding to $\zeta$, then the intersection must have exactly two elements because $\zeta$ is elementary. $\square$

The elements of $T$ represent sets of edges that are incident on a common vertex in both $\Gamma$ and $\Gamma^d$. Suppose $\{a, b\}$ is an element of $T$ for some $\Gamma$ and $\Gamma^d$. This means that $a$ and $b$ are adjacent in at least one dual path. Now it must be determined which edges can be placed adjacent to $a$ and $b$ such that a larger dual path can be constructed. For this purpose a table is constructed that is known as a *successor table.* The successor table is constructed beginning with all possible dual paths of length two. This information can be obtained from $T$. Each element of $T$, say $\{a, b\}$, is ordered in two ways, $(a \rightarrow b)$ and $(b \rightarrow a)$. These two orderings represent the traversal of $a$ then $b$ and $b$ then $a$, respectively. In order to determine which edge can be appended to the dual path $(a \rightarrow b)$, the possible successors in $\Gamma$ and $\Gamma^d$ are listed and their intersection is taken. This intersection represents the set of edges that may be appended to $(a \rightarrow b)$ in order to create a dual path of length three. The successor table for the dual plane multigraphs shown in Fig. 5 is given in Table 1.

Placed in the first column of the successor table are two ordered pairs (each in a separate row) for each element in $T$. Placed in the second (respectively, third) column are the possible successors of the ordered pair in $\Gamma$ (respectively, $\Gamma^d$). The intersection of the second and third columns is placed in the fourth column and represents the edges that may succeed the ordered pair in a dual path (i.e., a successor common to both $\Gamma$ and $\Gamma^d$). The sets in the second, third, and fourth columns of the successor table are referred to as the *$\Gamma$-successors, $\Gamma^d$-successors,* and *dual-successors,* respectively. For the example shown in Fig. 5 vertices $v_2$ of $\Gamma$ and $u_1$ of $\Gamma^d$ contribute the element $\{a, c\} \in T$ (i.e., $f^*(v_2) \cap f^*(u_1) = \{a, c\} \in T$). This element of $T$ contributes two rows to the successor table as shown in rows 9 and 10 of Table 1.

The ordered pairs in the first column of the successor table represent all possible dual paths of length two. The dual-successors of an ordered pair are a set of edges that can be appended to the dual path of length two to form a dual path of length three. Suppose $(x \rightarrow y)$ is an ordered pair of a successor table and $\{z\}$ is its dual-successor. $x\, y\, z$ is a dual path of length three, and therefore $y\, z$ must be a dual path of length two. Since all dual paths of length two are represented in the successor table, the

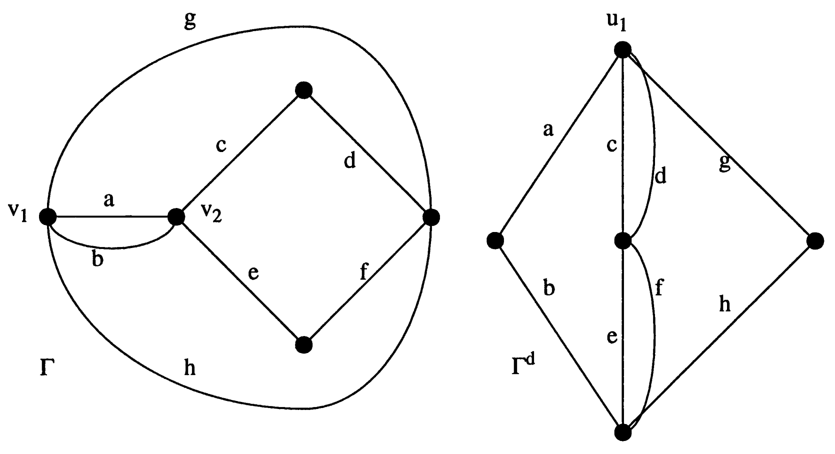


FIG. 5. *A pair of dual plane multigraphs.*

ordered pair $(y \rightarrow z)$ must be in the table. The ordered pair $(y \rightarrow z)$ has a set of dual-successors, so the dual path can be extended. Before taking this development further, let us analyze one exceptional case.

The case in which two edges are in parallel requires special consideration, since the ordered pair does not identify a unique vertex at which the dual path ends. Two edges in parallel are referred to as a *two-element circuit.* For example, in Γ of Fig. 5, $(a \rightarrow b)$ can imply either the dual path $v_1\,a\,v_2\,b\,v_1$ or $v_2\,a\,v_1\,b\,v_2$. Since it is desirable that an ordered pair identify a unique dual path, the ordered pairs corresponding to two-element circuits are split in two so that each identifies a unique dual path of length two. For example, rows 3, 4, 5, 6, 15, 16, 17, and 18 of the successor table shown in Table 1 must be split into two rows each. A new table is constructed by splitting these rows and is known as the *augmented successor table.* The augmented successor table derived from the successor table in Table 1 is shown in Table 2.

In the case where an entry in the first column of the successor table corresponds to a two-element circuit in either Γ or $\Gamma^d$ there are two sets in either the second or third column corresponding to the two possible ways to traverse the two-element circuit. Consider the two-element circuit $v_1\,a\,v_2\,b\,v_1$ in the graph of Fig. 5. The Γ-successor of $(a \rightarrow b)$ may be either $\{c, e\}$ or $\{g, h\}$ depending on whether $(a \rightarrow b)$ is traversed starting at $v_2$ or $v_1$, respectively. Rows 3 and 4 of Table 1 are contributed by the two-element circuit $\{a, b\} \in T$. The edge $b$ is the dual-successor of the ordered pairs $(g \rightarrow a), (g \rightarrow h), (c \rightarrow a)$, and $(f \rightarrow e)$ (rows 2, 5, 10, and 18 of Table 1, respectively). The dual path $a\,b$ is contained in the dual paths $g\,a\,b$ and $c\,a\,b$. Concatenating the dual paths $g\,a\,b$ and $c\,a\,b$ with the dual-successors of $(a \rightarrow b)$ yields the edge sequences $g\,a\,b\,e$, $g\,a\,b\,h$, $c\,a\,b\,e$, and $c\,a\,b\,h$. $g\,a\,b\,h$ and $c\,a\,b\,e$ are dual paths, but $g\,a\,b\,e$ and $c\,a\,b\,h$ are not. Therefore, the ordered pair $(a \rightarrow b)$ is split into two ordered pairs $(a \rightarrow b)'$ and $(a \rightarrow b)''$ with dual-successors $\{h\}$ and $\{e\}$, respectively (rows 17 and 18 of Table 2, respectively), and the dual-successors of $(g \rightarrow a)$ and $(c \rightarrow a)$ are updated

TABLE 1
*The successor table for the plane multigraphs shown in Fig.* 5.

| | Ordered Pairs | $\Gamma$-successors | $\Gamma^d$-successors | *Dual*-successors |
|---|---|---|---|---|
| 1 | $(a \to g)$ | $\{d,f,h\}$ | $\{h\}$ | $\{h\}$ |
| 2 | $(g \to a)$ | $\{b,c,e\}$ | $\{b\}$ | $\{b\}$ |
| 3 | $(a \to b)$ | $\{c,e\}$ or $\{g,h\}$ | $\{e,f,h\}$ | $\{e\}$ or $\{h\}$ |
| 4 | $(b \to a)$ | $\{c,e\}$ or $\{g,h\}$ | $\{c,d,g\}$ | $\{c\}$ or $\{g\}$ |
| 5 | $(g \to h)$ | $\{d,f\}$ or $\{a,b\}$ | $\{b,e,f\}$ | $\{f\}$ or $\{b\}$ |
| 6 | $(h \to g)$ | $\{d,f\}$ or $\{a,b\}$ | $\{a,c,d\}$ | $\{d\}$ or $\{a\}$ |
| 7 | $(b \to h)$ | $\{d,f,g\}$ | $\{g\}$ | $\{g\}$ |
| 8 | $(h \to b)$ | $\{a,c,e\}$ | $\{a\}$ | $\{a\}$ |
| 9 | $(a \to c)$ | $\{d\}$ | $\{d,e,f\}$ | $\{d\}$ |
| 10 | $(c \to a)$ | $\{b,g,h\}$ | $\{b\}$ | $\{b\}$ |
| 11 | $(c \to e)$ | $\{f\}$ | $\{b,f,h\}$ | $\{f\}$ |
| 12 | $(e \to c)$ | $\{d\}$ | $\{a,d,g\}$ | $\{d\}$ |
| 13 | $(b \to e)$ | $\{f\}$ | $\{c,d,f\}$ | $\{f\}$ |
| 14 | $(e \to b)$ | $\{a,g,h\}$ | $\{a\}$ | $\{a\}$ |
| 15 | $(c \to d)$ | $\{f,g,h\}$ | $\{a,g\}$ or $\{e,f\}$ | $\{g\}$ or $\{f\}$ |
| 16 | $(d \to c)$ | $\{a,b,e\}$ | $\{a,g\}$ or $\{e,f\}$ | $\{a\}$ or $\{e\}$ |
| 17 | $(e \to f)$ | $\{d,g,h\}$ | $\{b,h\}$ or $\{c,d\}$ | $\{h\}$ or $\{d\}$ |
| 18 | $(f \to e)$ | $\{a,b,c\}$ | $\{b,h\}$ or $\{c,d\}$ | $\{b\}$ or $\{c\}$ |
| 19 | $(d \to g)$ | $\{a,b,h\}$ | $\{h\}$ | $\{h\}$ |
| 20 | $(g \to d)$ | $\{c\}$ | $\{c,e,f\}$ | $\{c\}$ |
| 21 | $(d \to f)$ | $\{e\}$ | $\{b,e,h\}$ | $\{e\}$ |
| 22 | $(f \to d)$ | $\{c\}$ | $\{a,c,g\}$ | $\{c\}$ |
| 23 | $(f \to h)$ | $\{a,b,g\}$ | $\{g\}$ | $\{g\}$ |
| 24 | $(h \to f)$ | $\{e\}$ | $\{c,d,e\}$ | $\{e\}$ |

to $\{b\}'$ and $\{b\}''$, respectively (rows 2 and 6 of Table 2, respectively). Similarly, all other two-element circuits are split.

LEMMA 3.2. *The dual-successor of an ordered pair in the augmented successor table is unique.*

*Proof.* Assume the ordered pair is not a two-element circuit. Given an arbitrary ordered pair $(x \to y)$, without loss of generality assume it corresponds to the dual path $v_1\, x\, v_2\, y\, v_3$ in $\Gamma$ and $u_1\, x\, u_2\, y\, u_3$ in $\Gamma^d$. Since $y \in (f^*(v_3) \cap f^*(u_3))$ and $|f^*(v_3) \cap f^*(u_3)| = 2$ (Lemma 3.1), $|(f^*(v_3) \setminus \{y\}) \cap (f^*(u_3) \setminus \{y\})| = 1$. Hence, the dual-successor is unique.

Assume the ordered pair is a two-element circuit. Using the same argument as in the preceding paragraph, the dual-successor is unique for each starting vertex. Since the ordered pair is split, the dual-successor is unique. □

A directed graph $\Lambda(V, A)$, henceforth known as the *continuity graph*, is constructed where $V$ is the set of vertices and $A$ the set of directed arcs. The continuity graph is constructed in such a way that there is a one-to-one correspondence between the vertices of $\Lambda$ and the ordered pairs of the augmented successor table. Hence, for simplicity, in the following discussion the ordered pairs are used to denote the corresponding vertices of $\Lambda$. An arc is directed from vertex $(e_1 \to e_2)$ to vertex $(e_3, \to e_4)$ if and only if $e_2$ and $e_3$ are identical edges in $\Gamma$ and $\{e_4\}$ is the dual-successor of $(e_1 \to e_2)$. The arc indicates that it is possible to traverse the edges $e_1, e_2 = e_3$, and $e_4$ in $\Gamma$ and $\Gamma^d$ in that order. The continuity graph constructed from the augmented successor table of Table 2 is shown in Fig. 6.

A *chain* in a directed graph $\Lambda$ is a sequence of vertices $v_1\, v_2\, \ldots\, v_n$ such that there is an arc from $v_i$ to $v_{i+1}$ for $1 \le i < n$. A chain is said to be *closed* if $v_1 = v_n$. A chain

TABLE 2
*The augmented successor table derived from the successor table shown in Table 1.*

| | Ordered Pairs | $\Gamma$-successors | $\Gamma^d$-successors | *Dual*-successors |
|---|---|---|---|---|
| 1 | $(a \to g)$ | $\{d, f, h\}$ | $\{h\}$ | $\{h\}'$ |
| 2 | $(g \to a)$ | $\{b, c, e\}$ | $\{b\}$ | $\{b\}'$ |
| 3 | $(b \to h)$ | $\{d, f, g\}$ | $\{g\}$ | $\{g\}'$ |
| 4 | $(h \to b)$ | $\{a, c, e\}$ | $\{a\}$ | $\{a\}'$ |
| 5 | $(a \to c)$ | $\{d\}$ | $\{d, e, f\}$ | $\{d\}'$ |
| 6 | $(c \to a)$ | $\{b, g, h\}$ | $\{b\}$ | $\{b\}''$ |
| 7 | $(c \to e)$ | $\{f\}$ | $\{b, f, h\}$ | $\{f\}'$ |
| 8 | $(e \to c)$ | $\{d\}$ | $\{a, d, g\}$ | $\{d\}''$ |
| 9 | $(b \to e)$ | $\{f\}$ | $\{c, d, f\}$ | $\{f\}''$ |
| 10 | $(e \to b)$ | $\{a, g, h\}$ | $\{a\}$ | $\{a\}''$ |
| 11 | $(d \to g)$ | $\{a, b, h\}$ | $\{h\}$ | $\{h\}''$ |
| 12 | $(g \to d)$ | $\{c\}$ | $\{c, e, f\}$ | $\{c\}'$ |
| 13 | $(d \to f)$ | $\{e\}$ | $\{b, e, h\}$ | $\{e\}'$ |
| 14 | $(f \to d)$ | $\{c\}$ | $\{a, c, g\}$ | $\{c\}''$ |
| 15 | $(f \to h)$ | $\{a, b, g\}$ | $\{g\}$ | $\{g\}''$ |
| 16 | $(h \to f)$ | $\{e\}$ | $\{c, d, e\}$ | $\{e\}''$ |
| 17 | $(a \to b)'$ | $\{g, h\}$ | $\{e, f, h\}$ | $\{h\}$ |
| 18 | $(a \to b)''$ | $\{c, e\}$ | $\{e, f, h\}$ | $\{e\}$ |
| 19 | $(b \to a)'$ | $\{g, h\}$ | $\{c, d, g\}$ | $\{g\}$ |
| 20 | $(b \to a)''$ | $\{c, e\}$ | $\{c, d, g\}$ | $\{c\}$ |
| 21 | $(g \to h)'$ | $\{a, b\}$ | $\{b, e, f\}$ | $\{b\}$ |
| 22 | $(g \to h)''$ | $\{d, f\}$ | $\{b, e, f\}$ | $\{f\}$ |
| 23 | $(h \to g)'$ | $\{a, b\}$ | $\{a, c, d\}$ | $\{a\}$ |
| 24 | $(h \to g)''$ | $\{d, f\}$ | $\{a, c, d\}$ | $\{d\}$ |
| 25 | $(c \to d)'$ | $\{f, g, h\}$ | $\{a, g\}$ | $\{g\}$ |
| 26 | $(c \to d)''$ | $\{f, g, h\}$ | $\{e, f\}$ | $\{f\}$ |
| 27 | $(d \to c)'$ | $\{a, b, e\}$ | $\{a, g\}$ | $\{a\}$ |
| 28 | $(d \to c)''$ | $\{a, b, e\}$ | $\{e, f\}$ | $\{e\}$ |
| 29 | $(e \to f)'$ | $\{d, g, h\}$ | $\{c, d\}$ | $\{d\}$ |
| 30 | $(e \to f)''$ | $\{d, g, h\}$ | $\{b, h\}$ | $\{h\}$ |
| 31 | $(f \to e)'$ | $\{a, b, c\}$ | $\{c, d\}$ | $\{c\}$ |
| 32 | $(f \to e)''$ | $\{a, b, c\}$ | $\{b, h\}$ | $\{b\}$ |

in the continuity graph $\Lambda$ corresponds to a sequence of edges traversed in $\Gamma$ and $\Gamma^d$. A *maximal chain* is a chain whose set of vertices is not properly contained in the set of vertices of any other chain. For example, $(a \to c)\,(c \to d)'\,(d \to g)$ is a chain in the continuity graph of Fig. 6 and $(a \to g)\,(g \to h)'\,(h \to b)\,(b \to a)'$ is a maximal chain.

The *mirror image* of a chain $v_1\,v_2\,\ldots\,v_k$ in a continuity graph is the chain $v'_k\,v'_{k-1}\,\cdots\,v'_1$ where $v_i$ and $v'_i$ contain the same edge labels with opposite ordering (e.g., if $v_i = (a \to b)$, then $v'_i = (b \to a)$).

LEMMA 3.3. *For every chain in a continuity graph $\Lambda$, there is a mirror image of that chain also contained in $\Lambda$.*

*Proof.* Consider an arbitrary ordered pair $(a \to b)$ and its row in the augmented successor table. Suppose its dual-successor is $c$. Then the ordered pair $(b \to c)$ exists in the first column of the table, and $(a \to b)(b \to c)$ is a chain in $\Lambda$. It must be shown that $(c \to b)(b \to a)$ is also a chain in $\Lambda$. Since $(b \to c)$ is an ordered pair in the first column of the table, so is $(c \to b)$. The $\Gamma$-successors of $(a \to b)$ are the elements of the vertex cocycle of $\Gamma$ that contains $b$ and not $a$ with the element $b$ removed. That is, it is the set $f^*(x) \setminus \{b\}$ such that $b \in f^*(x)$ and $a \notin f^*(x)$. Similarly, the $\Gamma$-successors of $(c \to b)$ are the elements of the vertex cocycle $f^*(y)$ of $\Gamma$ less $b$, where $b \in f^*(y)$ and $c \notin f^*(y)$. Since $b \in f^*(x) \cap f^*(y)$ and $\Gamma$ has no loops, $x \neq y$. By hypothesis, $a$ and

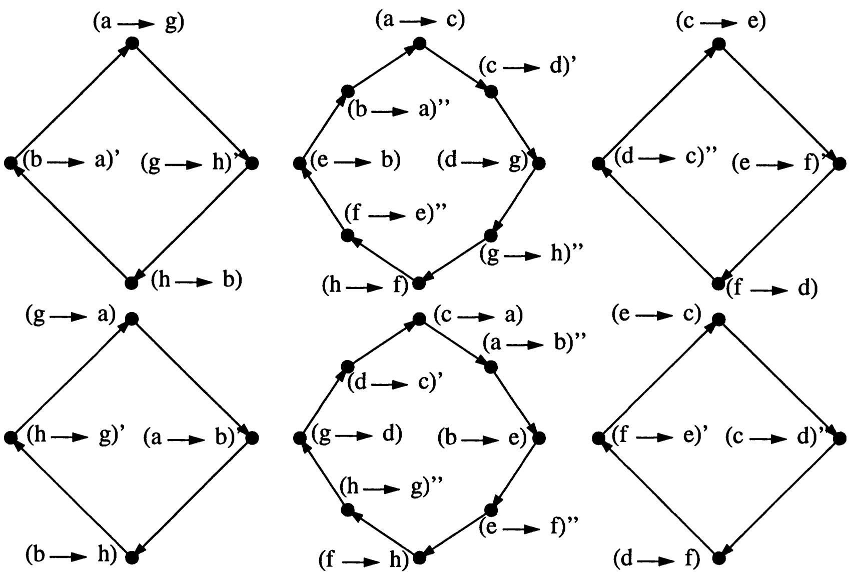


FIG. 6. *Continuity graph corresponding to the augmented successor table given in Table* 2.

$b$ must be in the same vertex cocycle of $\Gamma$ for some $v \in V$. $b$ is contained in only two vertex cocycles (i.e., $|f(e)| = 2, \forall e \in E$) and $a \notin f^*(x)$; therefore, $a \in f^*(y)$. Hence $a$ is in the set of $\Gamma$-successors of $(c \to b)$. That is, $\Gamma$ contains the subgraph shown in Fig. 7(a), and $\Gamma^d$ contains the subgraph shown in Fig. 7(b).

A similar argument shows that $a$ is in the set of $\Gamma^d$-successors of $(c \to b)$ in the augmented successor table, and therefore $(c \to b)(b \to a)$ is a chain in $\Lambda$. □

For example, the chains $(a \to g)\,(g \to h)'\,(h \to b)\,(b \to a)'$ and $(a \to b)'\,(b \to h)\,(h \to g)'\,(g \to a)$ in the continuity graph of Fig. 6 are mirror images of one another. The mirror image of a chain corresponds to the traversal of edges in $\Gamma$ and $\Gamma^d$ in the opposite direction.

LEMMA 3.4. *Every vertex in a continuity graph $\Lambda$ has exactly one incoming arc and one outgoing arc.*

*Proof.* Choose an arbitrary vertex $v \in V$. From Lemma 3.2 it is known that $v$ has one outgoing arc. From Lemma 3.3 it is known that there is a corresponding $v' \in V$ that is the mirror image of $v$. Since $v$ has one outgoing arc, $v'$ has only one incoming arc. It is also known from Lemma 3.2 that $v'$ has only one outgoing arc; hence, $v$ has only one incoming arc. □

COROLLARY 3.5. *A chain in a continuity graph $\Lambda$ is maximal if and only if it is closed.*

The property of the continuity graph $\Lambda$ stated in Lemma 3.4 makes the identification of chains in $\Lambda$ trivial. Since each chain is contained in a maximal chain, the set of all maximal chains is sufficient to characterize the edge relationships of interest. A dichotomy of the vertex set of $\Lambda$ is induced by the mirror image property. The dichotomy is obtained by placing in opposite sets chains that are mirror images of one another. Since the direction in which edges appear in the chains is not important,

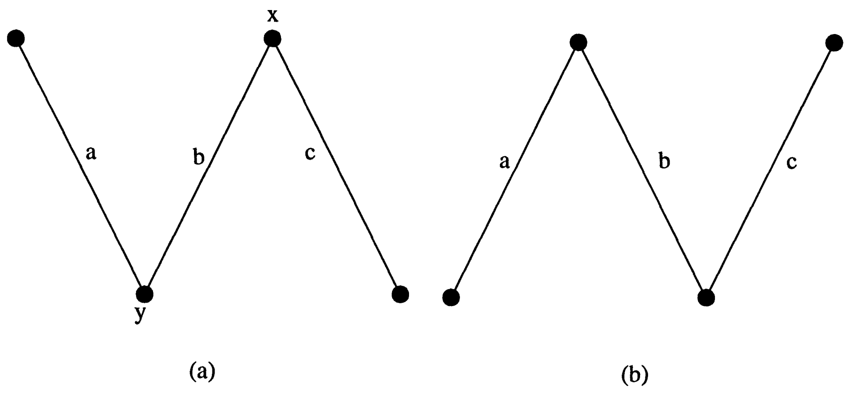


FIG. 7. *Subgraphs illustrating the proof of Lemma* 3.3.

it is sufficient to analyze only one set of the dichotomy. Let the chains that connect the vertices of one set of the dichotomy be known as the *representative set of maximal chains.*

A *desired chain* in a continuity graph $\Lambda$ is a chain, say $(e_0 \rightarrow e_1)(e_1 \rightarrow e_2)\ldots(e_n \rightarrow e_{n+1})$, such that $e_j \neq e_k \forall j \neq k \in \{1,\ldots,n\}$. A *maximal desired chain* is a desired chain of maximum length.

LEMMA 3.6. *There is a one-to-one correspondence between desired chains of* $\Lambda$ *and dual paths of* $\Gamma$.

The dual path corresponding to the desired chain $(e_0 \rightarrow e_1)(e_1 \rightarrow e_2)\ldots(e_n \rightarrow e_{n+1})$ is $e_1 e_2 \ldots e_n$. The above derivation has led to the major result of this paper, which is presented in the next theorem.

THEOREM 3.7. $\Gamma$ *is dual Eulerian if and only if there exists a desired chain that contains every edge of* $\Gamma$.

*Proof.* Assume there exists a desired chain that contains every edge. From Lemma 3.6 it is known that every desired chain has a corresponding dual path in $\Gamma$ and $\Gamma^d$. Hence, $\Gamma$ and $\Gamma^d$ contain a dual Euler path.

Suppose there is no desired chain that contains every edge. From Lemma 3.6 there is a desired chain corresponding to every dual path. Hence, there is no dual path that contains every edge. □

Theorem 2.3 in conjunction with Theorem 3.7 guarantee the successful identification of dual Eulerian plane multigraphs.

## 4. Algorithm realization and analysis.

The input to the algorithm is a specification of a plane connected undirected multigraph. The reduced graph can be obtained easily by searching for vertices of degree 2 in $\Gamma$ and $\Gamma^d$ and applying the series and parallel reduction operations. It is not necessary to identify parallel edges, since a parallel set of edges in $\Gamma$ (respectively, $\Gamma^d$) corresponds to a series set of edges in $\Gamma^d$ (respectively, $\Gamma$). The embedding of a multigraph on the plane can be specified by indexing the edges incident on each vertex in the clockwise direction. The indices are assigned in the clockwise direction with the first index being zero. Each edge has two indices assigned to it; one for each vertex on which it is incident. The index of the

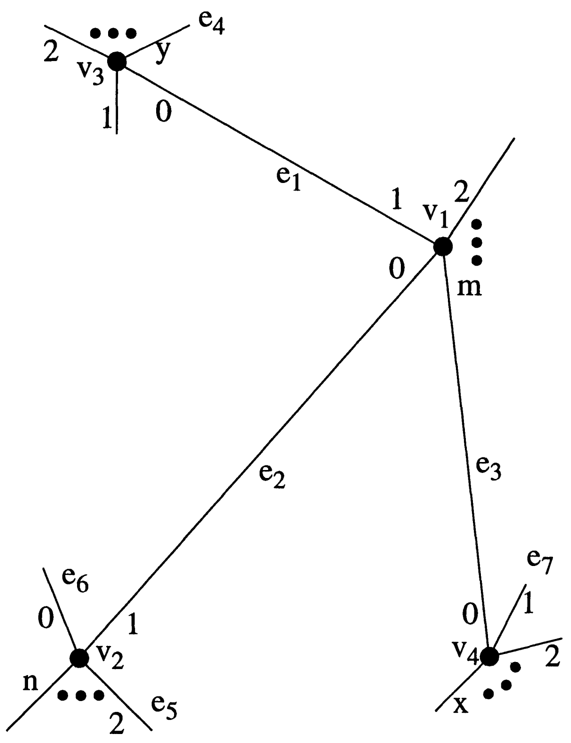


FIG. 8. *Partial graph used in the explanation of procedure* **Determine_Dual-Successors.**

edge $e$ incident on the vertex $v$ is denoted by index$(x, e)$. For example, in the partial graph shown in Fig. 8, index$(v_1, e_1) = 1$.

Each edge appears in exactly four elements of $T$, except for edges that are contained in two-element cycles; these edges appear in three elements of $T$. Consider the partial graph shown in Fig. 8. The elements of $T$ in which the edge $e_2$ appears are $\{e_2, e_1\}, \{e_2, e_3\}, \{e_2, e_5\}$, and $\{e_2, e_6\}$. These four elements can be determined in constant time due to the data structure used for a plane graph. Two edges $e_1$ and $e_2$ are said to be *neighbors* if and only if index$(v, e_1)$ = (index$(v, e_2) + 1)$ mod $|f^*(v)|$ or index$(v, e_1)$ = (index$(v, e_2) - 1)$ mod $|f^*(v)|$. The algorithm to form the set $T$ is shown in Fig. 9.

LEMMA 4.1. $|T| = 2|E| - k + 1$, *where $k$ is the number of two-element circuits in* $\Gamma$ *and* $\Gamma^d$.

*Proof.* Each $v \in V$ contributes exactly $|f^*(v)|$ elements to $T$. For each $v \in V$ these elements are unique except for vertices that are contained in two-element circuits. The same two-element set of edges in $T$ is contributed by different vertices if the vertices are contained in a two-element circuit. In addition, the empty set is an element of $T$. Hence,

$$|T| = \sum_{\forall v \in V} |f^*(v)| - k + 1.$$

```
Procedure Determine_T(){
        for (each $e \in E$){
                for (each neighbor, say $e_i$ of $e$){
                        if ($e_i$ is not marked)
                                add $\{e, e_i\}$ to $T$;
                }
                mark $e$;
        }
}
```

FIG. 9. *Algorithm for determining the set $T$.*

Since $\sum_{\forall v \in V} |f^*(v)| = 2|E|$[2], $|T| = 2|E| - k + 1$. □

Consider the example shown in Fig. 5. The neighbors of the edge $b$ in $\Gamma$ are $a$, $e$, and $h$. Therefore, during the construction of $T$ the sets $\{b, e\}$, $\{a, b\}$, and $\{b, h\}$ are placed in $T$ and another edge is considered. The neighbors of the edge $a$ in $\Gamma$ are $b$, $c$, and $g$. The sets $\{a, c\}$ and $\{a, g\}$ are placed in $T$. Since $b$ is marked, $\{a, b\}$ is not placed in $T$ a second time. This process is repeated for each edge and the result is $T = \{\emptyset, \{a, g\}, \{a, b\}, \{g, h\}, \{b, h\}, \{a, c\}, \{c, e\}, \{b, e\}, \{c, d\}, \{e, f\}, \{d, g\}, \{d, f\}, \{f, h\}\}$. This algorithm is clearly $O(|E|)$, since the number of neighbors of an edge is constant.

LEMMA 4.2. *The augmented successor table contains exactly* $4|E|$ *rows.*

*Proof.* From Lemma 4.1 it is known that $|T| = 2|E| - k + 1$, where $k$ is the number of two-element cycles in $\Gamma$ and $\Gamma^d$. Each element of $T$ (except the empty set) contributes two rows to the successor table. Therefore the successor table has $4|E| - 2k$ rows. Each two-element cycle contributes two rows to the successor table, and each row is split when the augmented successor table is formed. Hence, the number of rows in the augmented successor table is $4|E| - 2k + 2k = 4|E|$. □

The dual-successors of the ordered pairs are determined using the algorithm shown in Fig. 10.

In the graph of Fig. 5, the dual-successors of the ordered pairs $(a \rightarrow g)$ and $(g \rightarrow a)$ are $h$ and $b$, respectively. The dual-successors of the ordered pairs $(b \rightarrow h)$ and $(h \rightarrow b)$ are $g$ and $a$, respectively. The body of the *for* loop in the algorithm **Determine_Dual-Successors** is executed in constant time, and the body of the loop is executed once for each element of $T$. Therefore the time complexity of the procedure is $O(|E|)$.

Consider the augmented successor table given in Table 2. The procedure **Determine_Representative_Set_of_Maximal_Chains** (Fig. 11) works as follows. The ordered pair $(a \rightarrow g)$ is chosen as the starting point of the first maximal chain. The ordered pairs $(a \rightarrow g)$ and $(g \rightarrow a)$ are marked visited. Since the dual-successor of $(a \rightarrow g)$ is $\{h\}'$, the next ordered pair in the chain is $(g \rightarrow h)'$. The ordered pairs $(g \rightarrow h)'$ and $(h \rightarrow g)'$ are marked visited, and the next ordered pairs in the chain are $(h \rightarrow b)$ and $(b \rightarrow a)'$. Since $(a \rightarrow g)$ is the next ordered pair and it has already been visited, the maximal chain is complete. Not all the ordered pairs have been visited, so the body of the outer *while* loop is repeated. The next unvisited ordered pair in the augmented successor table is $(a \rightarrow c)$. Beginning with this ordered pair yields the maximal chain $(a \rightarrow c)\,(c \rightarrow d)'\,(d \rightarrow g)\,(g \rightarrow h)''\,(h \rightarrow f)\,(f \rightarrow e)''\,(e \rightarrow b)\,(b \rightarrow a)''$. The procedure continues until all of the ordered pairs in the table have been marked visited.

```
Procedure Determine_Dual-Successors(){
    for (each $\{e_i, e_j\} \in T$){ /*Fig. 8*/
        if ($\{e_i, e_j\}$is a two-element cycle){
                split it into two elements and mark the direction of
                traversal in each;
                place one of them in $T$ so that it gets processed in a
                successive iteration;
        }
        if (index($v_1, e_j$) = (index($v_1, e_i$) + 1) mod $|f^*(v_1)|$)){
                /*$e_i$ and $e_j$ correspond to $e_1$ and $e_2$ in Fig. 8, respectively*/
                the dual-successor of ($e_j \rightarrow e_i$) is the edge $e_l$ such that
                (index($v_3, e_l$) + 1) mod $|f^*(v_3)|$ = index($v_3, e_i$);
                /*$e_l$ corresponds to $e_4$ in Fig. 8*/
                the dual-successor of ($e_i \rightarrow e_j$) is the edge $e_l$ such that
                (index($v_2, e_l$) − 1) mod $|f^*(v_2)|$ = index($v_2, e_j$);
                /*$e_l$ corresponds to $e_5$ in Fig. 8*/
        }
        else{
                /*$e_i$ and $e_j$ correspond to $e_3$ and $e_2$ in Fig. 8, respectively*/
                the dual-successor of ($e_i \rightarrow e_j$) is the edge $e_l$ such that
                (index($v_2, e_l$) + 1) mod $|f^*(v_2)|$ = index($v_2, e_j$);
                /*$e_l$ corresponds to $e_6$ in the graph of Fig. 8*/
                the dual-successor of ($e_j \rightarrow e_i$) is the edge $e_l$ such that
                (index($v_4, e_l$) − 1) mod $|f^*(v_4)|$ = index($v_4, e_i$);
                /*$e_l$ corresponds to $e_7$ in Fig. 8*/
        }
    }
}
```

FIG. 10. *Algorithm for determining the dual-successors.*

```
Procedure Determine_Representative_Set_of_Maximal_Chains(){
    while (an unvisited ordered pair exists in the augmented successor table){
        Choose any unvisited ordered pair, say ($a \rightarrow b$), and mark it and its
        mirror image visited;
        ($x \rightarrow y$) = ($a \rightarrow b$);
        /*assign ($a \rightarrow b$) to the temporary variable ($x \rightarrow y$)*/
        do{
            choose the dual-successor of ($x \rightarrow y$), say $z$;
            ($x \rightarrow y$) = ($y \rightarrow z$);
            mark ($y \rightarrow z$) and ($z \rightarrow y$) visited;
        }while(($a \rightarrow b$) $\neq$ ($x \rightarrow y$));
        record the maximal chain;
    }
}
```

FIG. 11. *Algorithm for determining the representative set of maximal chains.*

```
Procedure Determine_Maximal_Desired_Chains(){
    for (each maximal chain in the representative set of maximal chains){
        if (the maximal chain is of the form ..., $(e_i \to e_{i+1}), (e_{i+1} \to e_{i+2}), \ldots,$
        $(e_j \to e_{i+1}), (e_{i+1} \to e_{j+1}), \ldots$){
        /*it has a repeated edge, namely $e_{i+1}$*/
            while ( $\exists e_{i+1}$ that is not the start of a maximal desired chain){
                    choose an $e_{i+1}$ arbitrarily;
                    $(y \to z) = (e_{i+1} \to e_{i+2})$;
                    do{
                        $(m \to n) = (y \to z)$;
                        while $(n \neq e_{k+1}, \forall (e_k \to e_{k+1}) \in MDC)$
                        /*MDC is the maximal desired chain currently being
                        constructed*/
                            $(m \to n) = (n \to e_l)$;
                        $(y \to z) = (e_{k+1} \to e_{k+2})$;
                    }while $((y \to z) \neq (e_{i+1} \to e_{i+2}))$;
            }
        }
        else
            the maximal chain is a maximal desired chain;
    }
}
```

FIG. 12. *Algorithm for determining the maximal desired chains.*

Lemma 3.4 and Corollary 3.5 guarantee the correctness of the procedure **Determine_Representative_Set_of_Maximal_Chains**. It is clear that this procedure has time complexity $O(|E|)$, since it visits exactly half the rows of the augmented successor table and performs a constant time task at each row.

Since an edge of $\Gamma$ appears in at most four elements of $T$, it appears in at most eight vertex labels in the continuity graph $\Lambda$ and, therefore, in at most four vertex labels in the representative set of maximal chains. Hence, an edge in $\Gamma$ can appear at most four times in a maximal chain. Based on these observations the set of maximal desired chains is obtained from the representative set of maximal chains as follows. Choose a maximal chain from the representative set of maximal chains, say $v_1 v_2 \ldots v_n v_1$. A desired chain is a *subchain* of a maximal chain that contains no repeated edges. A subchain is a chain that is contained in another chain. Therefore, a desired chain is maximal if it begins with an edge that appears twice in the maximal chain and ends just prior to an edge that is already contained in the desired chain. The maximal desired chains are determined using a constructive algorithm as follows. Start at any vertex in the maximal chain from which a maximal desired chain is computed. Traverse edges in the continuity graph until a vertex label is repeated. This sequence of edges is a maximal desired chain (i.e., it has maximum length without repeating any edge of $\Gamma$). Start the next maximal desired chain immediately following the first instance of the repeated label. Repeat this until the first maximal desired chain constructed from this maximal chain is repeated. If each repeated edge has been used as the start of a maximal desired chain, then choose a new maximal chain and repeat the outermost loop of the algorithm; otherwise, choose a repeated edge

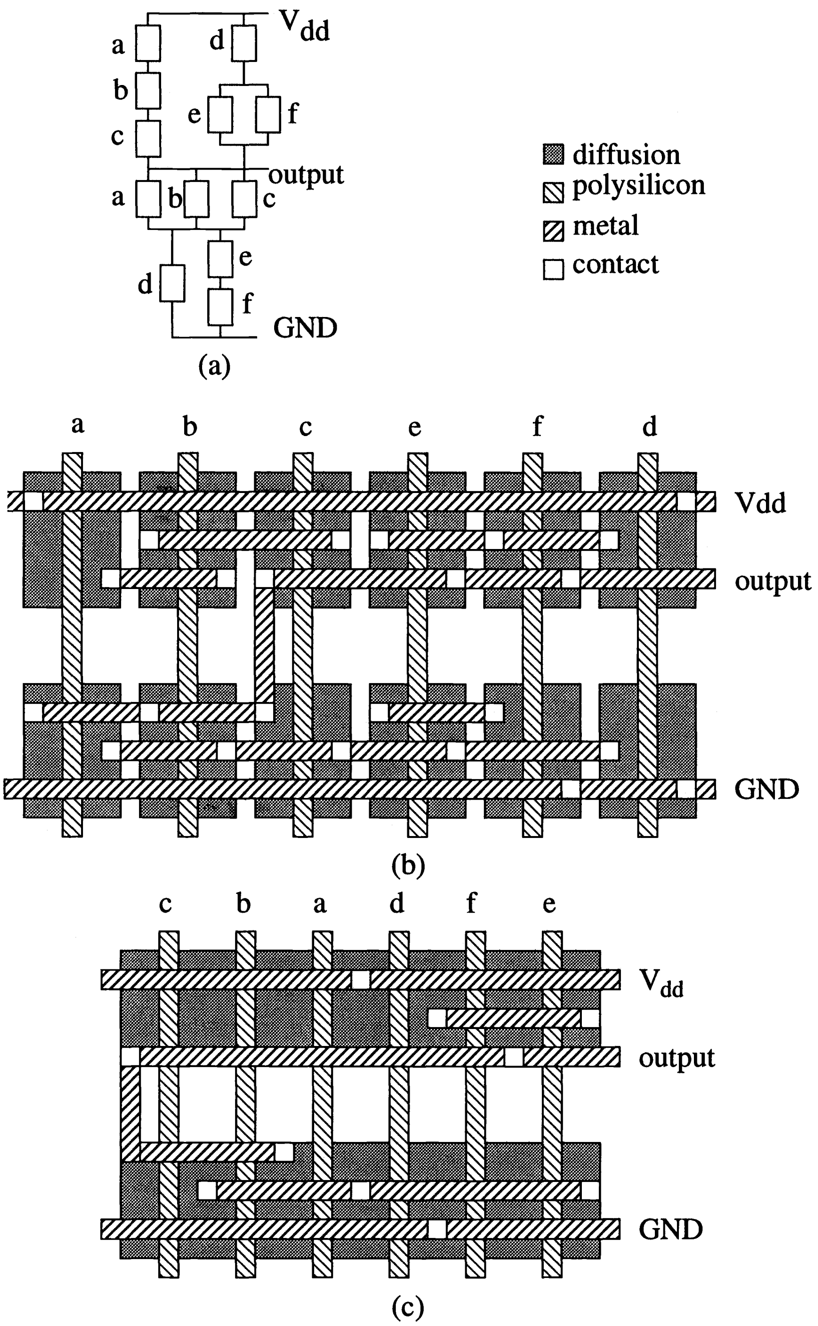


FIG. 13. (a) *A CMOS functional cell,* (b) *a physical implementation of the cell, and* (c) *a physical implementation of the cell after applying the dual Eulerian theory.*

that has not been used as the start of a maximal desired chain and continue. If a repeated edge does not exist in the maximal chain, then the maximal chain is a maximal desired chain. The algorithm is shown in Fig. 12. The maximal desired chains for the example of Fig. 5 are $a\,g\,h\,b$, $a\,c\,d\,g\,h\,f\,e\,b$, and $c\,e\,f\,d$. Since the maximal desired chain $a\,c\,d\,g\,h\,f\,e\,b$ contains every edge, the graphs shown in Fig. 5 are dual Eulerian. With the proper data structures and manipulations of memory pointers the procedure in Fig. 12 can be implemented to execute in $O(|E|)$ time.

THEOREM 4.3. *The overall time complexity of the algorithm that identifies dual Eulerian plane multigraphs is* $O(|E|)$.

## 5. Application of the theory.

In the design of functional cells for CMOS VLSI circuits it is desirable to implement each functional cell such that it requires a minimum amount of physical area [8]. For example, the CMOS functional cell shown in Fig. 13(a) can be physically implemented as shown in Fig. 13(b); however, if the dual Eulerian theory presented in this paper is applied, then it can be implemented as shown in Fig. 13(c). The reduction in area due to the application of the dual Eulerian theory is significant.

In order to apply the theory presented in this paper to the VLSI layout problem the transistor circuit is represented by an undirected multigraph. The transistors correspond to the edges of the graph, and the source and drain terminals of the transistors correspond to the vertices of the graph. In addition, an edge is placed between the output and ground (power) nodes of the circuit to eliminate isthmuses. It is also necessary that the $p$-transistor and $n$-transistor networks of the CMOS gate be dual to one another so that the corresponding graphs are dual. For details on the application of this technique we refer the reader to [1].

## 6. Conclusions.

It has been shown in this paper that the question of whether or not a plane undirected multigraph is dual Eulerian can be answered in a time proportional to a linear function of the number of edges in the graph, and an algorithm has been presented that answers this question. Interesting properties of plane multigraphs have been presented and can be summarized as follows.

1. A plane undirected multigraph $\Gamma$ is dual Eulerian if and only if its reduced graph $\Gamma_r$ is dual Eulerian (Theorem 2.3).
2. A dual path of length two can be extended in a unique way (Lemma 3.2).
3. A plane undirected multigraph $\Gamma$ is dual Eulerian if and only if there exists a desired chain that contains every edge of $\Gamma$ (Theorem 3.7).
4. The set of maximum length dual paths can be computed in linear time (Theorem 4.3).

The question of whether or not a planar multigraph admits an embedding that is dual Eulerian is interesting, since an algorithm to solve this problem can be used to determine more efficient layouts for CMOS functional cells. However, since the number of embeddings of a planar graph is exponential, it is not clear that a polynomial-time algorithm can be found for this problem. Nevertheless, to the best of our knowledge it is still an open problem and has not been proven to be $NP$-complete.

## REFERENCES

[1] B. CARLSON, *Transistor chaining and transistor reordering in the design of CMOS complex gates*, PhD thesis, Syracuse University, New York, 1991.

[2] J. GRAVER AND M. WATKINS, *Combinatorics with emphasis on the theory of graphs*, Springer-Verlag, New York, 1977.

[3] F. Harary, *Graph theory*, Addison-Wesley, Reading, MA, 1971.

[4] Y. Huang and M. Sarrafzadeh, *A parallel algorithm for minimum dual-cover with application to cmos layout*, J. Circuits, Systems Comput., 1 (1991), pp. 177–204.

[5] R. Maziasz and J. Hayes, *Layout optimization of static cmos functional cells*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 9 (1990), pp. 708–719.

[6] R. Nair, A. Bruss, and J. Reif, *Linear time algorithms for optimal cmos layout*, in VLSI: Algorithms and Architectures, P. Bertolazzi and F. Luccio, eds., Elsevier, North-Holland, Amsterdam, 1985, pp. 327–338.

[7] F. Preparata and R. Yeh, *Introduction to discrete structures*, Addison-Wesley, Reading, MA, 1973.

[8] T. Uehara and W. VanCleemput, *Optimal layout of cmos functional arrays*, IEEE Trans. Comput., C-30 (1981), pp. 305–314.

[9] S. Ueno, K. Tsuji, and Y. Kajitani, *On dual eulerian paths and circuits in plane graphs*, in Proc. of International Symposium on Circuits and Systems, 1988, pp. 1835–1838.

[10] S. Wimer, R. Pinter, and J. Feldman, *Optimal chaining of cmos transistors in a functional cell*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-6 (1987), pp. 795–801.

SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 51–61, February 1995

# ENUMERATION OF CONCRETE REGULAR COVERING PROJECTIONS [*]

M. HOFMEISTER[†]

**Abstract.** Counting covering spaces of graphs is one of the rapidly progressing aspects within the enumerative branch of topological graph theory. A covering projection is said to be concrete if it is accompanied by an explicit partition of the vertex set of the covering graph into "sheets" such that each sheet meets each vertex fiber exactly once. The natural projection (subscript erasure) of the voltage graph construction is the prototype of a concrete projection. An isomorphism of concrete covering projections maps sheets to sheets. Pólya and DeBruijn enumerative methods and Moebius inversion are used to derive a formula to count the isomorphism classes of regular covering projections of a graph.

**Key words.** graph covering, enumeration, voltage group, subgroup lattice

**AMS subject classifications.** 05C10, 05C30, 05E25, 57M15

## 1. Introduction.

In this paper we consider simple undirected graphs and their corresponding symmetric digraphs. As usual, the vertex set and the edge set of the graph $G$ are denoted by $V(G)$ and $E(G)$, respectively; the set $A(G)$ is the arc set of the corresponding symmetric digraph. An $r$-to-one graph epimorphism $p : H \to G$ which sends the neighbors of each vertex $x \in V(H)$ bijectively to the neighbors of $p(x) \in V(G)$ is called an *$r$-fold covering projection* of $G$. The graph $H$ is the *covering graph*, and the graph $G$ is the *base graph* of $p$. Topologically speaking, $p$ is a local homeomorphism. For an introduction into the field of topological graph theory see, e.g., the famous textbook by Gross and Tucker [4].

The *fibers* of the $r$-fold covering projection $p : H \to G$ are the sets $p^{-1}(v)$ ($v \in V(G)$). Inspired by Riemann surfaces, the covering projection $p$ is called *concrete*, if it is accompanied by an explicit partition $\mathcal{P} = \{P_1, \dots, P_r\}$ of the vertex set of $H$ such that every partition set $P_i$ meets every vertex fiber exactly once; we write $(p, \mathcal{P})$ for short. The partition sets $P_i$ are the *sheets* of $p$.

$$\begin{tikzcd} H \arrow[r, "\psi"] \arrow[d, "p"'] & \tilde{H} \arrow[d, "\tilde{p}"] \\ G \arrow[r, "\gamma"'] & G \end{tikzcd}$$

FIG. 1.

There is a natural kind of isomorphism between covering projections of $G$, given by a commutative diagram with an isomorphism $\psi$ and $\gamma \in \mathrm{Aut}(G)$ (see Fig. 1). An *isomorphism of concrete covering projections* $(p, \mathcal{P})$ and $(\tilde{p}, \tilde{\mathcal{P}})$ is an isomorphism of $p$

[*] Received by the editors March 21, 1993; accepted for publication (in revised form) December 27, 1993.

[†] Siemens AG, Corporate Research and Development, Department ZFE BT SE 14, D-81730 Munich, Germany (hofmeist@cayley.zfe.siemens.de).

and $\tilde{p}$ in the sense of Fig. 1 such that $\psi$ preserves sheets, that is, $\psi(P) \in \tilde{\mathcal{P}}$ for every sheet $P \in \mathcal{P}$; we write $(p, \mathcal{P}) \cong (\tilde{p}, \tilde{\mathcal{P}})$ for short.

The problem of enumerating $r$-fold covering projections of a graph $G$ up to isomorphism is still unsolved except in the cases of $r = 2$ [6] or trivial automorphism group [7]; however, nonisomorphic concrete $r$-fold covering projections of $G$ are counted in [8]. This could be done by an extensive usage of *permutation voltage assignments*, i.e., mappings $F : A(G) \to \mathcal{S}_r$ (where $\mathcal{S}_r$ is the symmetric group on the set $\{1, \dots, r\}$), such that inverse arcs obtain inverse assignments. From such an assignment one can construct the *derived graph* $G_F$ as follows. Its vertex set is $V(G) \times \{1, \dots, r\}$; two vertices $(x, i)$ and $(y, j)$ are adjacent in $G_F$ iff $x$ and $y$ are adjacent in $G$ and $j = F(x, y)(i)$. Gross and Tucker showed that the *natural projection* $p_F : G_F \to G$ (sending vertex $(x, i)$ of $G_F$ to vertex $x$ of $G$) is an $r$-fold covering projection [3]. This projection can be understood to be concrete in an obvious way by considering the sets $P_i = \{(x, i) | x \in V(G)\}$ as sheets of $p_F$; the resulting concrete $r$-fold covering projection is denoted by $(p_F, \mathcal{P}_F)$.

For the enumeration of concrete $r$-fold covering projections of the graph $G$ the following theorem is essential (see [8], Thm. 1).

THEOREM 1.1. *Let $(p, \mathcal{P})$ be a concrete $r$-fold covering projection of $G$. Then there is a permutation voltage assignment $F$ with voltages in $\mathcal{S}_r$ such that Fig.* 2 *is an isomorphism between $(p, \mathcal{P})$ and $(p_F, \mathcal{P}_F)$ for some isomorphism $\psi$.*

$$\begin{tikzcd} H \arrow[rr, "\psi"] \arrow[dr, "p"'] & & G_F \arrow[dl, "p_F"] \\ & G & \end{tikzcd}$$

FIG. 2.

For more interesting problems concerning covering projections of graphs we refer the reader to the articles [2], [10], [12]–[14].

## 2. Regular covering projections.

A covering projection $p : H \to G$ is called *regular* if there is a group $\mathcal{A}$ acting freely on $H$ such that $H/\mathcal{A}$ is isomorphic to $G$. In fact, regular $r$-fold covering projections of $G$ can be obtained from *ordinary voltage assignments* with some (abstract) *voltage group* $\mathcal{A}$ of order $r$, i.e., mappings $F : A(G) \to \mathcal{A}$ such that inverse arcs have inverse assignments. The *derived graph* $G_F$ is defined very similar to the case of permutation voltage assignments: its vertex set is $V(G) \times \mathcal{A}$. Two vertices $(x, a)$ and $(y, b)$ are adjacent in $G_F$ iff $x$ and $y$ are adjacent in $G$ and $b = F(x, y)\, a$. The natural projection $p_F : G_F \to G$ is defined by setting $p_F(x, a) = x$.

Again, Gross and Tucker showed that every regular covering projection can be represented by a derived graph of an appropriate ordinary voltage assignment (see, e.g., [4, Thm. 2.2.2]). As for $r$-fold covering projections, the general counting problem for regular $r$-fold covering projections is still unsolved; only some exceptional cases are known. Since every 2-fold covering projection of $G$ is regular, the result of [6] applies in this case, too. In [10] regular fourfold coverings of identity graphs are counted; in

[9] regular covering projections of identity graphs with voltages in finite vector spaces over finite fields are enumerated.

A concrete regular covering projection is a concrete covering projection $(p, \mathcal{P})$, such that $p : H \to G$ is regular and the members of the group $\mathcal{A}$ acting on $H$ preserve the sheets in $\mathcal{P}$. It is an easy exercise to show that every concrete regular covering projection is isomorphic to a concrete regular covering projection $(p_F, \mathcal{P}_F)$, where $p_F$ is a canonical projection that stems from an ordinary voltage assignment for $G$ in an appropriate voltage group $\mathcal{A}$, and $\mathcal{P}_F$ consists of the sets $P_a = \{(x, a) | x \in V(G)\}$ for $a \in \mathcal{A}$ as sheets; hence we may restrict attention on derived graphs.

As an example, consider the nonisomorphic concrete regular double covering projections of the complete graph $K_3$. It follows immediately from Fig. 3 that there exist regular covering projections which are isomorphic, but not in the concrete sense; for example, consider the first two covering projections of this figure, for which no sheet preserving isomorphism can be found. The purpose of this paper is to count nonisomorphic concrete regular $r$-fold covering projections. As an intermediate result, we give some Pólya-like formulas for ordinary voltage assignments.

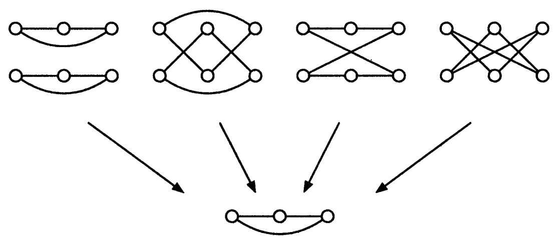


FIG. 3. *Concrete regular double covering projections of* $K_3$.

**3. A power group enumeration theorem.** Let $D$ and $R$ be finite sets, and let $\Gamma$ and $\Phi$ be two finite groups acting on $D$ and $R$, respectively. Then $\Gamma \times \Phi$ acts on the set of functions $f : D \to R$ via $(\gamma, \phi)(f) = \phi \circ f \circ \gamma^{-1}$. The computation problem for the number of orbits of this action is well studied in the literature [1]; a useful formula is given by the classical power group enumeration theorem [5].

A slight modification of this problem is to assume that the sets $D$ and $R$ contain some additional "comparable" structure, where the functions $f$ are demanded to preserve this structure. An example that will be important in the context of counting concrete regular covering projections but is also interesting on its own is given by ordinary voltage assignments of a graph $G$ with voltage group $\mathcal{A}$. The first impression is that we have a certain kind of power group enumeration: the domain is the arc set $A(G)$, where $\text{Aut}(G)$ acts in an obvious way on $A(G)$, and the range is the group $\mathcal{A}$ with the automorphism group $\text{Aut}(\mathcal{A})$ acting on it. The further structure that has to be preserved by ordinary voltage assignments is that inverse arcs must have inverse assignments. Let

$$(1) \qquad \mathcal{F}(G; \mathcal{A}) = \{F : A(G) \to \mathcal{A} \mid \forall (x, y) \in A(G) : F(x, y) = F(y, x)^{-1}\}$$

be the set of ordinary voltage assignments of $G$ with voltage group $\mathcal{A}$, and let $\Gamma \leq \text{Aut}(G)$ and $\Phi \leq \text{Aut}(\mathcal{A})$. Then $\Gamma \times \Phi$ acts on $\mathcal{F}(G; \mathcal{A})$ via

$$(2) \qquad (\gamma, \phi)(F)(x, y) = \phi(F(\gamma^{-1}(x), \gamma^{-1}(y)))$$

or $(\gamma,\phi)(F) = \phi \circ F \circ \gamma^{-1}$ for short. The number of the orbits of this action will be denoted by $\Omega(G|\Gamma;\mathcal{A}|\Phi)$. In order to count the orbits of this action, we need some further notation concerning the automorphisms of $G$ and $\mathcal{A}$. We start with the automorphisms of $G$.

Let $\gamma \in \Gamma$. Then the automorphism $\gamma$ may be understood as a permutation of vertices, edges, and arcs, respectively. A vertex cycle $\sigma$ of $\gamma$ is called *diagonal*, if it is of even length, $2q$ say, and for some (and hence for all) $x \in \sigma$, $[x, \gamma^q(x)] \in E(G)$. The corresponding edge (respectively, arc cycle) is called *diagonal*, too. For $i \in I\!N$ let $\mu_i(\gamma)$ be the number of diagonal edge cycles of $\gamma$ of length $i$, and let $\lambda_i(\gamma)$ be the number of edge cycles of length $i$ that are not diagonal. Set $\mathbf{s} = (s_1, s_2, \dots)$ and $\mathbf{t} = (t_1, t_2, \dots)$. We define the *cycle index of the graph $G$ with automorphism group* $\Gamma$ by setting

$$Z(G|\Gamma;\mathbf{s},\mathbf{t}) = \frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma}\prod_{i=1}^{m} s_i^{\lambda_i(\gamma)}\, t_i^{\mu_i(\gamma)}\,, \tag{3}$$

where $G$ is assumed to have $m$ edges. Note that $Z(G|\Gamma;\mathbf{s},\mathbf{t})$ reduces to the ordinary cycle index of the automorphism group of $G$, considered as a permutation group of edges of $G$, if $\Gamma = \mathrm{Aut}(G)$ and the automorphisms of $G$ do not contain diagonal cycles. The cycle indices of small complete graphs with full automorphism groups are tabulated in Table 1.

TABLE 1
*Graph cycle indices for complete graphs.*

| $n$ | $Z(K_n|\mathcal{S}_n;\mathbf{s},\mathbf{t})$ |
|---|---|
| 2 | $\frac{1}{2}(s_1 + t_1)$ |
| 3 | $\frac{1}{6}(s_1^3 + 3s_2t_1 + 2s_3)$ |
| 4 | $\frac{1}{24}(s_1^6 + 6s_1s_2^2t_1 + 8s_3^2 + 3s_2^2t_1^2 + 6s_4t_2)$ |
| 5 | $\frac{1}{120}(s_1^{10} + 10s_1^3s_2^3t_1 + 20s_1s_3^3 + 15s_2^4t_1^2 + 30s_4^2t_2 + 20s_3s_6t_1 + 24s_5^2)$ |

Now consider an automorphism $\phi \in \Phi$ of $\mathcal{A}$. Since $\phi$ is a permutation of the members of $\mathcal{A}$, it decomposes into disjoint cycles. We distinguish three types of cycles of $\phi$. A cycle $\tau$ of $\phi$ is of

(i) *type* 1, if $\forall\, a \in \tau : a^{-1} \notin \tau$. For $i \in I\!N$, let $\lambda_i(\phi)$ be the number of $i$-cycles of type 1 of $\phi$;

(ii) *type* 2, if $\forall\, a \in \tau : a^{-1} \neq a \wedge a^{-1} \in \tau$. For $i \in I\!N$, let $\mu_i(\phi)$ be the number of $i$-cycles of type 2 of $\phi$;

(iii) *type* 3, if $\forall\, a \in \tau : a^{-1} = a$. For $i \in I\!N$, let $\rho_i(\phi)$ be the number of $i$-cycles of type 3 of $\phi$.

We define, for $\phi \in \Phi$, the *full cycle contribution sequence* $\mathbf{C}(\phi) = (C_1(\phi), C_2(\phi), \dots)$ and the *diagonal cycle contribution sequence* $\mathbf{D}(\phi) = (D_1(\phi), D_2(\phi), \dots)$ by

$$C_i(\phi) = \sum_{d|i} d(\lambda_d(\phi) + \mu_d(\phi) + \rho_d(\phi))\,, \tag{4}$$

$$D_i(\phi) = \sum_{\frac{2i}{d}\equiv 1 \bmod 2} d\mu_d(\phi) \;+\; \sum_{d|i} d\rho_d(\phi)\,. \tag{5}$$

Now we are ready to state and to prove our power group enumeration theorem for ordinary voltage assignments. Remember the definition of $\Omega(G|\Gamma;\mathcal{A}|\Phi)$ above.

THEOREM 3.1. *The number of orbits of ordinary voltage assignments in $\mathcal{F}(G;\mathcal{A})$ determined by the power group action described in (2) is*

$$\Omega(G|\Gamma;\mathcal{A}|\Phi)=\frac{1}{|\Phi|}\sum_{\phi\in\Phi}Z(G|\Gamma;\mathbf{C}(\phi),\mathbf{D}(\phi)). \tag{6}$$

*Proof.* According to Burnside's lemma (which in fact is due to Cauchy-Frobenius, as we know from [11]), we have to count the ordinary voltage assignments $F: A(G)\to\mathcal{A}$ that are fixed under $(\gamma,\phi)\in\Gamma\times\Phi$. We conclude from (2) that the assignment $F$ is fixed under $(\gamma,\phi)$ iff

$$\phi(F(x,y))=F(\gamma(x),\gamma(y)) \tag{7}$$

for every $(x,y)\in A(G)$. Let $\pi$ be the edge cycle of the automorphism $\gamma$ containing $[x,y]$, and let $s(\pi)$ denote the length of $\pi$. We distinguish two cases.

*Case* 1. Assume that the edge cycle $\pi$ is not diagonal. Then $\pi$ corresponds to two arc cycles where the one of them contains the inverse arcs of the other. Assign to $(x,y)$ a value $F(x,y)\in\mathcal{A}$. From (7) we obtain by successive iteration that

$$\phi^{s(\pi)}(F(x,y))=F(x,y). \tag{8}$$

It follows that $F(x,y)$ has to be chosen from a cycle of the automorphism $\phi$ such that its length divides $s(\pi)$; hence there are $C_{s(\pi)}(\phi)$ possible choices for $F(x,y)$.

*Case* 2. If the edge cycle $\pi$ is diagonal, then it follows from (7) that

$$\phi^{s(\pi)}(F(x,y))=F(y,x)=F(x,y)^{-1}. \tag{9}$$

Hence the cycles of the automorphism $\phi$ from which $F(x,y)$ must be chosen cannot be of type 1. If such a cycle is of type 2, then $\frac{2s(\pi)}{d}\equiv 1 \bmod 2$, where $d$ is the length of this cycle. If it is of type 3, then its length is a divisor of $s(\pi)$. Hence there are $D_{s(\pi)}(\phi)$ possible choices for $F(x,y)$.

Summarizing, we obtain

$$\begin{aligned}\Omega(G|\Gamma;\mathcal{A}|\Phi)&=\frac{1}{|\Phi|}\frac{1}{|\Gamma|}\sum_{\phi\in\Phi}\sum_{\gamma\in\Gamma}\prod_{\pi\in\mathcal{C}(\gamma)}C_{s(\pi)}(\phi)\prod_{\pi\in\mathcal{D}(\gamma)}D_{s(\pi)}(\phi)\\&=\frac{1}{|\Phi|}\sum_{\phi\in\Phi}Z(G|\Gamma;\mathbf{C}(\phi),\mathbf{D}(\phi)),\end{aligned}$$

where $\mathcal{C}(\gamma)$ and $\mathcal{D}(\gamma)$ are the sets of not diagonal and diagonal edge cycles of $\gamma$, respectively. □

There are some conclusions that follow immediately. The first one we present is the case of classical power group enumeration, which appears if the automorphism group $\Gamma$ of $G$ does not contain automorphisms with diagonal edge cycles.

COROLLARY 3.2. *If $\Gamma$ does not contain automorphisms with diagonal edge cycles, then*

$$\Omega(G|\Gamma;\mathcal{A}|\Phi)=\frac{1}{|\Phi|}\sum_{\phi\in\Phi}Z(\Gamma_e;\mathbf{C}(\phi)), \tag{10}$$

*where the cycle index is the ordinary cycle index of the group $\Gamma$; $\Gamma$ is to be understood as permutation group of edges, which is indicated by the e-index.*

Another interesting case appears if the chosen automorphism group of $\mathcal{A}$ is trivial. This case corresponds to the classical Pólya theorem.

COROLLARY 3.3. *Let $s$ be the number of self-inverse elements of $\mathcal{A}$, and let $r = |\mathcal{A}|$. Then*

$$\Omega(G|\Gamma; \mathcal{A}|\mathcal{I}_{\mathcal{A}}) = Z(G|\Gamma; \mathbf{r}, \mathbf{s}), \tag{11}$$

*where $\mathbf{r} = (r, r, \dots)$ and $\mathbf{s} = (s, s, \dots)$.*

As a further example, assume that $\mathcal{A} = \mathbb{Z}_r$, the cyclic group of order $r$, which is written additively. Then $\mathrm{Aut}(\mathbb{Z}_r) = \mathbb{Z}_r^*$, the (multiplicative) group of numbers in $\mathbb{Z}_r$ that are relatively prime to $r$. Let $\lambda \in \mathbb{Z}_r^*$. It is easy to see that $C_i(\lambda)$ is the number of solutions of $\lambda^i z = z$ in $\mathbb{Z}_r$, while $D_i(\lambda)$ is the number of solutions of the equation $\lambda^i z = -z$ in $\mathbb{Z}_r$; hence $C_i(\lambda) = \gcd(\lambda^i - 1, r)$, while $D_i(\lambda) = \gcd(\lambda^i + 1, r)$. Table 2 shows the numbers $\Omega(K_n|\mathcal{S}_n; \mathbb{Z}_r|\mathbb{Z}_r^*)$ for small values of $n$ and $r$. For the computation we took the cycle indices of the complete graphs from Table 1.

TABLE 2
*Some numbers $\Omega(K_n|\mathcal{S}_n; \mathbb{Z}_r|\mathbb{Z}_r^*)$.*

| $n\backslash r$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 2 | 3 | 2 | 4 | 2 | 4 | 3 | 4 |
| 3 | 1 | 4 | 6 | 13 | 10 | 16 | 16 | 42 | 33 | 40 |
| 4 | 1 | 11 | 30 | 148 | 205 | 1181 | 906 | 3154 | 3923 | 11021 |
| 5 | 1 | 34 | 342 | 5162 | 21240 | 259965 | 396593 | 2263962 | 4861983 | 20909774 |

It is often helpful for concrete computations to dispose of an effective encoding of the cycle types of group automorphisms. For this we introduce the *cycle index of the group $\mathcal{A}$ with automorphism group $\Phi$* to be the polynomial

$$Z(\mathcal{A}|\Phi, \mathbf{s}, \mathbf{t}, \mathbf{u}) = \frac{1}{|\Phi|} \sum_{\phi \in \Phi} \prod_{i=1}^{r} s_i^{\lambda_i(\phi)} t_i^{\mu_i(\phi)} u_i^{\rho_i(\phi)}, \tag{12}$$

where $\mathbf{s} = (s_1, s_2, \dots)$, $\mathbf{t} = (t_1, t_2, \dots)$, and $\mathbf{u} = (u_1, u_2, \dots)$. Remember that $\lambda_i(\phi)$, $\mu_i(\phi)$, and $\rho_i(\phi)$ are the numbers of $i$-cycles of type 1, 2, and 3, respectively. Note that the classical cycle index of the automorphism group of $\mathcal{A}$ can be obtained by setting $\mathbf{s} = \mathbf{t} = \mathbf{u}$. Some examples of group cycle indices with full automorphism group are given in Table 3. Note that, for $i \in \mathbb{N}$, $\mathcal{D}_i$ is the dihedral group on $i$

TABLE 3
*Some group cycle indices.*

| $\mathcal{A}$ | $Z(\mathcal{A}|\mathrm{Aut}(\mathcal{A}); \mathbf{s}, \mathbf{t}, \mathbf{u})$ |
|---|---|
| $\mathbb{Z}_2^2$ | $\frac{1}{6}(u_1^4 + 3u_1^2u_2 + 2u_1u_3)$ |
| $\mathbb{Z}_2 \times \mathbb{Z}_4$ | $\frac{1}{8}(s_1^4u_1^4 + 2s_1^2t_2u_1^2u_2 + 2t_4u_1^2u_2 + 2s_2^2u_1^4 + t_2^2u_1^4)$ |
| $\mathcal{D}_3$ | $\frac{1}{6}(s_1^2u_1^4 + 3t_2u_1^2u_2 + 2s_1^2u_1u_3)$ |
| $\mathcal{D}_4$ | $\frac{1}{4}(s_1^2u_1^6 + s_1^2u_1^2u_2^2 + 2t_2u_1^4u_2)$ |

vertices. Using the information encoded in Table 3 we can compute the numbers $\Omega(K_n|\mathcal{S}_n; \mathcal{A}|\mathrm{Aut}(\mathcal{A}))$ presented in Table 4.

TABLE 4
*Some numbers* $\Omega(K_n|\mathcal{S}_n;\mathcal{A}|\mathrm{Aut}(\mathcal{A}))$.

| $n\backslash\mathcal{A}$ | $\mathbb{Z}_2^2$ | $\mathbb{Z}_2\times\mathbb{Z}_4$ | $\mathcal{D}_3$ | $\mathcal{D}_4$ |
|---|---|---|---|---|
| 2 | 2 | 4 | 3 | 5 |
| 3 | 7 | 34 | 17 | 51 |
| 4 | 64 | 1896 | 469 | 3547 |
| 5 | 1908 | 1152547 | 88477 | 2295980 |

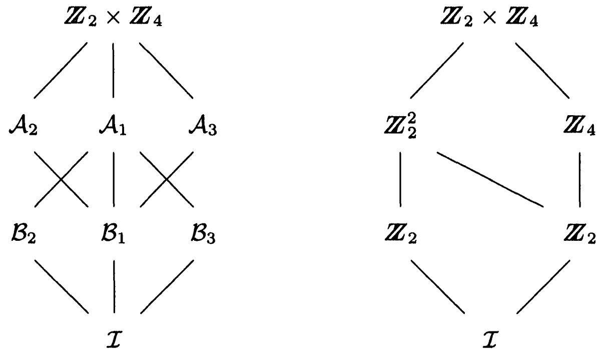


FIG. 4. *A subgroup lattice and its abstract subgroup poset.*

In the area of ordinary power group enumeration the question of counting orbits containing surjective functions arises in a natural way. Several counting formulas are known (see, e.g., [1]). The analogous problem for ordinary voltage assignments is to count orbits such that the image of the assignment generates the whole voltage group. More precisely, let

$$\overline{\mathcal{F}}(G;\mathcal{A}) = \{F \in \mathcal{F}(G;\mathcal{A}) \mid \ < \mathrm{im}\, F > = \mathcal{A}\}. \tag{13}$$

Then $\Gamma\times\Phi$ acts on $\overline{\mathcal{F}}(G;\mathcal{A})$ as described by (2). The number of orbits of this action will be denoted by $\overline{\Omega}(G|\Gamma;\mathcal{A}|\Phi)$. Our purpose is to develop a formula for these numbers for automorphism groups $\Phi$ of $\mathcal{A}$ that satisfy a certain kind of regularity that will be explained now.

Let $\mathcal{L}(\mathcal{A})$ be the subgroup lattice of the finite group $\mathcal{A}$. Then the automorphism group $\Phi$ of $\mathcal{A}$ acts on $\mathcal{L}(\mathcal{A})$ in a canonical way; the orbits of subgroups $\mathcal{U}$ of $\mathcal{A}$ will be denoted by $\Phi(\mathcal{U})$. The subgroup relation which is the partial order of $\mathcal{L}(\mathcal{A})$ induces an ordering of the orbits of this action by setting $\Phi(\mathcal{U}) \leq \Phi(\mathcal{V})$ iff there exist $\tilde{\mathcal{U}} \in \Phi(\mathcal{U}), \tilde{\mathcal{V}} \in \Phi(\mathcal{V})$ such that $\tilde{\mathcal{U}} \leq \tilde{\mathcal{V}}$. The poset built up by this relation is denoted by $\mathcal{L}(\mathcal{A}/\Phi)$, which we will call the *abstract subgroup poset* of the group $\mathcal{A}$.

As an example, consider the subgroup lattice of $\mathbb{Z}_2\times\mathbb{Z}_4$. The subgroups of $\mathbb{Z}_2\times\mathbb{Z}_4$ are $\mathcal{A}_1 = <(1,0),(0,2)> \cong \mathbb{Z}_2^2$, $\mathcal{A}_2 = <(0,1)> \cong \mathbb{Z}_4$, $\mathcal{A}_3 = <(1,1)> \cong \mathbb{Z}_4$, $\mathcal{B}_1 = <(0,2)> \cong \mathbb{Z}_2$, $\mathcal{B}_2 = <(1,0)> \cong \mathbb{Z}_2$, $\mathcal{B}_3 = <(1,2)> \cong \mathbb{Z}_2$, and $\mathcal{I}$. The Hasse diagram of the subgroup lattice is depicted on the left side of Fig. 4.

Now assume that $\Phi = \mathrm{Aut}(\mathbb{Z}_2\times\mathbb{Z}_4)$. There are automorphisms mapping $\mathcal{A}_2$ onto $\mathcal{A}_3$ and $\mathcal{B}_2$ onto $\mathcal{B}_3$. The Hasse diagram of the abstract subgroup poset is depicted

$$\begin{tikzcd}
<\operatorname{im} F> \arrow[r, "\phi"] \arrow[d, "\psi"'] & <\operatorname{im} \tilde{F}> \arrow[d, "\tilde{\psi}"] \\
\mathcal{U} \arrow[r, "\varphi"'] & \mathcal{U}
\end{tikzcd}$$

FIG. 5.

on the right side of Fig. 4. We used abstract representations for the members of this poset. This notation is a bit sloppy but suggestive: $\mathbb{Z}_2 \times \mathbb{Z}_4$ contains two subgroups $\mathbb{Z}_2$ that are essentially different in the sense that there is no automorphism in $\Phi$ mapping one of these subgroups onto the other.

We write $\Phi_{\mathcal{U}}$ to denote the fixed group of the subgroup $\mathcal{U}$ of $\mathcal{A}$ in $\Phi$. We say that $\Phi$ *preserves fixed groups*, if

$$\mathcal{U} \le \mathcal{V} \Rightarrow \forall \phi \in \Phi_{\mathcal{U}} \exists \tilde{\phi} \in \Phi_{\mathcal{V}} : \phi|\mathcal{U} = \tilde{\phi}|\mathcal{U}. \tag{14}$$

It is easy to see that every automorphism group $\Phi$ of $\mathcal{A}$ preserves fixed groups, if $\mathcal{L}(\mathcal{A})$ has dimension $\le 2$.

Let $[\Phi(\mathcal{U})]$ be the order ideal of $\mathcal{L}(\mathcal{A}/\Phi)$ generated by $\Phi(\mathcal{U})$. We say that the automorphism group $\Phi$ of $\mathcal{A}$ is *order ideal preserving* if

$$[\Phi(\mathcal{U})] \cong \mathcal{L}(\mathcal{U}/\Phi_{\mathcal{U}}), \tag{15}$$

for every subgroup $\mathcal{U}$ of $\mathcal{A}$. Clearly, cyclic groups are order ideal preserving, no matter what automorphism groups are considered; however, there exist groups together with automorphisms that are not. Consider for example the group $\mathbb{Z}_2^3$. For abbreviation we introduce the symbols $a = (1,0,0)$, $b = (0,1,0)$, and $c = (0,0,1)$. We take the cyclic group generated by $(a\,b\,c)$ as automorphism group $\Phi$. The group $\Phi_{<a,b>}$ is trivial; it follows that $\mathcal{L}(<a,b>/\Phi_{<a,b>})$ looks like a diamond, while $[\Phi(<a,b>)]$ is a chain of $\mathcal{L}(\mathbb{Z}_2^3/\Phi)$.

The Moebius function of $\mathcal{L}(\mathcal{A}/\Phi)$ is denoted by $\mu(\Phi(\mathcal{U}), \Phi(\mathcal{V}))$. Remember the previous definition of $\overline{\Omega}(G|\Gamma; \mathcal{A}|\Phi)$.

THEOREM 3.4. *Let the automorphism group $\Phi$ of $\mathcal{A}$ preserve fixed groups and order ideals. Then the number of orbits of ordinary voltage assignments in $\overline{\mathcal{F}}(G; \mathcal{A})$ determined by the power group action described by* (2) *is*

$$\overline{\Omega}(G|\Gamma; \mathcal{A}|\Phi) = \sum_{\Phi(\mathcal{U}) \in \mathcal{L}(\mathcal{A}/\Phi)} \mu(\Phi(\mathcal{U}), \Phi(\mathcal{A}))\, \Omega(G|\Gamma; \mathcal{U}|\Phi_{\mathcal{U}}). \tag{16}$$

*Proof.* Let $\mathcal{V} \le \mathcal{A}$. We will develop an expression for $\Omega(G|\Gamma; \mathcal{V}|\Phi_{\mathcal{V}})$. Let $\mathcal{U} \le \mathcal{V}$. Assume that $F, \tilde{F} \in \mathcal{F}(G; \mathcal{V})$ are chosen so that $<\operatorname{im} F>$ and $<\operatorname{im} \tilde{F}>$ are isomorphic to $\mathcal{U}$ via automorphisms $\psi, \tilde{\psi} \in \Phi_{\mathcal{V}}$. If $\tilde{F}(\gamma(x), \gamma(y)) = \phi(F(x,y))$ with $(\gamma, \phi) \in \Gamma \times \Phi_{\mathcal{V}}$, then Fig. 5 commutes with $\varphi = \tilde{\psi} \circ \phi \circ \psi^{-1} \in \Phi_{\mathcal{V}}$. Clearly, $\varphi \in \Phi_{\mathcal{U}}$. We conclude that $H = \psi(F)$ and $\tilde{H} = \tilde{\psi}(\tilde{F})$ are in the same orbit of power group action of $\Gamma \times \Phi_{\mathcal{U}}$ on $\overline{\mathcal{F}}(G, \mathcal{U})$. These arguments can be conversed by observing that each $\varphi \in \Phi_{\mathcal{U}}$ can be chosen to be in $\Phi_{\mathcal{V}}$, since $\Phi$ preserves fixed groups. This gives rise to a bijection between orbits of ordinary voltage assignments $F \in \mathcal{F}(G;\mathcal{V})$ under $\Gamma \times \Phi_{\mathcal{V}}$ such that $< \operatorname{im} F >$ is isomorphic to $\mathcal{U}$ via a member of $\Phi_{\mathcal{V}}$ and orbits of ordinary voltage assignments in $\overline{\mathcal{F}}(G;\mathcal{U})$ under $\Gamma \times \Phi_{\mathcal{U}}$. We conclude that

$$\begin{aligned}\Omega(G|\Gamma;\mathcal{V}|\Phi_{\mathcal{V}}) &= \sum_{\Phi(\mathcal{U})\in\mathcal{L}(\mathcal{V}/\Phi_{\mathcal{V}})} \overline{\Omega}(G|\Gamma;\mathcal{U}|\Phi_{\mathcal{U}}) \\ &= \sum_{\Phi(\mathcal{U})\in[\Phi(\mathcal{V})]} \overline{\Omega}(G|\Gamma;\mathcal{U}|\Phi_{\mathcal{U}}),\end{aligned}$$

where the last equation follows from the fact that $\Phi$ preserves order ideals. Applying Moebius inversion we obtain

$$\overline{\Omega}(G|\Gamma;\mathcal{V}|\Phi_{\mathcal{V}}) = \sum_{\Phi(\mathcal{U})\in[\Phi(\mathcal{V})]} \mu(\Phi(\mathcal{U}),\Phi(\mathcal{V}))\,\Omega(G|\Gamma;\mathcal{U}|\Phi_{\mathcal{U}}). \tag{17}$$

The assertion follows by using the full group $\mathcal{A}$ for $\mathcal{U}$. □

Again we consider the particular case of trivial automorphism group of $\mathcal{A}$. The corresponding formula for surjective functions can be obtained by the principle of inclusion and exclusion.

COROLLARY 3.5. *Let $s_{\mathcal{U}}$ be the number of self-inverse elements of the subgroup $\mathcal{U}$ of $\mathcal{A}$, and let $r_{\mathcal{U}} = |\mathcal{U}|$. Then*

$$\overline{\Omega}(G|\Gamma;\mathcal{A}|\mathcal{I}_{\mathcal{A}}) = \sum_{\mathcal{U}\le\mathcal{A}} \mu(\mathcal{U},\mathcal{A})\,Z(G|\Gamma;\mathbf{r}_{\mathcal{U}},\mathbf{s}_{\mathcal{U}}), \tag{18}$$

*where $\mathbf{r}_{\mathcal{U}} = (r_{\mathcal{U}}, r_{\mathcal{U}}, \dots)$, $\mathbf{s}_{\mathcal{U}} = (s_{\mathcal{U}}, s_{\mathcal{U}}, \dots)$, and $\mu$ is the Moebius function of $\mathcal{L}(\mathcal{A})$.*

Table 5 contains numbers $\overline{\Omega}(K_n|S_n;\mathcal{A}|\mathrm{Aut}(\mathcal{A}))$ for small numbers $n$ and groups $\mathcal{A}$. Each of them is computed by the formula of Theorem 3.4 using the results of Tables 2 and 4.

TABLE 5
*Some numbers $\overline{\Omega}(K_n|S_n;\mathcal{A}|\mathrm{Aut}(\mathcal{A}))$.*

| $n\backslash\mathcal{A}$ | $\mathcal{I}$ | $\mathbb{Z}_2$ | $\mathbb{Z}_3$ | $\mathbb{Z}_4$ | $\mathbb{Z}_2^2$ | $\mathbb{Z}_5$ | $\mathbb{Z}_6$ | $\mathcal{D}_3$ | $\mathbb{Z}_7$ | $\mathbb{Z}_8$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 3 | 5 | 9 | 3 | 9 | 7 | 8 | 15 | 29 |
| 4 | 1 | 10 | 29 | 137 | 53 | 204 | 1141 | 429 | 905 | 3006 |
| 5 | 1 | 33 | 341 | 5128 | 1874 | 21239 | 259590 | 88102 | 396592 | 2258800 |

## 4. Topological enumeration.

**4. Topological enumeration.** Remember that our original problem was to count nonisomorphic concrete regular covering projections. We start with a theorem that describes how to characterize such concrete regular covering projections up to isomorphism by their ordinary voltage assignments and isomorphisms of subgroups of their voltage groups.

THEOREM 4.1. *Let $F \in \mathcal{F}(G;\mathcal{A})$ and $\tilde{F} \in \mathcal{F}(G;\tilde{\mathcal{A}})$ be ordinary voltage assignments. Then the following are equivalent:*

1. $(p_F, \mathcal{P}_F) \cong (p_{\tilde{F}}, \mathcal{P}_{\tilde{F}})$.

2. *There exists a bijection $\varphi : \mathcal{A} \to \tilde{\mathcal{A}}$ and $\gamma \in \mathrm{Aut}(G)$ such that, for every $(x,y) \in A(G)$ and $a \in \mathcal{A}$,*

$$\tilde{F}(\gamma(x), \gamma(y)) = \varphi(F(x,y)\,a)\,\varphi(a)^{-1}. \tag{19}$$

*Moreover, $\varphi$ can be chosen so that $\varphi$ restricted to $< \mathrm{im}\, F >$ is an isomorphism between $< \mathrm{im}\, F >$ and $< \mathrm{im}\, \tilde{F} >$; in this case, $\varphi$ maps cosets of $< \mathrm{im}\, F >$ to cosets of $< \mathrm{im}\, \tilde{F} >$ via $\varphi(< \mathrm{im}\, F > a) =< \mathrm{im}\, \tilde{F} > \varphi(a)$.*

*Proof.* Let $(p_F, \mathcal{P}_F) \cong (p_{\tilde{F}}, \mathcal{P}_{\tilde{F}})$. Then Fig. 1 commutes for $p_F$ and $p_{\tilde{F}}$ with an isomorphism $\psi$ and $\gamma \in \mathrm{Aut}(G)$. Since $\psi$ preserves sheets, we have $\psi(x,a) = (\gamma(x), \varphi(a))$ for some bijection $\varphi : \mathcal{A} \to \tilde{\mathcal{A}}$. Since $\psi$ is an isomorphism, there is an edge between $(x,a)$ and $(y,b)$ in $G_F$ iff there is an edge between $(\gamma(x), \varphi(a))$ and $(\gamma(y), \varphi(b))$ in $G_{\tilde{F}}$. It follows from the definition of derived graphs that $b = F(x,y)\,a$ and $\varphi(b) = \tilde{F}(\gamma(x), \gamma(y))\,\varphi(a)$. The first part of part (2) of the theorem can now be obtained by a simple substitution.

A short calculation shows that (19) is satisfied with $\tilde{\varphi}(a) := \varphi(a)\,\varphi(1)^{-1}$ instead of $\varphi(a)$, too; note that $\tilde{\varphi}(1) = 1$. Clearly, $\tilde{\varphi}$ preserves multiplication; hence it is an isomorphism between $< \mathrm{im}\, F >$ and $< \mathrm{im}\, \tilde{F} >$ that maps the cosets as claimed by the theorem.

Conversely, it is easy to see that $\psi(x,a) = (\gamma(x), \varphi(a))$ is the desired graph isomorphism. □

For $r \in \mathbb{N}$, we denote the set of nonisomorphic groups of order $r$ by $\mathcal{G}_r$. Remember that $\overline{\Omega}(G|\Gamma; \mathcal{A}|\Phi)$ is the number of orbits of the action of $\Gamma \times \Phi$ on $\overline{\mathcal{F}}(G; \mathcal{A})$. The classification given in Theorem 4.1 is the base of the following enumeration formula.

THEOREM 4.2. *The number of nonisomorphic concrete regular $r$-covering projections of the graph $G$ is*

$$\sum_{d|r} \sum_{\mathcal{U} \in \mathcal{G}_d} \overline{\Omega}(G|\mathrm{Aut}(G); \mathcal{U}|\mathrm{Aut}(\mathcal{U})). \tag{20}$$

*Proof.* Let $\mathcal{A}, \tilde{\mathcal{A}} \in \mathcal{G}_r$. Assume that there are given $F \in \mathcal{F}(G; \mathcal{A})$ and $\tilde{F} \in \mathcal{F}(G; \tilde{\mathcal{A}})$ such that $(p_F, \mathcal{P}_F) \cong (p_{\tilde{F}}, \mathcal{P}_{\tilde{F}})$. Set $< \mathrm{im}\, F >= \mathcal{V}$ and $< \mathrm{im}\, \tilde{F} >= \tilde{\mathcal{V}}$. By Theorem 4.1 there exists an isomorphism $\varphi : \mathcal{V} \to \tilde{\mathcal{V}}$ such that $\tilde{F}(\gamma(x), \gamma(y)) = \varphi(F(x,y))$ for some $\gamma \in \mathrm{Aut}(G)$. Assume that $|\mathcal{V}| = |\tilde{\mathcal{V}}| = d$, and let $\mathcal{U} \in \mathcal{G}_d$ such that $\mathcal{V}$ and $\tilde{\mathcal{V}}$ are isomorphic to $\mathcal{U}$ via isomorphisms $\chi$ and $\tilde{\chi}$, respectively. Setting $H = \chi(F)$, $\tilde{H} = \tilde{\chi}(\tilde{F})$, and $\phi = \tilde{\chi} \circ \varphi \circ \chi^{-1}$ we obtain two ordinary voltage assignments in $\overline{\mathcal{F}}(G; \mathcal{U})$ that are in the same orbit of power group action by $\mathrm{Aut}(G) \times \mathrm{Aut}(\mathcal{U})$ via $(\gamma, \phi)$.

TABLE 6
*Numbers of nonisomorphic concrete regular covering projections of complete graphs.*

| $n \backslash r$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 2 | 3 | 2 | 4 | 2 |
| 3 | 1 | 4 | 6 | 16 | 10 | 24 | 16 |
| 4 | 1 | 11 | 30 | 201 | 205 | 1610 | 906 |
| 5 | 1 | 34 | 342 | 7036 | 21240 | 348067 | 396593 |

Conversely, if $H, \tilde{H} \in \overline{\mathcal{F}}(G; \mathcal{U})$ are in the same orbit via $(\gamma, \phi)$, set $F = \chi^{-1}(H)$ and $\tilde{F} = \tilde{\chi}^{-1}(\tilde{H})$. Then part 2 of Theorem 4.1 is satisfied with $\varphi = \tilde{\chi}^{-1} \circ \phi \circ \chi$, extending $\varphi$ to the full group $\mathcal{A}$ in the following way. If $(c_i)$ and $(\tilde{c}_i)$ are representative systems for the cosets of $\mathcal{V}$ and $\tilde{\mathcal{V}}$, respectively (excluding $\mathcal{V}$ and $\tilde{\mathcal{V}}$), then set $\varphi(c_i) = \tilde{c}_i$ and $\varphi(f\,c_i) = \varphi(f)\,\tilde{c}_i$ for $f \in \mathcal{V}$.

Hence there is a bijection between nonisomorphic concrete regular covering projections of $G$ such that the corresponding voltage assignments generate a group isomorphic to $\mathcal{U}$, and the orbits of power group action $\mathrm{Aut}(G) \times \mathrm{Aut}(\mathcal{U})$ on $\overline{\mathcal{F}}(G;\mathcal{U})$, which proves the theorem. □

COROLLARY 4.3. *If $p$ is a prime, then the number of nonisomorphic concrete regular $p$-fold covering projections is $\Omega(G|\mathrm{Aut}(G);\mathbb{Z}_p|\mathbb{Z}_p^*)$.*

Table 6 presents the numbers of nonisomorphic concrete regular $r$-fold covering projections of complete graphs $K_n$ for small numbers $r$ and $n$.

## REFERENCES

[1] N. deBruijn, *Pólya's theory of counting*, in Applied Combinatorial Mathematics, E. F. Beckenbach, ed., Wiley, New York, 1964, pp. 144–184.

[2] R. Ewen and M. Hofmeister, *On coverings of the complete graph with 4 vertices*, Ars Combin., 35 (1993), pp. 87–96.

[3] J. L. Gross and T. W. Tucker, *Generating all graph coverings by permutation voltage assignments*, Discrete Math., 18 (1977), pp. 273–283.

[4] ———, *Topological Graph Theory*, Wiley Interscience Series in Discrete Mathematics and Optimization, John Wiley, 1987.

[5] F. Harary and E. M. Palmer, *Graphical Enumeration*, Academic Press, New York and London, 1973.

[6] M. Hofmeister, *Counting double covers of graphs*, J. Graph Theory, 12 (1988), pp. 437–444.

[7] ———, *Isomorphisms and automorphisms of graph coverings*, Discrete Math., 98 (1991), pp. 175–183.

[8] ———, *Concrete graph covering projections*, Ars Combin., 32 (1991), pp. 121–127.

[9] ———, *Graph covering projections arising from finite vector spaces over finite fields*, Discrete Math., to appear.

[10] S. Hong and J. H. Kwak, *Regular fourfold coverings with respect to the identity isomorphism*, J. Graph Theory, 17 (1993), pp. 621–627.

[11] A. Kerber, *Algebraic Combinatorics via Finite Group Actions*, BI-Wiss. Verl., Mannheim, Wien, Zürich, 1991.

[12] J. H. Kwak and J. Lee, *Isomorphism classes of graph bundles*, Canad. J. Math., 42 (1990), pp. 747–761.

[13] ———, *Counting some finite-fold coverings of a graph*, Graphs Combinatorics, 8 (1992), pp. 277–285.

[14] ———, *Isomorphism classes of cycle permutation graphs*, Discrete Math., 105 (1992), pp. 131–142.

SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 62–75, February 1995

# EXTREMAL CAYLEY DIGRAPHS OF FINITE CYCLIC GROUPS*

XING-DE JIA†

**Abstract.** Let Cay $(m, A)$ denote the Cayley digraph of $\mathbf{Z}_m$ generated by $A$, where $\mathbf{Z}_m$ is the cyclic group of residues modulo $m$. Let $r(m, A)$ denote the average distance of Cay $(m, A)$. For any $r \geq 1$ and $k \geq 1$ define $m^*(r, k)$ as the largest positive integer $m$ such that the average distance of the Cayley digraph Cay$(m, A)$ is at most $r$ for some set $A$ with $k$ elements. In this paper, an asymptotic formula for $m^*(r, 2)$ is proved and a lower bound for $m^*(r, k)$ is also obtained for $k \geq 3$. Applications to the construction of optimal distributed loop networks are discussed in this paper. A lower bound of the average order of subsets for asymptotic bases in number theory is proved using the main theorem of this paper.

**Key words.** Cayley digraphs, average distance, explicit construction, extremal problems

**AMS subject classifications.** Primary 05C25, 05C35; Secondary 11B13

## 1. Introduction.

Let $\Gamma$ be a given nontrivial finite group with a generating set $A$. The *Cayley digraph* of $\Gamma$ with respect to $S$, denoted Cay $(\Gamma, A)$, is a digraph whose vertex set is $\Gamma$, and $(x, y)$ is a (directed) edge if and only if $x^{-1}y \in A$. Let $m$ be a positive integer. Let $\mathbf{Z}_m$ denote the additive group of residue classes modulo $m$. Let $A$ be a generating set of $\mathbf{Z}_m$. We use Cay $(m, A)$ to denote the Cayley digraph Cay $(\mathbf{Z}_m, A)$. In this paper, we focus on Cayley digraphs of $\mathbf{Z}_m$.

Let $x, y \in \mathbf{Z}_m$, and $d(x, y)$ denote the *distance* from $x$ to $y$, and $d(\text{Cay}\,(m, A))$ denote the diameter of Cay $(m, A)$. For any integers $m \geq 2$ and $k \geq 2$, define

$$d(m, k) = \max_{A \text{ with } |A|=k} d(\text{Cay}\,(m, A)),$$

where $|A|$ denotes the cardinality of the set $A$. For any $d \geq 1$ and $k \geq 2$, define

$$m(d, k) = \max\{m \mid d(\text{Cay}\,(m, A)) \leq d \text{ for some } A \text{ with} |A| = k\}.$$

Wong and Coppersmith [22] proved that

$$m(d, k) \geq \left(\frac{d}{k} + 1\right)^k.$$

Hsu and Jia [13] proved among other results that

$$m(d, 2) = \left\lfloor \frac{d(d+4)}{3} \right\rfloor + 1$$

for all $d \geq 2$, and as $d \to \infty$,

$$\frac{1}{16}d^3 + \frac{3}{8}d^2 + O(d) \leq m(d, 3) \leq \frac{1}{14 - 3\sqrt{3}}(d + 3)^3.$$

Jia [18] recently proved that, for fixed $k \geq 4$ as $d \to \infty$,

$$m(d, k) \geq \alpha_k \left(\frac{256}{125}\right)^{\lfloor k/4 \rfloor} \left(\frac{d}{k}\right)^k + O(d^{k-1}),$$

* Received by the editors September 30, 1992; accepted for publication (in revised form) February 7, 1994. This research was supported in part by a Southwest Texas State University summer research grant.

† Department of Mathematics, Southwest Texas State University, San Marcos, Texas 78666 (jia@xingde.math.swt.edu).

where

$$\alpha_k = \begin{cases} 1 & \text{if } k \equiv 0 \text{ or } 1 \pmod 4, \\ \dfrac{4}{3} & \text{if } k \equiv 2 \pmod 4, \\ \dfrac{27}{16} & \text{if } k \equiv 3 \pmod 4. \end{cases}$$

Chen and Gu improved this lower bound in [6]. The best-known lower bound is due to Su [21], who proved that

$$\begin{aligned} n(d,k) &\geq \beta_k \left(\frac{5^5 \cdot 7^4}{17^5}\right)^{\lfloor k/5 \rfloor} \left(\frac{d}{k}\right)^k + O(d^{k-1}) \\ &\approx \beta_k (5.2844)^{\lfloor k/5 \rfloor} \left(\frac{d}{k}\right)^k + O(d^{k-1}), \end{aligned}$$

where

$$\beta_k = \begin{cases} 1 & \text{if } k \equiv 0, 1 \pmod 5, \\ 4/3 & \text{if } k \equiv 2 \pmod 5, \\ \dfrac{4752}{2197} \approx 2.163 & \text{if } k \equiv 3 \pmod 5, \\ \dfrac{165888}{50625} = 3.2768 & \text{if } k \equiv 4 \pmod 5. \end{cases}$$

Let $r(m, A)$ denote the average distance of $\mathrm{Cay}(m, A)$, that is

$$r(m, A) = \frac{1}{m} \sum_{x \in \mathbf{Z}_m} d(0, x),$$

where $d(x, y)$ is the distance from $x$ to $y$. It is obvious that the average distance is less than or equal to its diameter. In many cases, the average distance reflects more information about the Cayley digraph.

Given $m \geq 2$ and $k \geq 1$, define

$$r(m, k) = \min_{|A|=k} r(m, A).$$

On the other hand, given a real number $r \geq 1$ for any finite set $A$ of integers, let $m^*(r, A)$ denote the greatest positive integer $m$ such that the average distance $r(m, A)$ of the Cayley digraph $\mathrm{Cay}(m, A)$ associated with $\mathbf{Z}_m$ and $A$ is less than or equal to $r$. For any real number $r \geq 1$ and any integer $k \geq 1$, define

$$m^*(r, k) = \max_{|A|=k} m^*(r, A).$$

In other words, $m^*(r, k)$ is the largest positive integer $m$ such that the average distance of the Cayley digraph $\mathrm{Cay}(m, A)$ is at most $r$ for some set $A$ with $k$ elements. In this paper, we are interested in these two extremal functions. We shall prove in §2 an asymptotic formula for $m^*(r, 2)$, namely

$$m^*(r, 2) = \frac{27}{25} r^2 + O(r) \quad \text{as } r \to \infty.$$

We obtain in §3 the following lower bound for $m^*(r,3)$:

$$m^*(r,3) \geq \frac{256}{729}r^3 + O(r^2) \approx 0.351r^3 + O(r^2).$$

In §4, using the lower bounds for $m^*(r,2)$ and $m^*(r,3)$, we prove the following lower bound $m^*(r,k)$ for any fixed $k \geq 3$ as $r$ tends to infinity:

$$m^*(r,k) \geq \gamma_k \left(\frac{4\sqrt[3]{4}}{3}\right)^k \left(\frac{r}{k}\right)^k + O(r^{k-1}),$$

where $\gamma_k$ is a constant defined as follows:

$$\gamma_k = \begin{cases} 1 & \text{if } k \equiv 0 \pmod 3, \\ \dfrac{3}{2\sqrt[3]{4}} \approx 0.945 & \text{if } k \equiv 1 \pmod 3, \\ \dfrac{243}{200\sqrt[3]{2}} \approx 0.964 & \text{if } k \equiv 2 \pmod 3. \end{cases}$$

One fundamental goal in constructing a communication network is to minimize the transmission delay within the network when the number of nodes and the number of links are given. We often face the following problem when we design a network. Given a limit to the transmission delay and the number of links that a node can have, what is the maximal number of nodes a network may have and what is its structure? The Cayley digraph Cay$(m, A)$ is often referred as a distributed loop network. Its vertices represent the nodes of the network, and the edges represent the links in the network. The diameter of the Cayley digraph corresponds to the transmission delay within the network, and the average distance of the Cayley digraph corresponds to the average transmission delay of the network. Distributed loop networks with minimal transmission delay have been studied extensively by a number of authors, see, for instance, Wong and Coppersmith [22], Fiol et al. [9], Erdös and Hsu [8], Hsu and Jia [13], Hsu and Shapiro [14], [15], and a recent survey by Bermond, Comellas, and Hsu [3]. In §5, we shall apply the results of this paper to obtain an infinite class of optimal double loop networks with respect to their average distances and an upper bound for $r(m,k)$ for $k \geq 3$, which improves that of Wong and Coppersmith [22]. In the last section of this paper, we study an application to a problem in additive number theory concerning the average order of subsets of asymptotic bases.

**2. Asymptotic formula for $m^*(r, 2)$.** A subset $A$ of $\mathbf{Z}_m$ is called a *basis* of order $h$ if every element in $\mathbf{Z}_m$ is a sum of at most $h$ not necessarily distinct elements of $A$. For any $n \in \mathbf{Z}_m$, let $f(A,n)$ denote the least number of elements in $A$ with sum $n$. The *average order* of $A$ as a basis for $\mathbf{Z}_m$ is defined as

$$\frac{1}{m}\sum_{n\in\mathbf{Z}_n} f(A,n).$$

It is clear that $f(A,n)$ is the distance from 0 to $n$ in the Cayley digraph Cay$(m,A)$, and the average order of $A$ is the average distance $r(m,A)$ of Cay$(m,A)$. A similar result can be found also in [9].

THEOREM 2.1.

$$m^*(r,2) = \frac{27}{25}r^2 + O(r) \quad \text{as} \quad r \to \infty.$$

*Proof.* First, we show that

$$m^*(r,2) \geq \frac{27}{25}r^2 + O(r) \quad \text{as} \quad r \to \infty.$$

It suffices to construct a basis $A = \{a_1, a_2\}$ of average order at most $r$ for $\mathbf{Z}_m$, where $m = \frac{27}{25}r^2 + O(r)$.

Let $r \geq 2$ be any positive real number. Let

$$\lambda = \left\lfloor \frac{3r}{5} \right\rfloor, \quad a = 3\lambda, \quad m = \lambda a + \lambda.$$

Define $A = \{1, a\}$. We now calculate the average order $r(m, A)$ of $A$ as a basis for $\mathbf{Z}_m$. Let $n \in [0, m)$ be any integer. If $n \in [\lambda a, m)$, we write

$$n = (n - \lambda a) \cdot 1 + \lambda \cdot a.$$

Hence

(1) $$f(A,n) \leq (n - \lambda a) + \lambda \quad \text{for} \quad n \in [\lambda a, m).$$

Now assume that $n \in [ta, (t+1)a)$ for some $t : 0 \leq t < \lambda$. Noticing that $a = 3\lambda$, we see that $(t+1)a - \lambda \in [ta, (t+1)a)$. When $n \in [ta, (t+1)a - \lambda)$, $n$ can be represented as $n = (n - ta) \cdot 1 + t \cdot a$. This implies that

(2) $$f(A,n) \leq (n - ta) + t \quad \text{for} \quad n \in [ta, (t+1)a - \lambda).$$

If $n \in [(t+1)a - \lambda, (t+1)a)$, then $n + m \in [(\lambda + t + 1)a, (\lambda + t + 1)a + \lambda)$. Hence we have

$$n \equiv n + m = (n - (\lambda + t + 1)a) \cdot 1 + (\lambda + t + 1) \cdot a \pmod{m}.$$

Hence

(3) $$f(A,n) \leq (n - (\lambda + t + 1)a) + (\lambda + t + 1),$$

for $n \in [(t+1)a - \lambda, (t+1)a)$. It then follows from (1), (2), and (3) that

$$\begin{aligned}
\sum_{n=0}^{m-1} f(A,n) &= \sum_{n=0}^{\lambda a - 1} f(A,n) + \sum_{n=\lambda a}^{m-1} f(A,n) \\
&= \sum_{t=0}^{\lambda-1} \left( \sum_{n=ta}^{(t+1)a-\lambda-1} f(A,n) + \sum_{n=(t+1)a-\lambda}^{(t+1)a-1} f(A,n) \right) + \sum_{n=\lambda a}^{m-1} f(A,n) \\
&\leq \sum_{t=0}^{\lambda-1} \left( \sum_{x=0}^{2\lambda-1} (x+t) + \sum_{x=0}^{\lambda-1} (x + \lambda + 1) \right) + \sum_{x=0}^{\lambda-1} (x + \lambda) \\
&= \sum_{t=0}^{\lambda-1} \left( \frac{7\lambda^2}{2} - \frac{\lambda}{2} + 3\lambda t \right) + \frac{3\lambda^2}{2} - \frac{\lambda}{2} \\
&= 5\lambda^3 - \frac{\lambda^2}{2} - \frac{\lambda}{2} \\
&\leq 5\lambda^3.
\end{aligned}$$

Therefore,

$$\begin{aligned} r(m, A) &= \frac{1}{m} \sum_{n=0}^{m-1} f(A, n) \leq \frac{5\lambda^3}{\lambda a + \lambda} \\ &= \frac{5\lambda^3}{3\lambda^2 + \lambda} \leq \frac{5}{3}\lambda \leq r, \end{aligned}$$

which implies that

$$m^*(r, 2) \geq m^*(r, A) = m = 3\lambda^2 + \lambda = \frac{27}{25} r^2 + O(r).$$

We now prove

$$m^*(r, 2) \leq \frac{27}{25} r^2 + O(r). \tag{4}$$

Let $A = \{a, b\}$ be a set such that

$$m^*(r, A) = m^*(r, 2) = m.$$

Using a similar argument of Wong and Coppersmith [22],[1] we can show that

$$r(m, A) \geq \sqrt{\frac{25m}{27}} - 1,$$

which implies (4) because $r \geq r(m, A)$. The proof of Theorem 2.1 is complete. □

**3. Lower bound for $m^*(r, 3)$.** In this section, we shall construct a set $A$ of three elements, which gives the following lower bound for $m^*(r, 3)$.

THEOREM 3.1.

$$m^*(r, 3) \geq \frac{256}{729} r^3 + O(r^2) \approx 0.351 r^3 + O(r^2).$$

*Proof.* Let $r \geq 3$. Define

$$\begin{aligned} \lambda &= \left\lfloor \frac{4r}{9} \right\rfloor, \\ b &= 4\lambda, \\ c &= \lambda b + \lambda, \\ m &= \lambda c + 2\lambda. \end{aligned}$$

Let $A = \{1, b, c\}$. We now calculate the average order $r(m, A)$ of $A$ as a basis for $\mathbf{Z}_m$.

Let $1 \leq v \leq \lambda$ and $1 \leq w \leq \lambda$ be any integers. If $n \in [m + (v-1)b + wc,\ vb + (r+w)c)$, then

$$n = (n - m - (v-1)b - wc) \cdot 1 + (v-1) \cdot b + w \cdot c,$$

which implies

$$f(A, n) \leq (n - (v-1)b - wc) + (v-1) + w.$$

[1] Wong and Coppersmith originally considered only a special case where $A = \{1, b\}$. But their proof with a suitable modification still works in the general case $A = \{a, b\}$.

Noting that

$$\{vb + (\lambda + w)c\} - \{m + (v-1)b + wc\} = 2\lambda,$$

we see that

$$\begin{aligned}\sum_{m+(v-1)b+wc\le n<vb+(\lambda+w)c} f(A,n) &\le \sum_{t=0}^{2\lambda-1}((v-1)+w+t)\\ &= 2\lambda(v+w)+2\lambda^2-3\lambda.\end{aligned}$$

It follows directly from the definition that the interval

$$[vb + (\lambda + w)c,\ m + (\lambda + v)b + (w-1)c)$$

contains $\lambda$ integers. If $n$ is one of these integers, then

$$n = (n - vb - (\lambda + w)c)\cdot 1 + v\cdot b + (\lambda + w)\cdot c,$$

which implies that

$$f(A,n) \le (n - vb - (\lambda + w)c) + v + (\lambda + w).$$

Therefore,

$$\begin{aligned}\sum_{vb+(\lambda+w)c\le n<m+(\lambda+v)b+(w-1)c} f(A,m) &\le \sum_{t=0}^{\lambda-1}(v+\lambda+w+t)\\ &= \lambda(v+w)+\frac{3}{2}\lambda^2-\frac{1}{2}\lambda.\end{aligned}$$

Noting that

$$\{m + vb + wc\} - \{m + (\lambda + v)b + (w-1)c\} = \lambda,$$

and that every integer $n$ in $[m + (\lambda + v)b + (w-1)c,\ m + vb + wc)$ can be written as

$$n = (n - m - (\lambda + v)b - (w-1)c)\cdot 1 + (\lambda + v)\cdot b + (w-1)\cdot c,$$

we have

$$\begin{aligned}\sum_{m+(\lambda+v)b+(w-1)c\le n<m+vb+wc} f(A,n) &\le \sum_{t=0}^{\lambda-1}(\lambda+v+w-1+t)\\ &= \lambda(v+w)+\frac{3}{2}\lambda^2-\frac{3}{2}\lambda.\end{aligned}$$

Therefore,

$$\begin{aligned}\sum_{m+(v-1)b+wc\le n<m+vb+wc} f(A,n) = &\sum_{m+(v-1)b+wc\le n<vb+(\lambda+w)c} f(A,n)\\ &+\sum_{vb+(\lambda+w)c\le n<m+(\lambda+v)b+(w-1)c} f(A,m)\end{aligned}$$

$$\begin{aligned}
&+ \sum_{m+(\lambda+v)b+(w-1)c\le n<m+vb+wc} f(A,n)\\
&\le \{2\lambda(v+w)+2\lambda^2-3\lambda\}\\
&\quad+\left\{\lambda(v+w)+\frac{3}{2}\lambda^2-\frac{1}{2}\lambda\right\}\\
&\quad+\left\{\lambda(v+w)+\frac{3}{2}\lambda^2-\frac{3}{2}\lambda\right\}\\
&=4\lambda(v+w)+5\lambda^2-5\lambda.
\end{aligned}$$

Since $v$ is arbitrary on $[1,\lambda]$, we see that

$$\begin{aligned}
\sum_{m+wc\le n<m+\lambda b+wc} f(A,n) &= \sum_{v=1}^{\lambda} \sum_{m+(v-1)b+wc\le n<m+vb+wc} f(A,n)\\
&\le \sum_{v=1}^{\lambda}(4\lambda(v+w)+5\lambda^2-5\lambda)\\
&=4\lambda^2w+7\lambda^3-3\lambda^2.
\end{aligned}$$

It is easy to see that the length of the interval $[m+\lambda b+wc,\ m+(w+1)c)$ is $\lambda$, and that every integer $n$ is this interval can be written as

$$n=(n-m-\lambda b-wc)\cdot 1+\lambda\cdot b+w\cdot c.$$

Hence

$$\begin{aligned}
\sum_{m+\lambda b+wc\le n<m+(w+1)c} f(A,n) &\le \sum_{t=0}^{\lambda-1}(\lambda+w+t)\\
&=\lambda w+\frac{3}{2}\lambda^2-\frac{1}{2}\lambda.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\sum_{m+wc\le n<m+(w+1)c} f(A,n) &= \sum_{m+wc\le n<m+\lambda b+wc} + \sum_{m+\lambda b+wc\le n<m+(w+1)c}\\
&\le \{4\lambda^2w+7\lambda^3-3\lambda^2\}+\left\{\lambda w+\frac{3}{2}\lambda^2-\frac{1}{2}\lambda\right\}\\
&=4\lambda^2w+7\lambda^3-\frac{3}{2}\lambda^2+\lambda w-\frac{1}{2}\lambda.
\end{aligned}$$

Since $w$ is arbitrary on $[1,\lambda]$, we have

$$\begin{aligned}
\sum_{m+c\le n<m+(\lambda+1)c} f(A,n) &= \sum_{w=1}^{\lambda} \sum_{m+wc\le n<m+(w+1)c} f(A,n)\\
&\le \sum_{w=1}^{\lambda}\left\{4\lambda^2w+7\lambda^3-\frac{3}{2}\lambda^2+\lambda w-\frac{1}{2}\lambda\right\}\\
&=9\lambda^4-3\lambda^3-\lambda^2.
\end{aligned}$$

Similarly,

$$\sum_{m+(\lambda+1)c\le n<2m+c} f(A,n) \le \sum_{t=0}^{2\lambda-1} (\lambda+1+t) = 4\lambda^2+\lambda,$$

because the length of the interval $[m+(\lambda+1)c,\ 2m+c)$ is $2\lambda$, and every integer $n$ in this interval has representation

$$n = (n-m-(\lambda+1)c)\cdot 1 + (\lambda+1)\cdot c.$$

Therefore,

$$\sum_{m+c\le n<2m+c} f(A,n) \le 9\lambda^4 - 3\lambda^3 + 3\lambda^2 + \lambda.$$

Finally, we have

$$\begin{aligned} r(m,A) &= \frac{1}{m}\sum_{m+c\le n<2m+c} f(A,n) \\ &\le \frac{9\lambda^4-3\lambda^3+3\lambda^2+\lambda}{4\lambda^3+\lambda^2+2\lambda} \\ &\le \frac{9}{4}\lambda \le r, \end{aligned}$$

that is, $A$ is a basis of average order at most $r$ for $\mathbf{Z}_m$. Therefore,

$$m^*(r,3) \ge m = 4\lambda^3+\lambda^2+2\lambda = \frac{256}{729}r^3 + O(r^2).$$

The proof is complete. □

## 4. Lower bound for $m^*(r,k)$.

THEOREM 4.1. *For any* $k \ge 1$,

$$m^*(r,k) \ge \gamma_k \left(\frac{4\sqrt[3]{4}}{3}\right)^k \left(\frac{r}{k}\right)^k + O(r^{k-1}),$$

*where*

$$\gamma_k = \begin{cases} 1 & \text{if } k\equiv 0 \pmod 3, \\ \dfrac{3}{2\sqrt[3]{4}} \approx 0.945 & \text{if } k \equiv 1 \pmod 3, \\ \dfrac{243}{200\sqrt[3]{2}} \approx 0.964 & \text{if } k\equiv 2 \pmod 3. \end{cases}$$

To prove Theorem 4.1, we need the following lemma, which is analogous to Lemma 1 in [18].

LEMMA 4.2. *Let* $r_1 \ge 1$ *and* $r_2 \ge 1$ *be any real numbers,* $k_1$ *and* $k_2$ *any positive integers. Then*

$$m^*(r_1+r_2, k_1+k_2) \ge m^*(r_1,k_1)\cdot m^*(r_2,k_2).$$

*Proof.* Assume $A_s = \{a_{s1}, a_{s2}, \dots, a_{sk_s}\}$ is such that

$$m^*(r_s, A_s) = m^*(r_s, k_s) \quad \text{for} \quad s = 1, 2.$$

Let $n$ be any integer. Let $p = f(A_1, n)$. Suppose

$$n \equiv a_{1i_1} + \cdots + a_{1i_p} \pmod{m_1},$$

thus,

$$n = n'm_1 + a_{1i_1} + \cdots + a_{1i_p}$$

for some $n'$. Suppose $q = f(A_2, n')$ and

$$n' \equiv a_{2i_1} + \cdots + a_{2i_q} \pmod{m_2}.$$

Hence,

$$n \equiv a_{1i_1} + \cdots + a_{1i_p} + m_1 a_{2i_1} + \cdots + m_1 a_{2i_q} \pmod{m_1 m_2}.$$

Let $A = A_1 \cup \{m_1 a_{21} \dots, m_1 a_{2k_2}\}$; then

$$f(A, n) \le p + q = f(A_1, n) + f(A_2, n').$$

Since

$$r_s \ge r(m_s, A_s) = \frac{1}{m_s} \sum_{x=1}^{m_s} f(A_s, x),$$

we see that $A$, as a basis for $\mathbf{Z}_{m_1 m_2}$, has average order

$$\begin{aligned}
r(m, A) &= \frac{1}{m_1 m_2} \sum_{n=1}^{m_1 m_2} f(A, n) \\
&\le \frac{1}{m_1 m_2} \sum_{n=1}^{m_1 m_2} (f(A_1, n) + f(A_2, n')) \ \text{(where } n = n'm_1 + n_0,\ 0 \le n_o < m_1) \\
&= \frac{1}{m_1 m_2} \left( m_2 \sum_{x=1}^{m_1} f(A_1, x) + m_1 \sum_{x=1}^{m_2} f(A_2, x) \right) \\
&= \frac{1}{m_1} \sum_{x=1}^{m_1} f(A_1, x) + \frac{1}{m_2} \sum_{x=1}^{m_2} f(A_2, x) \\
&\le r_1 + r_2.
\end{aligned}$$

The proof of the lemma is complete. $\square$

*Proof of Theorem* 3.1. If $k = 3q$, we write $u = r/q$. It follows from Theorem 4.1 and Lemma 4.2 that

$$\begin{aligned}
m^*(r, k) = m^*(qu, 3q) &\ge m^*(u, 3)^q \\
&\ge \left( \frac{256}{729} u^3 + O(u^2) \right)^q \\
&= \left( \frac{256}{729} \right)^q \left( \frac{r}{q} \right)^{3q} + O(u^{2q-1}) \\
&\ge \left( \frac{4\sqrt[3]{4}}{3} \right)^k \left( \frac{r}{k} \right)^k + O(r^{k-1}).
\end{aligned}$$

It is easy to see that

$$m^*(r,1) = \lfloor 2r \rfloor. \tag{5}$$

We now assume that $k \equiv 1$ or $2 \pmod 3$. Write $k = k' + u$, where $u = 1$ or $2$. Let $q = r/k$. It follows from the above argument and (5), Theorem 2.1, and Lemma 4.2 that

$$\begin{aligned} m^*(r,k) &= m^*(qk' + qu, k' + u) \\ &\geq m^*(qk', k')m^*(qu, u) \\ &\geq \left\{ \left(\frac{4\sqrt[3]{4}}{3}\right)^{k'} \left(\frac{qk'}{k'}\right)^{k'} + O((qk')^{k'-1}) \right\} \cdot m^*(qu, u) \\ &= \gamma_k \left(\frac{4\sqrt[3]{4}}{3}\right)^{k} \left(\frac{r}{k}\right)^{k} + O(r^{k-1}), \end{aligned}$$

where $\gamma_k$ is as defined in the theorem. The proof of Theorem 4.1 is complete. □

We end this section with the following problem. Is it true that, for any fixed integer $r \geq 1$,

$$\lim_{k\to\infty} \frac{m^*(r,k)}{n(r,k)} = 1,$$

where $n(r,k)$ denotes the greatest positive integer $m$ such that there exists a subset $A$ of $\mathbf{Z}_m$ for which the diameter $d(m,A)$ of the Cayley digraph $\mathrm{Cay}(m,A)$ is no more than $r$. This is clearly true for $r = 1$. For more results about $n(r,k)$, see [10], [13], [18], and [12].

## 5. Minimal average distance of distributed loop networks.

Distributed loop networks that are optimal or nearly optimal with respect to their diameters have been studied extensively (see a survey by Bermond, Comellas, and Hsu [3]). Since the average distance of a network reflects its global performance, the investigation of loop networks that are optimal with respect to their average distances is of great interest and importance.

It is easy to see that

$$r(m,1) = \frac{m-1}{2}.$$

For $k = 2$, Wong and Coppersmith [22] proved that

$$\frac{5}{3\sqrt{3}}\sqrt{m} - 1 \leq r(m,2) \leq \sqrt{m} - 1,$$

where the lower bound holds for all $m$, whereas the upper bound holds for an infinite family of positive integers $m$. Theorem 2.1 leads to an infinite family of positive integers $m$ such that

$$r(m,2) = \frac{5}{3\sqrt{3}}(1 + o(1))\sqrt{m}.$$

This infinite class of directed double loop networks is asymptotically optimal with respect to average distances. In fact, we can construct several other infinite classes of asymptotically optimal double loop networks. For $k \geq 3$, Wong and Coppersmith proved that

$$\frac{k}{k+1}\sqrt[k]{k!m} \leq r(m,k) \leq \frac{k}{2}\sqrt[k]{m} - \frac{k}{2},$$

where the lower bound holds for all $m$, whereas the upper bound holds for an infinite family of positive integers $m$. Theorem 4.1 implies the following.

THEOREM 5.1. *Let $k \geq 1$ be an integer. Then there exists an infinite class of positive integers $m$, for which*

$$r(m,k) \leq (1+o(1))\frac{3k}{4\sqrt[3]{4}}\sqrt[k]{\delta_k m} \approx 0.472k\sqrt[k]{\delta_k m},$$

*where*

$$\delta_k = \frac{1}{\gamma_k} = \begin{cases} 1 & \textit{if } k \equiv 0 \pmod 3, \\ \dfrac{2\sqrt[3]{4}}{3} \approx 1.058 & \textit{if } k \equiv 1 \pmod 3, \\ \dfrac{200\sqrt[3]{2}}{243} \approx 1.037 & \textit{if } k \equiv 2 \pmod 3. \end{cases}$$

This improves the lower bound of Wong and Coppersmith. Moreover, from the proofs of Lemma 4.2 and Theorems 3.1 and 4.1, we have the precise construction of the basis corresponding to each $m$ in the infinite class. We list, to each $1 \leq k \leq 5$, the infinite class of integers $\{m_t\}$, the corresponding generating set $A_t$, and the average distance $r(m_t, A_t)$

$k = 1$: $m_t = t$ for $t = 1, 2, \ldots$,
$A_t = \{1\}$ for $t = 1, 2, \ldots$,
$r(m_t, A_t) = \frac{t-1}{2}$;
$k = 2$: $m_t \leq 3t^2 + t$ for $t = 1, 2, \ldots$,
$A_t = \{1, 3t\}$ for $t = 1, 2, \ldots$,
$r(m_t, A_t) = \frac{5t}{3}$;
$k = 3$: $m_t = 4t^3 + t^2 + 2t$ for $t = 1, 2, \ldots$,
$A_t = \{1, 4t, 4t^2 + t\}$ for $t = 1, 2, \ldots$,
$r(m_t, A_t) = \frac{9t}{4}$;
$k = 4$: $m_t = 4t^4 + t^3 + 2t^2$ for $t = 1, 2, \ldots$,
$A_t = \{1, t, 4t^2, 4t^3 + t^2\}$ for $t = 1, 2, \ldots$,
$r(m_t, A_t) = \frac{11t-2}{4}$;
$k = 5$: $m_t = 12t^5 + 7t^4 + 7t^3 + 2t^2$ for $t = 1, 2, \ldots$,
$A_t = \{1, 3t, 3t^2 + t, 12t^3 + 4t^2, 12t^4 + 7t^3 + t^2\}$ for $t = 1, 2, \ldots$,
$r(m_t, A_t) = \frac{47t}{12}$.

From our proofs, it is easy to construct many other infinite classes of integers and corresponding generating sets that give similar average distances. For $k \geq 2$, it is of great interest to find an infinite class of optimal loop networks with respect to their average distance, that is, to find an infinite class of positive integers $m_t$ and $k$-element–generating sets $A_t$ such that

$$r(m_t, k) = r(m_t, A_t) \quad \text{for all } t.$$

**6. Average order of subsets of asymptotic bases.** Theorems 2.1, 3.1, and 4.1 can be applied to a problem in number theory concerning the average order of subsets of asymptotic bases. A set $A$ of nonnegative integers is called an *asymptotic basis of order $h$* if a very large integer is a sum of at most $h$ elements in $A$. Let $g(A)$ denote the least such positive integer $h$. For any $h \geq 2$ and $k \geq 1$, define

$$G_k(h) = \max_{g(A)\leq h} \max_{\substack{|F|=k, F\subset A \\ g(A\backslash F)<\infty}} g(A\backslash F).$$

This extremal function has been studied extensively by many authors (see, for instance, Erdös and Graham [7], Nathanson [20], Nash[19], Jia [16], [17], and Chen and Gu [6]). In this section, we consider the analogue problem with respect to the average order.

Let $A$ be an asymptotic basis.[2] The average order $r(A)$ of $A$ is defined as follows:

$$r(A) = \limsup_{m\to\infty} \frac{1}{m} \sum_{n=N_0}^{m+N_0} f(A,n),$$

where $N_0$ is such that every integer $n \geq N_0$ is a sum of elements in $A$.

Let $A$ be an asymptotic basis of average order $r$ and $F$ a subset of $A$. We are interested in the growth of the average order $r(A\backslash F)$ of $A\backslash F$ in terms of $r$, the average order of $A$. Let $r \geq 1$ be a real number and $k$ a positive integer. Define

$$G_k^*(r) = \max_{r(A)\leq r} \max_{\substack{|F|=k, F\subset A \\ r(A\backslash F)<\infty}} r(A\backslash F).$$

THEOREM 6.1. *For any integer $k \geq 1$,*

$$G_k^*(r) \geq \lambda_k \left(\frac{4\sqrt[3]{4}}{3}\right)^k \left(\frac{r}{k+1}\right)^{k+1} + O(r^k),$$

*where*

$$\lambda_k = \begin{cases} \dfrac{2\sqrt[3]{4}}{3} \approx 1.058 & \text{if } k \equiv 0 \pmod 3, \\ 1 & \text{if } k \equiv 1 \pmod 3, \\ \dfrac{81\sqrt[3]{2}}{100} \approx 1.021 & \text{if } k \equiv 2 \pmod 3. \end{cases}$$

*In particular, we have*

$$G_1^*(r) \geq \frac{27}{50} r^2 + O(r).$$

Given a real number $r \geq 1$ and an integer $k \geq 1$, let $m_1^*(r,k)$ denote the greatest positive integer $m$ such that the average distance of the Cayley digraph $\mathrm{Cay}(m, A)$ is no more than $r$ for some set $A = \{a_1 = 1, a_2, \ldots, a_k\}$ of integers. It is clear that

[2] In fact, $A$ is not necessarily an asymptotic basis. The only condition we need is that every large positive integer is a finite sum of elements in $A$.

$m_1^*(r,k) \le m^*(r,k)$ for any $r$ and $k$. From the proof of Theorem 4.1, the lower bound for $m^*(r,k)$ is also valid for $m_1^*(r,k)$. Therefore, Theorem 6.1 follows immediately from the following theorem and Theorem 4.1.

THEOREM 6.2. *For any integers $r \ge 2$ and $k \ge 1$,*

$$G_k^*(r) \ge \frac{1}{2} m_1^*(r-1, k+1).$$

*Proof.* Let $m = m_1^*(r-1,k+1)$. Let $A_{k+1} = \{a_0 = 1 < a_1 < \cdots < a_k\}$ be a set such that $m^*(r-1, A_{k+1}) = m^*(r-1,k+1)$.

Define $F = \{a_1, \ldots, a_k\}$, and $A = \{1, im \mid i = 0, 1, \ldots\} \cup F$. It is clear that the average order of $A \backslash F$ is $\frac{m}{2}$. Therefore, it suffices to show that the average order $r(A) \le r$.

If $q$ is sufficiently large, it is clear that $n \in [qm, (q+1)m)$ has representation with the least number of elements

$$n = pm + a_{i_1} + \cdots + a_{i_t}$$

if and only if

$$n \equiv a_{i_1} + \cdots + a_{i_t} \pmod{m}$$

is a representation of $n \in \mathbf{Z}_m$ as a sum of elements in $A_{k+1}$ with the least number of summands. Since $f(A, pm+n) = f(A, qm+n)$ if $p, q$ are sufficiently large, we see that, for some $q$ sufficiently large,

$$\begin{aligned} r(A) &= \frac{1}{m}\sum_{n=0}^{m-1} f(A, qm+n) = 1 + \frac{1}{m}\sum_{n=0}^{m-1} f(A_{k+1}, n) \\ &= 1 + r(m, A_{k+1}) \le 1 + (r-1) = r. \end{aligned}$$

The proof is complete. $\square$

## REFERENCES

[1] S. B. AKERS AND B. KRISHNAMURTHY, *A group-theoretic model for symmetric interconnection networks*, IEEE Trans. Comput., 38(1989), p. 555.

[2] R. BEIVIDE, E. HERRADA, AND J. L. BALCAZAR, *Optimal distance networks of low degree for parallel computers*, IEEE Trans. Comput., 40(1991) p. 1109.

[3] J.-C. BERMOND, F. COMELLAS, AND D. F. HSU, *Distributed loop computer networks: A survey*, to appear.

[4] J.-C. BERMOND, A. ILLIADES, AND M. PEYRAT, *An optimization problem in distributed loop computer networks*, in 3rd Int. Conf. on Combinatorial Math., 1987.

[5] F. T. BOESCH AND J.-F. WANG, *Reliable circulant networks with minimum transmission delay*, IEEE Trans. Circuits Syst., CAS-32, 1985.

[6] S. CHEN AND W. GU, *Exact order of subsets of asymptotic bases*, J. Number Theory, 41(1992), pp. 15-21.

[7] P. ERDÖS AND R. L. GRAHAM, *On bases with an exact order*, Acta Arith., 37(1980), pp. 201–207.

[8] P. ERDÖS AND D. F. HSU, *Distributed loop networks with minimum transmission delay*, Theoret. Comput. Sci., to appear.

[9] M. A. FIOL, J. L. A. YEBRA, I. ALEGRE, AND M. VALERO, *A discrete optimization problem in local networks and data alignment*, IEEE Trans. Comput., C-36(1987), pp. 702–713.

[10] R. L. GRAHAM AND N. J. A. SLOANE, *On additive bases and Harmonious graphs*, SIAM J. Alg. Discrete Meth., 1(1980), pp. 382–404.

[11] G. GREKOS, *Quelques Aspects de la Théorie Additive des Nombres*, Ph.D. thesis, Université de Bordeaux I, 1982.

[12] D. F. HSU AND X.-D. JIA, *Additive bases and graph labelling problems*, a survey, preprint.

[13] ———, *Extremal problems in the construction of distributed loop networks*, SIAM J. Discrete Math., 7(1994), pp. 57–71.

[14] D. F. HSU AND J. SHAPIRO, *A census of tight one-optimal double loop networks*, Networks, to appear.

[15] ———, *Bounds for the minimal number of transmission delays in double loop networks*, J. Combin. Inform. System Sci., 16(1991), p. 55.

[16] X.-D. JIA, *Exact order of subsets of asymptotic bases in additive number theory*, J. Number Theory, 28(1988), pp. 205–218.

[17] ———, *On the order of subsets of asymptotic bases*, J. Number Theory, 37(1991), pp. 37–46.

[18] ———, *Extremal bases for finite cyclic groups*, J. Number Theory, 41(1992), pp. 116–127.

[19] J. C. M. NASH, *Results on Bases in Additive Number Theory*, thesis, Rutgers University, New Brunswick, NJ, 1985.

[20] M. B. NATHANSON, *The exact order of subsets of additive bases*, in Proc., Number Theory Seminar, 1982, Lecture Notes in Mathematics, 1052, Springer-Verlag, 1984, pp. 273–277.

[21] W. SU, *A combinatorial problem in the construction of distributed loop networks*, M. S. thesis, Southwest Texas State University, San Marcos, TX, 1993.

[22] C. K. WONG AND D. COPPERSMITH, *A combinatorial problem related to multimodule memory organization*, J. Assoc. Comput. Mach., 21(1974), pp. 392–402.

SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 76–92, February 1995

# FRACTIONAL COVERS AND COMMUNICATION COMPLEXITY*

MAURICIO KARCHMER[†], EYAL KUSHILEVITZ[‡], AND NOAM NISAN[§]

**Abstract.** It is possible to view communication complexity as the minimum solution of an *integer programming* problem. This integer programming problem is relaxed to a *linear programming* problem and from it information regarding the original communication complexity question is deduced. A particularly appealing avenue this opens is the possibility of proving *lower* bounds on the communication complexity (which is a minimization problem) by exhibiting *upper* bounds on the maximization problem defined by the *dual* of the linear program.

This approach works very neatly in the case of nondeterministic communication complexity. In this case a special case of Lovász's fractional cover measure is obtained. Through it the *amortized* nondeterministic communication complexity is completely characterized. The power of the approach is also illustrated by proving lower and upper bounds on the nondeterministic communication complexity of various functions.

In the case of deterministic complexity the situation is more complicated. Two attempts are discussed and some results using each of them are obtaied. The main result regarding the first attempt is negative: one cannot use this method for proving superpolynomial lower bounds for formula size. The main result regarding the second attempt is a "direct-sum" theorem for two-round communication complexity.

**Key words.** communication complexity, linear programming bound

**AMS subject classifications.** 68, 94A15, 94A29, 94A49

## 1. Introduction.

Many combinatorial optimization problems can be expressed as integer programming problems. Relaxing an integer programming problem to a linear programming problem often gives useful information regarding the original one. In this paper we apply this technique to the study of *communication complexity.*

We consider communication complexity in the wide context of computing relations: we have two players $P_1$ and $P_2$, holding $n$-bit input strings, $x$ and $y$ respectively. They wish to find a value $z$ satisfying a relation $R(x, y, z)$.[1] The goal of the players is to communicate as few bits as possible. This general communication complexity problem contains as special cases the communication complexity of functions, as defined by Yao [Y79] (and studied in numerous works later on), and the relations defined by Karchmer and Wigderson [KW88], which are important because of their close relationship with boolean circuit depth.

It is convenient to count the number of different histories of the protocol. It is well known (see [K89]) that the logarithm of this quantity is equal (up to a constant factor) to the communication complexity. In the case of relations corresponding to circuit depth of boolean functions, this measure gives exactly the formula size. We

* Received by the editors October 13, 1992; accepted for publication (in revised form) December 27, 1993. A preliminary version of this paper appeared as an invited paper in *Proc. 7th IEEE Structure in Complexity Theory*, June 1992, pp. 262–274.

[†] Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (mauricio@math.mit.edu). This author's research was supported by National Science Foundation grant NSF-CCR-90-10533.

[‡] Department of Computer Science, Technion, Haifa 32000, Israel (eyalk@cs.technion.ac.il). This author's research was done while the author was at Aiken Computation Lab., Harvard University, and was supported by National Science Foundation grant NSF-CCR-90-07677.

[§] Department of Computer Science, Hebrew University, Jerusalem 91904, Israel (noam@cs.huji.ac.il). This author's research was supported by the Wolfson Research awards administered by the Israel Academy of Sciences and Humanities and by USA–Israel BSF 89-00126.

[1] We assume that such $z$ always exists. Alternatively, we can say that if there is no such $z$ then any output is legal.

may thus view the communication problem as a covering problem: cover the whole space of possible inputs by possible histories. As an integer programming problem this becomes the following: assign 0-1 weights to the possible histories of a communication protocol such that each possible input is covered with weight 1.

We formalize this integer programming problem and then study the linear programming relaxation of it. Two of the most intriguing features of this approach are

- It allows one to study the *dual* linear programming problem. In particular, one can give *lower bounds* to the original problems by providing *upper bounds* to their dual problems.
- It turns out that the linear programming relaxation often has "direct sum" properties; i.e., the complexity of solving two independent problems simultaneously is exactly equal to the sum of the separate complexities. These results then imply similar results for the original complexity measure.

In this paper we study three different formalizations and relaxations. The first formalization deals with the *nondeterministic case.* It is presented first since it is the most elegant and successful case. Two formalizations for the *deterministic* case are also presented, neither of them without problems.

Our first formalization, for nondeterministic communication complexity, is studied in §2. The main results we obtain in this case are

- The linear relaxation gives exactly Lovász's "fractional cover" measure [L75]. On the other hand it has a natural interpretation in communication complexity terms.
- The linear relaxation is always very close to the "true" nondeterministic communication complexity.
- We get direct-sum results for nondeterministic communication complexity, re-proving and strengthening the recent results of [FKN91]. In particular we show that the linear relaxation completely characterizes the *amortized* nondeterministic communication complexity.
- Various known upper and lower bounds are given new simple proofs using the linear programming relaxations.
- Some connections are shown with the private-coins vs. public-coins question in randomized communication complexity.

Our second formalization, studied in §3, considers the "natural" approach to describing deterministic communication complexity as integer and then linear programs. Our main concern in this section is with communication complexity of relations that correspond to boolean circuit depth and formula size (as in [KW88]). The main results we obtain using this approach are

- We give a surprising new proof of Khrapchenko's quadratic lower bound [K71] on the formula size of the parity function.
- We show that this approach *cannot* give superquadratic lower bounds for the formula size of any boolean function. In particular, the solution of the integer program may be vastly different from the solution of the linear program.
- We give some indication that for *monotone* circuit depth and formula size, this approach may yield exponential lower bounds.

The basic failure of the "natural" approach to deterministic communication complexity led us to consider the third formalization, discussed in §4. This formalization uses a round-by-round approach to communication complexity, and we were only able to obtain results for one-round and two-round protocols. The main results we obtain are

- Direct-sum results for deterministic one-round and two-round communication complexity.
- En route we generalized some of Lovász's results regarding fractional covers to the weighted case, results that may be of independent interest. (Generalizations of some of the results were already known [C79].)

The three different sections of this paper are technically nearly independent of each other. Each section contains an introduction which describes the formalization studied in the section and mentions the basic results obtained.

## 2. Nondeterministic complexity.

**2.1. Introduction.** In the nondeterministic model of communication complexity the two players may act nondeterministically, but once they reach an answer, they must be sure of its correctness. It is well known that the nondeterministic communication complexity of a relation $R$, denoted $C_N(R)$, is simply the logarithm (base 2) of the number of monochromatic rectangles needed to cover the matrix associated with the function.

DEFINITION 2.1. *Given a relation $R \subseteq \{0,1\}^n \times \{0,1\}^n \times Z$ we denote by $M_R$ the matrix representing this relation. That is, each row of $M_R$ corresponds to an input $x$ of $P_1$, and each column corresponds to an input $y$ of $P_2$. The entry $(x, y)$ contains the set of all $z$'s that satisfy $R(x, y, z)$. A* rectangle *of $M_R$ is a submatrix of the form $A \times B$ where $A, B \subseteq \{0,1\}^n$. A rectangle $A \times B$ is called* monochromatic *if there exists some element $z$ which is a member of all entries of the rectangle.*

DEFINITION 2.2. *The* nondeterministic cover number *of a relation $R$, denoted $N(R)$, is the minimum number of monochromatic rectangles that cover $M_R$, allowing overlaps.*

We associate with every relation $R$ a hypergraph $H_R = (V, E)$ as follows. The vertices of $H_R$ are all possible inputs (i.e., $V = \{0,1\}^n \times \{0,1\}^n$). The hyperedges are all monochromatic rectangles. We can write the nondeterministic cover number as an integer programming problem. Let $R$ be a relation, and let $H_R = (V, E)$ be the corresponding hypergraph. A nondeterministic cover of $R$ can be viewed as a *boolean* function $\phi : E \longrightarrow \{0,1\}$, such that

$$\forall v \in V : \sum_{e \in E\ :\ v \in e} \phi(e) \geq 1.$$

The cover number $N(R)$ is defined as $\min_\phi \sum_{e \in E} \phi(e)$ where $\phi$ ranges over all nondeterministic covers of $H_R$.[2] We now define the relaxation of $N(R)$.

DEFINITION 2.3. *A* nondeterministic fractional cover *of $H_R$ is a* real *function $\phi : E \longrightarrow [0,1]$, such that*

$$\forall v \in V : \sum_{e \in E\ :\ v \in e} \phi(e) \geq 1.$$

*The* fractional cover number *of $R$, denoted $N^*(R)$, is defined as $\min_\phi \sum_{e \in E} \phi(e)$ where $\phi$ ranges over all nondeterministic fractional covers of $H_R$.*

[2] Note that in fact all the definitions of the various cover numbers do not make any use of the special structure of the hypergraphs of the form $H_R$, and therefore can be generalized to any hypergraph (as in [L75]). For making the exposition more clear we concentrate on hypergraphs of the form $H_R$. In the technical part of this section we will be interested in the cover numbers of other hypergraphs as well.

As mentioned, this definition is just a special case of Lovász's definition of fractional covers [L75], and therefore we may apply his more general results. In particular, Lovász shows (see Theorem 2.6 below) that the fractional cover number $N^*$ can never be much smaller than the cover number $N$ (clearly, $N^*(R) \leq N(R)$, for every $R$). Thus, the linear program will give us much information regarding the original nondeterministic communication complexity problem. We use this approach to obtain some very simple proofs of (basically known) *upper* and *lower* bounds to nondeterministic communication complexity.

We now give a simple interpretation for the fractional cover number in the case of communication complexity: a simple way to give a lower bound to $N(R)$ is to give an upper bound to the size of any monochromatic rectangle. This can of course be done relative to any distribution $P$ on $X \times Y$: let $\text{Bound}_P(R) = \max_e Pr_P(e)$, where $e$ ranges over all monochromatic rectangles of $R$. It is clear that for any distribution $P$, $1/\text{Bound}_P(R)$ is a lower bound for $N(R)$. It turns out that the best bound one can obtain this way is exactly $N^*(R)$.

LEMMA 2.4. *$N^*(R) = \max_P \frac{1}{\text{Bound}_P(R)}$, where $P$ ranges over all probability distributions on $X \times Y$.*

*Proof.* By the same argument as above, $N^*(R) \geq \frac{1}{\text{Bound}_P(R)}$, for every $P$. Therefore one direction follows immediately. For the second direction we use the primal-dual theorem for linear programming. The dual of the linear program defining $N^*(R)$ is

$$\max_{\xi} \sum_{x,y} \xi(x,y),$$

where $\xi$ is any real function $\xi : V \to [0,1]$, such that

$$\forall e \in E \quad \sum_{(x,y)\in e} \xi(x,y) \leq 1.$$

The lemma can be verified by associating with every $\xi$ a distribution $P_\xi$, $P_\xi(x,y) = \xi(x,y)/\sum_{x',y'} \xi(x',y')$. □

The main result we obtain regarding the fractional cover number is that this measure captures completely the cost of solving simultaneously several problems on independent inputs. In particular, we show that $N^*$ is multiplicative with respect to the direct sum.

THEOREM. *Let $R, R_1, \ldots, R_k$ be arbitrary relations. Then,*

- $\log N^*(R) \leq C_N(R) \leq \log N^*(R) + O(\log n)$,
- $\prod_i N^*(R_i) = N^*(R_1 \times \cdots \times R_k)$

These results immediately imply the "direct sum" results in [FKN91]. In fact the following corollary gives the underlying reason for these "direct sum" results. Let $k$ be an integer and let $R^k$ denote the "direct sum" of $R$ $k$ times. Namely, to compute $R^k$ we need to compute simultaneously $R$ on $k$ independent inputs. Denote by $\tilde{C}_N(R)$ the *amortized communication complexity* of $R$ [FKN91], i.e., $\limsup_{k\to\infty}, C_N(R^k)/k$.

COROLLARY. *Let $R$ be any relation. Then, $\tilde{C}_N(R) = \log N^*(R) \geq C_N(R) - O(\log n)$.*

A similar theorem holds for the "one-sided" version of nondeterministic complexity of boolean *functions*, where the players have to be sure about the output only if they output 1. In this case, the problem is to cover the 1's of the function using 1-monochromatic rectangles. (We denote by $C_{NP}(f), \tilde{C}_{NP}(f)$, and $NP^*(f)$ the analogues of $C_N(f), \tilde{C}_N(f)$, and $N^*(f)$, for this case.)

To demonstrate the tightness of our results we exhibit a simple function which has a large gap between $NP(f)$ and $NP^*(f)$, and thus also between $C_{NP}(f)$ and $\tilde{C}_{NP}(f)$. Let $NE(x,y)$ be the nonequality function giving "1" iff the $n$-bit strings $x$ and $y$ are not equal. We show

$$C_{NP}(NE) = \Theta(\log n) \text{ but } \tilde{C}_{NP}(NE) = \Theta(1).$$

It is interesting to note that for this function the complexity difference between $C_{NP}(f)$ and $\tilde{C}_{NP}(f)$ mirrors the complexity difference between the "private-coins" and "public-coins" variants of *randomized* complexity. We explain this phenomena (that randomization in the public-coins model is more powerful than nondeterminism) and prove that while the (one-sided error) randomized complexity in the public-coins model can be smaller than $C_{NP}(f)$, it is always at least $\tilde{C}_{NP}(f) = \log NP^*(f)$.

**2.2. Direct sums.** We start by defining the product of two hypergraphs.

DEFINITION 2.5. *Given two hypergraphs $H_1$ and $H_2$ we define their* product $H_1 \times H_2$ *by* $V(H_1 \times H_2) = V(H_1) \times V(H_2)$ *and* $E(H_1 \times H_2) = \{e_1 \times e_2 | e_1 \in E(H_1), e_2 \in E(H_2)\}$.

The following result of Lovász is crucial.

THEOREM 2.6 ([L75]). *Let $H, H_1$, and $H_2$ be any hypergraphs then*

1. $N^*(H) \geq \frac{N(H)}{\ln |V(H)|}$;
2. $N^*(H_1 \times H_2) = N^*(H_1) \cdot N^*(H_2)$.

The first statement directly yields the following corollary.

COROLLARY 2.7. *Let $R$ be any relation. Then,* $\log N^*(R) \leq C_N(R) \leq \log N^*(R) + \log n + O(1)$.

DEFINITION 2.8. *Given two relations $R$ and $S$, their* direct sum, *denoted $R \times S$, is the problem of solving both $R$ and $S$ simultaneously on independent inputs.*

Note that usually, for two relations $R$ and $S$, the hypergraph $H_{R\times S}$ is not the same as the hypergraph obtained by the product $H_R \times H_S$. However, the following lemma claims that both have the same nondeterministic fractional cover number.

LEMMA 2.9. *Let $R$ and $S$ be two relations. Then $N^*(H_{R\times S}) = N^*(H_R \times H_S)$.*

*Proof.* For proving that $N^*(H_{R\times S}) \leq N^*(H_R \times H_S)$, note that if $e_R$ is a monochromatic rectangle of $M_R$ and $e_S$ is a monochromatic rectangle of $M_S$, then $e_R \times e_S$ is a monochromatic rectangle of $M_{R\times S}$. Therefore, $E(H_R \times H_S) \subseteq E(H_{R\times S})$. This implies that every nondeterministic fractional cover $\phi$ defined for $H_R \times H_S$ can be extended with zeroes to a nondeterministic fractional cover of $H_{R\times S}$, and thus the inequality follows.

On the other hand, given the optimal nondeterministic fractional cover $\phi$ defined on $E(H_{R\times S})$, we can take every hyperedge $e = X \times Y \subseteq V(H_R) \times V(H_S)$ with $\phi(e) > 0$, and define $X_R, X_S, Y_R$, and $Y_S$ to be the projections of $X$ and $Y$ on the first and second coordinates respectively (i.e., the projections on $V(H_R)$ and $V(H_S)$ respectively). Now, define $e_R = X_R \times Y_R$ and $e_S = X_S \times Y_S$. These are monochromatic rectangles of $M_R$ and $M_S$ (respectively) and thus $e_R \times e_S$ is a hyperedge of $H_R \times H_S$. Define, for every $e$, $\phi'(e_R \times e_S) = \phi(e)$ (if more than one hyperedge correspond to the same $e_R, e_S$ then $\phi'(e_R \times e_S)$ is the sum of $\phi(e)$ for all those $e$'s). We get that $\phi'$ is a nondeterministic fractional cover of $H_R \times H_S$ (since the monochromatic rectangle $e_R \times e_S$ contains the monochromatic rectangle $e$) and therefore $N^*(H_R \times H_S) \leq N^*(H_{R\times S})$. □

We can now get the following set of "direct sum" results.

THEOREM 2.10. *Let $R, R_1, \ldots, R_k$ be the arbitrary relations:*

- $N^*(\times_{i=1}^k R_i) = \prod_{i=1}^k N^*(R_i)$;
- $\sum_{i=1}^k \log N^*(R_i) \le C_N(\times_{i=1}^k R_i) \le \sum_{i=1}^k \log N^*(R_i) + \log kn + O(1)$;
- $\tilde{C}_N(R) = \log N^*(R)$.

*Proof.* The lower bounds on $C_N$ follow from Theorem 2.6 and Lemma 2.9. The upper bounds follow from Theorem 2.6 and Corollary 2.7. The bound for $\tilde{C}_N$ is obtained by taking $k$ copies of $R$ and letting $k$ approach infinity. □

By using Corollary 2.7 we can eliminate $N^*$ from the statement, getting as a corollary the somewhat weaker results of [FKN91].

COROLLARY 2.11. *Let $R, R_1, \ldots, R_k$ be the arbitrary relations:*

- $\sum_{i=1}^k C_N(R_i) - k\log n - O(1) \le C_N(\times_{i=1}^k R_i) \le \sum_{i=1}^k C_N(R_i)$
- $C_N(R) - \log n - O(1) \le \tilde{C}_N(R) \le C_N(R)$

**2.3. One-sided nondeterministic complexity.** In the case that boolean functions are computed, one is frequently only interested in the "NP"-version of nondeterministic complexity, i.e., where the players need only be sure of the answer in the case where $f(x,y) = 1$. We denote this complexity by $C_{NP}(f)$. It is not difficult to see that the corresponding covering problem is simply to cover all the 1-inputs of $f$ by 1-monochromatic rectangles.

It is straightforward to carry over all of our results to this case as well, where in the direct sum of $f$ and $g$, we need only cover the joint 1's of $f$ and $g$, i.e., cover the 1's of $f \wedge g$. In particular we get the following corollaries.

THEOREM 2.12. *Let $f_1, \ldots, f_k$ be any $k$ functions. Then*

$$\sum_{i=1}^k C_{NP}(f_i) - k\log n - O(1) \le C_{NP}(f_1 \wedge \cdots \wedge f_k) \le \sum_{i=1}^k C_{NP}(f_i).$$

The following example shows how the above results can be used for proving lower bounds on the nondeterministic communication complexity.

*Example* 1. Let the "disjointness" function be defined as follows: $\mathrm{DISJ}_n(x,y)$ is defined for every $x, y \in \{0,1\}^n$ as 1 if there is no index $i$ such that $x_i = y_i = 1$ (and 0 otherwise). Clearly,

$$\mathrm{DISJ}_n(x,y) = \bigwedge_{i=1}^n \mathrm{DISJ}_1(x_i, y_i).$$

Therefore, $NP(\mathrm{DISJ}_n) \ge NP^*(\mathrm{DISJ}_n) = (NP^*(\mathrm{DISJ}_1))^n = 2^n$, where the last equality follows by noting that $NP^*(\mathrm{DISJ}_1) = 2$. Thus we have

$$C_N(\mathrm{DISJ}_n) = C_{NP}(\mathrm{DISJ}_n) = n.$$

**2.4. Fractional covers and randomized complexity.** The following theorem relates $NP^*(f)$ to $C_{R\text{-pub}}(f)$—the communication complexity of computing $f$ by a *probabilistic one-side error protocol* (i.e., a protocol that might err only if $f(x,y) = 1$ with probability smaller than, say, $\frac{1}{2}$) in the *public coins* model.[3] It is known that $C_{R\text{-pub}}(f)$ is smaller than $C_{R\text{-priv}}(f)$ (one-sided error protocols in the *private-coins* model) by at most an additive factor of $\log n$. Clearly, $C_{NP}(f) \le C_{R\text{-priv}}(f)$.[4] We

[3] In the public coins model, instead of flipping coins locally, the two parties share a string of random coins. For a formal definition of the model and some results on the relations between the public coins and private coins models, see [N91].

[4] The parties "guess" good random coins and run the randomized protocol.

already proved that $\log NP^*(f)$ is smaller than $C_{NP}(f)$ by at most an additive term of $\log n$. To complete the picture we give the following theorem.

THEOREM 2.13. *Let $f$ be a function. Then* $\log NP^*(f) \leq C_{R\text{-pub}}(f) + 1$.

*Proof.* Given $F$, a probabilistic one-sided error protocol for $f$ in the public-coins model, we will construct a fractional cover for the 1's of $M_f$, as needed. Let $r$ be a possible (public) random string and let $p(r)$ be its probability. Fixing $r$, then $F$ is just a deterministic protocol and therefore induces a cover of the 1's of $M_f$ by at most $2^{C_{R\text{-pub}}(f)}$ monochromatic rectangles. We add to the cover all the rectangles in which the output is "1." As the protocol has only one-sided errors then these rectangles cover only "1"-entries. With each such rectangle $e$ we associate a value $\phi(e) = 2p(r)$. We repeat this process for every possible random string $r$. We claim that the obtained cover is what we aim for. First, note that for every $(x, y)$ such that $f(x, y) = 1$ we have

$$\sum_{e:\ (x,y)\in e} \phi(e) = 2 \cdot \text{Prob}(F \text{ outputs 1 on input } (x, y)) \geq 2 \cdot \frac{1}{2} = 1.$$

Finally, note that the cover we construct satisfies

$$\sum_{e} \phi(e) \leq \sum_{r} 2p(r) \cdot 2^{C_{R\text{-pub}}(f)} = 2 \cdot 2^{C_{R\text{-pub}}(f)}. \qquad \square$$

We now show that the above theorem can be used to estimate $NP^*(f)$.

*Example* 2. Let the "nonequality" function be defined as follows: $NE_n(x, y)$ is defined for every $x, y \in \{0,1\}^n$ as 1 if $x \neq y$ and 0 otherwise. It is known that $C_D(NE_n) = n$, and that $C_N(NE_n) = \Theta(\log n)$.[5] On the other hand, $C_{R\text{-pub}}(NE_n) = O(1)$.[6] By Theorem 2.13, we get that $NP^*(NE_n) = O(1)$. (Note that for this function $C_{R\text{-pub}}$ is less than $NP$.)

In the following example we show how to use these techniques to derive nontrivial *upper* bounds on the nondeterministic communication complexity. Interestingly, this is done without describing explicitly protocols that compute the functions.

*Example* 3. Let $n$ be a perfect square. Let $f_n$ be the following function: view each input string as $\sqrt{n}$ substrings of length $\sqrt{n}$ (i.e., $x = \overline{x}_1\overline{x}_2 \ldots \overline{x}_{\sqrt{n}}$ and $y = \overline{y}_1\overline{y}_2 \ldots \overline{y}_{\sqrt{n}}$, where $\overline{x}_i, \overline{y}_i \in \{0,1\}^{\sqrt{n}}$, for every $i$). Let $f_n$ be defined as follows: $f_n(x, y)$ is 1 if there exists an $i$ such that $\overline{x}_i = \overline{y}_i$. This function was studied in [MS82], [F87], and a (tight) $O(\sqrt{n})$ upper bound was proved for its nondeterministic communication complexity, using a complex protocol. Here we give a very simple proof for this upper bound. Clearly, $C_{NP}(f_n) = O(\sqrt{n})$.[7] Therefore to prove that $C_N(f_n) = O(\sqrt{n})$ it is enough to prove that $C_{NP}(\overline{f}_n) = O(\sqrt{n})$.[8] For this, we write $\overline{f}_n(x) = \bigwedge_{i=1}^{\sqrt{n}} NE_{\sqrt{n}}(\overline{x}_i, \overline{y}_i)$. Therefore, $NP^*(\overline{f}_n) = (NP^*(NE_{\sqrt{n}}))^{\sqrt{n}}$ which equals by the previous example to $(O(1))^{\sqrt{n}}$. This implies that $C_{NP}(\overline{f}_n) = O(\sqrt{n})$.

[5] $P_1$ "guesses" an index $i$ and sends the index $i$ together with $x_i$ to $P_2$.

[6] The parties can view the public random string as a $n$-bit vector $b$ and exchange the inner product of $b$ with $x$ and $y$.

[7] $P_1$ "guesses" $i$ and sends $i$ and $\overline{x}_i$ to $P_2$ who checks whether $\overline{x}_i = \overline{y}_i$.

[8] The trivial upper bound for $C_{NP}(\overline{f}_n)$ is $O(\sqrt{n}\log n)$: $P_1$ "guesses" for every $1 \leq i \leq \sqrt{n}$ an index in which $\overline{x}_i$ differs from $\overline{y}_i$. It sends to $P_2$ all those indices with their values.

## 3. Deterministic complexity: disjoint cover.

**3.1. Introduction.** As shown, our approach works well for *nondeterministic* complexity. In the case of computing *functions* we do get some nontrivial information regarding *deterministic* complexity, as [AUY83] showed that there can be at most a quadratic gap between deterministic and nondeterministic complexity. In the case of *relations* we may get no information at all, as the gap between deterministic and nondeterministic complexity can be exponential. However, we will show that the suggested approach leads to some results.

A natural approach to present deterministic communication complexity as a covering problem is simply to forbid overlap of any two rectangles in the monochromatic cover.

DEFINITION 3.1. *The* deterministic cover number *of a relation $R$, denoted $D(R)$, is the minimum number of monochromatic rectangles in a* disjoint *(nonoverlapping) cover of the set of inputs.*

As opposed to the nondeterministic case where the nondeterministic complexity, $C_N(R)$, was always $\Theta(\log N(R))$, it is still an open problem whether the deterministic complexity, $C_D(R)$, is always $\Theta(\log D(R))$. However, it is still true that $\log D(R) \leq C_D(R)$. Furthermore, it is implicit in [AUY83] that $\log D(R) \geq \sqrt{C_D(R)}$, and thus these two measures are quite close. Let us also mention that if $R_g$ is a relation associated with the circuit depth of a boolean function $g$ (à la [KW88]) then $D(R_g)$ yields a lower bound to the formula size complexity.

Therefore it is important to understand the measure $D(R)$. This measure has the advantage of being more combinatorial than $C_D(R)$. As previously, we can express $D(R)$ as an integer program. For a relation $R$, let $H_R = (V, E)$ be the corresponding hypergraph. A *deterministic cover* of $H_R$ is a *boolean* function $\phi : E \rightarrow \{0,1\}$, such that

$$\forall v \in V : \sum_{e\in E \,:\, v\in e} \phi(e) = 1.$$

The *deterministic cover number* of $R$, denoted $D(R)$, is $\min_\phi \sum_{e\in E} \phi(e)$ where $\phi$ is a deterministic cover. Again, we can relax the integrality condition. Thus, we get the following definition.

DEFINITION 3.2. *A deterministic* fractional *cover of the hypergraph $H_R$ is a* real *function $\phi : E \rightarrow [0,1]$, such that*

$$\forall v \in V : \sum_{e\in E \,:\, v\in e} \phi(e) = 1.$$

*$D^*(R)$ is defined as $\min_\phi \sum_{e\in E} \phi(e)$ where $\phi$ is a deterministic fractional cover.*

Our goal is to prove lower bounds on $D(R)$ by proving lower bounds for $D^*(R)$. For this, we look at the dual linear program which is defined as follows. Let $R$ be a relation, and let $H_R$ be the corresponding hypergraph. Then, by the *duality* theorem,

$$D^*(R) = \max_w \sum_{v\in V} w(x,y),$$

where $w$ ranges over all real functions that satisfy

$$\forall e \in E \ \sum_{v\in E} w(x,y) \leq 1.$$

It is important to notice that, as opposed to the nondeterministic case, there may be a huge gap between $D(R)$ and $D^*(R)$ (we will present an example below). Still, lower bounds for $D^*(R)$ do give lower bounds to $D(R)$. Our first result does exactly that, giving a new proof to Khrapchenko's quadratic lower bound for the formula size of the parity function, by proving a lower bound for $D^*(R_{\oplus_n})$, where $R_{\oplus_n}$ is the relation associated (à la [KW88]) with parity.[9] The new proof is achieved by exhibiting an *upper* bound to the dual problem.

THEOREM. *Let $R_{\oplus_n}$ be the relation associated with the parity function (as above) then $D^*(R_{\oplus_n}) = \Theta(n^2)$.*

Our major result regarding this approach is negative though. We show that this method cannot prove superquadratic lower bounds to the formula size of *any* boolean function.

THEOREM. *Let $f$ be any boolean function and let $R_f$ be the relation associated with it. Then $D^*(R_f) = O(n^2)$.*

Note that for most boolean functions $f$, $D(R_f) = 2^{\Theta(n)}$. This result specifically suggests that anyone aiming to prove lower bounds for the circuit depth of boolean functions should abandon this approach. However, we do give some indication that proving lower bounds to *monotone* circuit depth might be possible using this approach. This gives another example of the big difference between monotone and nonmonotone computation.

**3.2. Khrapchenko's lower bound.** Khrapchenko [K71] gives the only known general lower bound for search problems. Let $R \subseteq X \times Y \times Z$ be any relation, and let $M$ be the corresponding matrix. Let $A \subseteq X \times Y$ be any set with the following properties:

1. $\forall (x,y) \in A$, $|M_{x,y}| = 1$.
2. $\forall x \in X$ and $z \in Z$ there is at most one $y \in Y$ such that $(x,y) \in A$ and $M_{x,y} = \{z\}$.
3. $\forall y \in Y$ and $z \in Z$ there is at most one $x \in X$ such that $(x,y) \in A$ and $M_{x,y} = \{z\}$.

THEOREM 3.3 ([K71]). $D(R) \geq \frac{|A|^2}{|X|\cdot|Y|}$.

As an example of an application of Theorem 3.3 consider the matrix of $R_{\oplus_n}$ indexed by $\{x : \oplus_i x_i = 1\} \times \{y : \oplus_i y_i = 0\}$ and whose $(x,y)$ entry is $\{i : x_i \neq y_i\}$.

COROLLARY 3.4. *$D(R_{\oplus_n}) \geq n^2$.*

*Proof.* Let $A = \{(x,y) \in X \times Y$ such that $d(x,y) = 1\}$.[10] It is easy to see that $A$ has the required properties and provides the desired lower bound. □

We prove a slight strengthening of this result.

THEOREM 3.5. $D^*(R) \geq \frac{|A|^2}{|X|\cdot|Y|}$.

COROLLARY 3.6. *$D^*(R_{\oplus_n}) \geq n^2$, with equality for $n = 2^k$.*

We give here the proof of the corollary. The same ideas with some technical algebraic calculations can be used to prove Theorem 3.5 in its full generality. In §3.2.1, we give some general heuristics that can help in such proofs. In §3.2.2, we use these heuristics for proving the corollary.

**3.2.1. Heuristics for proving an upper bound for the dual.** To prove a lower bound for $D^*(R)$, we only have to *exhibit* a solution to the dual program. For

[9] The relation associated with a function $f$, denoted $R_f$, consists of all triples $(x, y, i)$ such that $f(x) = 1$, $f(y) = 0$, and $x_i \neq y_i$.

[10] $d(x,y) = |\{i : x_i \neq y_i\}|$.

this, we have at our disposal the powerful paradigm of *trial and error*. The following heuristics can be quite helpful in our quest for a solution for the dual problem. After presenting these heuristics we will use them for the proof of Corollary 3.6.

- Let $\Pi(H_R)$ be the automorphism group of $H_R$. A solution $\vec{w}$ for the dual problem is *invariant* under $\Pi(H_R)$ if $w_{(x,y)} = w_{\pi(x,y)}$ for every $(x,y)$ and $\pi \in \Pi(H_R)$. Similarly, we can define invariant solutions for $D^*$. A symmetrization argument can be used to show that, without loss of generality, the optimal solutions to both $D^*$ and its dual are invariant under $\Pi(H_R)$. This clearly reduces the size of both linear programs.
- Intuitively, it is worthwhile to give $(x,y)$ a positive weight if it does not appear in many monochromatic rectangles. This is because such a positive weight does not affect many rectangles. Conversely, if $(x,y)$ appears in lots of monochromatic rectangles then we could benefit by making $w_{(x,y)}$ negative, thus helping many rectangles without lowering by much the value of $D^*$.
- Having decided which pairs $(x,y)$ will get positive weights, we could test this decision by asking whether the following modified version of $M$ has the same $D^*$:

$$\tilde{M}_{x,y} = \begin{cases} M_{x,y} & \text{if } w_{(x,y)} \text{ is positive,} \\ Z & \text{otherwise,} \end{cases}$$

  where $Z$ is the set of all possible solutions. In a sense, this means that we have to assign positive weights to the hardest pairs. Conversely, if we have a solution for the dual problem, we will get information about the *core* of the problem.
- Given an optimal solution to $D^*$, the theory of linear programming tells us which of the inequalities of the dual have to be saturated. In particular, if for a given monochromatic rectangle $e$, $\phi(e)$ is positive then the corresponding inequality in the dual has to be saturated. That is, the sum of the weights of the entries in $e$ have to add up to one. Given that we suspect that a given solution to $D^*$ is optimal, we can use this information to try to construct a suitable solution for the dual.

**3.2.2. Proof of Corollary 3.6.** We will assume that $n = 2^k$. For general $n$ the corollary follows from the theorem. The upper bound follows from the protocol attaining $\Gamma(R_{\oplus_n}) \leq n^2$, where $\Gamma$ denotes the number of different histories in the protocol. We describe it here: let $I = \{1, \ldots, n/2\}$. The players start by exchanging the parities of their vectors on $I$. That is, $\oplus_{i \in I} x_i$ and $\oplus_{i \in I} y_i$. The players then continue recursively in either $I$ or $[n] \setminus I$ depending on whether $\oplus_{i \in I} x_i \neq \oplus_{i \in I} y_i$ or not. It is easy to see that this is a correct protocol with $n^2$ different histories.

The lower bound will follow by providing a specific function $w$ for the dual problem which add up to $n^2$. At this point we could provide $w$ and finish in two more lines. Instead, we will reason using our heuristics and derive the desired solution. We therefore can get away with some informality.

First, we start with a belief that the upper bound just described is optimal. If so, we know that the inequalities associated with the chosen rectangles have to be saturated. We therefore have to understand better our upper bound. Let $A = \{(x,y) \in X \times Y \text{ such that } d(x,y) = 1\}$. A closer look at the protocol reveals that each of its histories is followed by the same number of entries from $A$. Furthermore, each history defines a square rectangle with exactly one entry from $A$ in each row and column. Note that the set of histories partition $M_{R_{\oplus_n}}$.

Following our first heuristic, $w(x, y)$ will depend only on $d(x, y)$. Following our second heuristic, it is worthwhile to give entries from $A$ a positive weight. Let us try

$$w(x, y) = \begin{cases} a & \text{if } (x, y) \in A, \\ -b & \text{otherwise,} \end{cases}$$

for some $a$ and $b$. We will finish the proof if we find $a$ and $b$ which respect all inequalities and saturate those associated with histories from our upper bound. This is because we have $n^2$ saturated rectangles which partition the whole matrix.

Let us look now at the monochromatic rectangles. In each one there is at most one entry from $A$ in every row or column. Therefore, the heaviest rectangles are the square ones with exactly one entry from $A$ in each row and column. For a $k \times k$ such square we have the inequality

$$ka - k(k-1)b \leq 1$$

with equality when $k = |A|/n^2 = N$ (the size of the rectangles in the *optimal* solution). Writing the above inequalities as $-bk^2 + (a+b)k - 1 \leq 0$ we have one root of the left-hand side, namely $k = N$. Noticing that the inequality is only restricted to integral $k$, we can let the second root be $N - 1$ and solve for $a$ and $b$. This finishes our proof. For the skeptic, we provide the final values $a = 2/N$ and $b = 1/N(N-1)$. $\square$

**3.3. The linear programming bound and boolean relations.** Let $f : \{0,1\}^n \mapsto \{0,1\}$ be a boolean function and let $R_f$ be indexed by $f^{-1}(1) \times f^{-1}(0)$, and for $(x, y) \in f^{-1}(1) \times f^{-1}(0)$ let the corresponding entry be $\{i : x_i \neq y_i\}$. We call relations of the form $R_f$ *boolean relations*. For example, $R_{\oplus_n}$ is a boolean relation. The relevance of this definition comes from the following theorem.

THEOREM 3.7 ([KW88]). *For every $f$, $d(f) = C(R_f)$ and $L(f) = \Gamma(R_f)$.*

Here, $d(f)$ and $L(f)$ are the *depth* and *formula size* of $f$ respectively. For definitions of circuits and related material concerning the above theorem see [BS90], [K89].

Let $U_n$ be the relation indexed by $\{0,1\}^n \times \{0,1\}^n$ and for $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, $x \neq y$, let the corresponding entry be $\{i : x_i \neq y_i\}$. If $x = y$, the corresponding entry remains undefined. The following claim is trivial and explains why we call $U_n$ the *universal relation* [K89].

CLAIM 3.8. *For every $f : \{0,1\}^n \mapsto \{0,1\}$, $D(R_f) \leq D(U_n)$ and $D^*(R_f) \leq D^*(U_n)$.*

We will show that the best lower bound for boolean relations attainable via the linear programming bound is very weak by proving the following theorem.

THEOREM 3.9. $D^*(U_n) = O(n^2)$.

*Proof.* We will use some of the ideas behind the logarithmic randomized protocol for $U_n$ [K89]. For $S \subseteq [n]$, let $A_S = \{x : \oplus_{i \in S} x_i = 1\} \times \{y : \oplus_{i \in S} y_i = 0\}$ and $B_S = \{x : \oplus_{i \in S} x_i = 0\} \times \{y : \oplus_{i \in S} y_i = 1\}$. The upper bound of Corollary 3.6 implies that both $D(A_S)$ and $D(B_S)$ are at most $n^2$. Let $P_S$ be an optimal partition of $A_S$ and $B_S$ into monochromatic rectangles.

It is easy to see that for every $x \neq y$, $(x, y) \in A_S \cup B_S$ for exactly half the subsets $S$. We will give a weight of $2^{-(n-1)}$ to every rectangle from $\cup_{S \subseteq [n]} P_S$. Each pair $(x, y)$ with $x \neq y$ is covered by $2^{n-1}$ such rectangles with total unit weight. Also, the total weight is $2^n \cdot (2n^2) \cdot 2^{-(n-1)} = 4n^2$. $\square$

**3.4. The linear programming bound and monotone boolean relations.** Let $f : \{0,1\}^n \mapsto \{0,1\}$ be a *monotone* boolean function. We denote by $\min(f)$ and $\max(f)$ the set of minterms and maxterms of $f$ respectively (see [K89] for definitions). Note that each minterm and each maxterm, as a set of variables, intersect. Let $R_f^m$ be a relation indexed by $\min(f) \times \max(f)$ and for $(p,q) \in \min(f) \times \max(f)$ let the corresponding entry be $p \cap q$. We call matrices of the form $R_f^m$ *monotone boolean relations.* The following theorem is the monotone analogue of Theorem 3.7.

THEOREM 3.10 ([KW88]). *For every monotone function $f$, $d_m(f) = C(R_f^m)$ and $L_m(f) = \Gamma(R_f^m)$.*

Here, $d_m$ and $L_m$ denote monotone depth and monotone formula size (see [K89] for definitions).

As in the preceding section, we define a *monotone universal relation.* Let $U_n^m$ be a relation indexed by $\mathcal{P}([n]) \times \mathcal{P}([n])$ and for $p, q \in \mathcal{P}([n])$ with $p \cap q \neq \emptyset$ let the corresponding entry be $p \cap q$. If $p \cap q = \emptyset$ the entry of $U_n^m$ remains undefined. We have the following claim.

CLAIM 3.11. *For every monotone function $f : \{0,1\}^n \mapsto \{0,1\}$ we have $D(R_f^m) \leq D(U_n^m)$ and $D^*(R_f^m) \leq D^*(U_n^m)$.*

The main reason to define universal relations is to try on them new ideas to prove lower bounds. The fact that $D^*(U_n) = O(n^2)$ means that the best lower bound for boolean relations attainable with the linear programming bound is at most quadratic. The following theorem gives evidence to the fact that the linear programming bound may give exponential lower bounds for monotone boolean relations.

THEOREM 3.12. *$D^*(U_n^m) \geq d^n$ for some constant $d > 1$.*

*Proof.* We will construct a feasible solution to the dual problem whose value is the desired bound. Following our first heuristic, $w(p,q)$ depends only on $|p \cap q|$. It is natural to try the Hadamard matrix $((-1)^{|p\cap q|})_{p,q\in\mathcal{P}([n])}$ as the sign pattern for our weights. We want to give a positive weight to those entries $(p,q)$ with $|p \cap q| = 1$ and we must give zero weights to the undefined entries. Let us try

$$w(p,q) = \begin{cases} 0 & \text{if } p \cap q = \emptyset, \\ -(-1)^{|p\cap q|}c & \text{otherwise,} \end{cases}$$

for some constant $c$ to be specified later. Using the fact that

$$\sum_{p,q\in\mathcal{P}([n])} (-1)^{|p\cap q|} = 2^n$$

we get

$$\sum_{p,q\in\mathcal{P}([n])} w(p,q) = c \cdot |\{(p,q) : p \cap q = \emptyset\}| = c \cdot (3^n - 2^n).$$

We now look at the monochromatic rectangles. Let $R_i = \{p : i \in p\} \times \{q : i \in q\}$. Every monochromatic rectangle is a subrectangle of $R_i$ for some $i$. The pattern of signs of weights of entries from $R_i$ constitute an $2^{n-1} \times 2^{n-1}$ Hadamard matrix. We will use the following lemma of Lindsey [ES74, p. 88] which says that minors of a Hadamard matrix are *balanced.*

LEMMA 3.13. *Let $H$ be an $N \times N$ Hadamard matrix and let $S$ and $T$ be subsets of rows and columns respectively. Then,*

$$\sum_{i\in S, j\in T} H_{i,j} \leq \sqrt{N \cdot |S| \cdot |T|}\,.$$

In our case, we use the lemma to show that for every subrectangle $R$ of $R_i$

$$\sum_{(p,q)\in R} w_{(p,q)} \leq c \cdot \sqrt{2^{n-1}2^{n-1}2^{n-1}} \leq c \cdot \sqrt{2^{3n}} \leq c \cdot (\sqrt{8})^n$$

which is less than 1 if $c = (\sqrt{8})^{-n}$. We have therefore found a feasible solution whose value is $(\sqrt{8})^{-n}(3^n - 2^n) \geq d^n$ for any $d < 3/\sqrt{8}$. □

## 4. Deterministic communication: two rounds.

### 4.1. Introduction.

The previous approach tried to look at the protocol globally, and failed. Our next approach deals with the protocol in a round-by-round fashion. We will associate a covering problem with every round of the protocol. Unfortunately, we are not able to carry our results to protocols having an arbitrary number of rounds, but only succeed for one-round and two-round protocols.

To best explain our approach let us first limit ourselves to one-round protocols.[11] Intuitively, in a one-round protocol, $P_2$ partitions the columns of the matrix in a way that enables $P_1$ to decide on the answer. Formally, we associate with any relation $R$ the following covering problem. Let $X$ denote the space of all possible inputs to $P_1$, and $Y$ the space of inputs to $P_2$. A set $A \subseteq Y$ is called *compatible* if for every $x \in X$ there exists an answer $z$ that is a legal answer for all $y \in A$ (i.e., such that $R(x, y, z)$ holds for all $y \in A$). $D_1(R)$ is defined to be the minimum number of compatible sets that are needed in order to cover $Y$.

It is not difficult to see that $\log D_1(R)$ gives the *one-round communication complexity* of $R$ (denoted by $C_{D_1}(R)$). It is also not difficult to see that in this case the disjoint and nondisjoint covers are the same, and thus when we relax the integer problem to a linear one, giving $D_1{}^*(R)$, we will be able to use Lovász's results regarding fractional covers. At this point we will already be able to reprove the "direct sum" results for one-round deterministic complexity obtained in [FKN91], specifically $C_{D_1} \geq \tilde{C}_{D_1} \geq C_{D_1} - O(\log n)$. In [KRW91] it was conjectured that for every $R$, $\tilde{C}_D(R) \geq C_D(R) - O(\log n)$. In [FKN91] it was proved that $\tilde{C}_D(R) \geq \sqrt{C_D(R)} - O(\log n)$. Here we show that the conjecture is true for two-round protocols.

We look at two-round protocols[12] in the following way: in the first round, $P_1$ partitions the rows of the matrix, and then the parties continue with a one-round protocol on the subdomain. This can be expressed as the following *weighted* covering problem. Our aim is to cover $X$, where we are allowed to use any subset of $X$ in the cover, and the cost of using a subset $A \subseteq X$ is the one-round complexity of solving $R$ given that $x \in A$, denoted $D_1(A)$.

DEFINITION 4.1. *A* cover *of $X$ is a* boolean *function* $\phi : P(X) \to \{0,1\}$, *such that*

$$\forall x \in X : \sum_{A\in P(X)\,:\,x\in A} \phi(A) \geq 1.$$

[11] $P_2$ sends to $P_1$ a single message, and $P_1$ then needs to compute the answer.

[12] $P_1$ sends a message to $P_2$, who sends another message to $P_1$, who computes the answer.

*The* weighted cover number *of $R$, denoted $D_2(R)$, is defined as*

$$D_2(R) = \min_{\phi} \sum_{A \in P(Y)} \phi(A) D_1(A),$$

*where $\phi$ is a cover.*

It is not difficult to see that $\log D_2(R)$ is equal (up to a constant factor) to the two-round deterministic complexity, $C_{D_2}(R)$. Again, we relax the integrality conditions and look at the resulting linear program giving $D_2{}^*$. We can now no longer use Lovász's results, as we have a "weighted" covering problem. This problem was already considered by Chvátal [C79] who extended the first part of Theorem 2.6 to the "weighted" case. We prove that the second part of Theorem 2.6 can be generalized as well. We believe that these generalizations are of independent interest. Using these generalization we can prove a direct-sum result for two-round communication complexity. In particular, let $\tilde{C}_{D_2}(R)$ denote the amortized two-round communication complexity of $R$.

THEOREM 4.2. *For every two relations $R$ and $S$,*
- $D_2(R)D_2(S)/\text{poly}(n) \leq D_2(R \times S) \leq D_2(R)D_2(S)$;
- $\tilde{C}_{D_2}(R) = \Theta(C_{D_2}(R)) - O(\log n)$.

**4.2. Weighted fractional covers.** In this subsection we present the new notion of *weighted fractional covers.* This notion will be later used in the proof of Theorem 4.2.

DEFINITION 4.3. *Let $H$ be a hypergraph and let $w$ be a* weight function *defined on $E(H)$ such that $w(e) \geq 1$, for every hyperedge $e$. Given a deterministic/nondeterministic integral/fractional cover $\phi$ the weight function $w$ gives it a weight $w(\phi) = \sum_{e \in E(H)} w(e)\phi(e)$. The* weighted cover numbers *$D(H,w), D^*(H,w), N(H,w)$, and $N^*(H,w)$ are defined as the minimum of $w(\phi)$ over all appropriate covers $\phi$. (Note that the original definitions, as presented in §2.1, are special cases of the new definitions with $w \equiv 1$.)*

The next theorem is an extension of Theorem 2.6.

THEOREM 4.4. *Let $H$, $H_1$, and $H_2$ be any hypergraphs, and let $w$, $w_1$, and $w_2$ be weight functions on $E(H)$, $E(H_1)$, and $E(H_2)$ (respectively) that give weights $\geq 1$ for every hyperedge (i.e., $w(e) \geq 1$, for all $e \in E(H)$). Then*

1. $N^*(H,w) \geq \frac{N(H,w)}{\ln |V(H)|}$.
2. $N^*(H_1 \times H_2, w_1 \times w_2) = N^*(H_1,w_1) \cdot N^*(H_2,w_2)$, *where $w_1 \times w_2$ is defined as $w_1 \times w_2(e_1 \times e_2) = w_1(e_1) \cdot w_2(e_2)$.*

*Proof.* Part (1) of the theorem was proved in [C79].[13] To prove (2) we first prove that $N^*(H_1 \times H_2, w_1 \times w_2) \leq N^*(H_1,w_1) \cdot N^*(H_2,w_2)$. Let $\phi_1$ and $\phi_2$ be the optimal-weight fractional covers for $H_1$ and $H_2$ (i.e., those that give the minimum for $N^*(H_1,w_1)$ and $N^*(H_2,w_2)$ respectively). Define $\phi(e_1 \times e_2) = \phi_1(e_1) \cdot \phi_2(e_2)$. We show that $\phi$ is a nondeterministic fractional cover of $H_1 \times H_2$ and that its weight is the multiplication of the weights of $\phi_1$ and $\phi_2$. Clearly, $\phi$ is a function from $E \triangleq E(H_1 \times H_2)$ to $[0,1]$. In addition, every vertex $(v_1,v_2) \in V(H_1 \times H_2)$ is covered as needed:

$$\sum_{e_1 \times e_2 \in E \,:\, (v_1,v_2) \in e_1 \times e_2} \phi(e_1 \times e_2) = \sum_{e_1 \in E_1 \,:\, v_1 \in e_1} \sum_{e_2 \in E_2 \,:\, v_2 \in e_2} \phi(e_1 \times e_2)$$

[13] In fact, the result stated in [C79] is somewhat different than the one stated here. However, the proof in [C79] immediately implies the result stated here.

$$\begin{aligned}
&= \sum_{e_1\in E_1\,:\,v_1\in e_1} \sum_{e_2\in E_2\,:\,v_2\in e_2} \phi_1(e_1)\phi_2(e_2)\\
&= \sum_{e_1\in E_1\,:\,v_1\in e_1} \phi_1(e_1) \sum_{e_2\in E_2\,:\,v_2\in e_2} \phi_2(e_2)\\
&\geq 1\cdot 1 = 1.
\end{aligned}$$

Thus, $\phi$ is a legal cover and we get

$$\begin{aligned}
N^*(H_1\times H_2, w_1\times w_2) &\leq \sum_{e_1\times e_2\in E} w_1\times w_2(e_1\times e_2)\cdot\phi(e_1\times e_2)\\
&= \sum_{e_1\times e_2\in E} w_1(e_1)w_2(e_2)\phi_1(e_1)\phi_2(e_2)\\
&= \sum_{e_1\in E_1} w_1(e_1)\phi_1(e_1)\cdot\sum_{e_2\in E_2} w_2(e_2)\phi_2(e_2)\\
&= N^*(H_1,w_1)\cdot N^*(H_2,w_2).
\end{aligned}$$

For proving the other direction, that is $N^*(H_1\times H_2,w_1\times w_2)\geq N^*(H_1,w_1)\cdot N^*(H_2,w_2)$, it is convenient to use again the dual program

$$N^*(H,w) = \max\left\{\vec{1}^T\Phi | A\Phi \leq w, \Phi\geq\vec{0}\right\}.$$

We can think about every such vector $\Phi$ as a real function defined over $V(H)$. Let $\Phi_1$ and $\Phi_2$ be the functions that give the maximum for $N^*(H_1,w_1)$ and $N^*(H_2,w_2)$ (respectively), at the above linear program. Define $\Phi(v_1,v_2)=\Phi_1(v_1)\cdot\Phi_2(v_2)$. We show that $\Phi$ satisfies the conditions in the linear program for $H_1\times H_2$ and that its value (i.e, $\sum_{v\in V(H_1\times H_2)}\Phi(v)$) is the multiplication of the values of $\Phi_1$ and $\Phi_2$. Clearly, $\Phi$ is a nonnegative function defined over $V\overset{\triangle}{=}V(H_1\times H_2)$. In addition, for every hyperedge $e_1\times e_2\in E(H_1\times H_2)$

$$\begin{aligned}
\sum_{(v_1,v_2)\in V\,:\,(v_1,v_2)\in e_1\times e_2} \Phi(v_1,v_2) &= \sum_{v_1\in V_1\,:\,v_1\in e_1}\sum_{v_2\in V_2\,:\,v_2\in e_2} \Phi(v_1,v_2)\\
&= \sum_{v_1\in V_1\,:\,v_1\in e_1}\sum_{v_2\in V_2\,:\,v_2\in e_2} \Phi_1(v_1)\Phi_2(v_2)\\
&= \sum_{v_1\in V_1\,:\,v_1\in e_1}\Phi_1(v_1)\sum_{v_2\in V_2\,:\,v_2\in e_2} \Phi_2(v_2)\\
&\leq w_1(e_1)\cdot w_2(e_2)\\
&= w_1\times w_2(e_1\times e_2).
\end{aligned}$$

Thus,

$$\begin{aligned}
N^*(H_1\times H_2,w_1\times w_2) &\geq \sum_{(v_1,v_2)\in V}\Phi(v_1,v_2)\\
&= \sum_{(v_1,v_2)\in V}\Phi_1(v_1)\Phi_2(v_2)\\
&= \sum_{v_1\in V_1}\Phi_1(v_1)\cdot\sum_{v_2\in V_2}\Phi_2(v_2)\\
&= N^*(H_1,w_1)\cdot N^*(H_2,w_2).
\end{aligned}$$

This completes the proof of the theorem. □

**4.3. Proof of Theorem 4.2.** Now, we can come back to the proof of Theorem 4.2. To analyze the two-round deterministic communication complexity of a relation $R$, we define the following hypergraph $H_R^2$. The vertices are again all the pairs in $\{0,1\}^n \times \{0,1\}^n$. The hyperedges are all the rectangles of the form $A \times \{0,1\}^n$, where $A \subseteq \{0,1\}^n$. Now, for each such hyperedge $e$ we define its weight to be $D_1(e)$, the one-way deterministic communication complexity of computing $R$ on the subdomain $e$.

Note that for every relation $R$, the definitions of $H_R^2$ and of $D_1$ imply that $D(H_R^2, D_1) = N(H_R^2, D_1)$ and $D^*(H_R^2, D_1) = N^*(H_R^2, D_1)$.

We are interested in the relations between $D(H_{R\times S}^2, D_1)$ and $D(H_R^2, D_1), D(H_S^2, D_1)$. Again, it can be easily verified that $D(H_{R\times S}^2, D_1) \le D(H_R^2, D_1) \cdot D(H_S^2, D_1)$. For proving connections in the opposite direction, let us concentrate for a while on the case of computing functions. We need the following lemma.

LEMMA 4.5. *Let $e \in E(H_{f\times g}^2)$. That is, $e = A \times (\{0,1\}^n \times \{0,1\}^n)$, where $A \subseteq (\{0,1\}^n \times \{0,1\}^n)$. Let $A_f$ and $A_g$ be the projection of $A$ on the first and second coordinates (respectively). Then,*

1. $D_1((A_f \times A_g) \times (\{0,1\}^n \times \{0,1\}^n)) = D_1(A_f \times \{0,1\}^n) \cdot D_1(A_g \times \{0,1\}^n)$;
2. $D_1(e) = D_1((A_f \times A_g) \times (\{0,1\}^n \times \{0,1\}^n))$.

*Proof.* (1) was proved in [FKN91]. We now prove (2): As $A \subseteq A_f \times A_g$ then one direction is trivial. Therefore, it is enough to prove that for any $B \subseteq (\{0,1\}^n \times \{0,1\}^n)$, if the submatrix $A \times B$ is constant in each row then so is the bigger submatrix $(A_f \times A_g) \times B$. By the definitions if $A \times B$ in constant in each row then for every $(x_1, x_2) \in A$ and every $(y_1, y_2), (y_1', y_2') \in B$ we have $f \times g((x_1,x_2),(y_1,y_2)) = f \times g((x_1,x_2),(y_1',y_2'))$. In particular, this means that for every $x_1 \in A_f, x_2 \in A_g$ and every $(y_1,y_2),(y_1',y_2') \in B$ we have $f(x_1,y_1) = f(x_1,y_1')$ and $g(x_2,y_2) = g(x_2,y_2')$, which gives us what we need. □

The following theorem is an analogue of Lemma 2.9.

THEOREM 4.6. *Let $f$ and $g$ be two functions. Then $N^*(H_{f\times g}^2, D_1) = N^*(H_f^2 \times H_g^2, D_1 \times D_1)$.*

*Proof.* The proof that $N^*(H_{f\times g}^2, D_1) \le N^*(H_f^2 \times H_g^2, D_1 \times D_1)$ is similar to the proof of Theorem 2.9 (second direction), together with the first part of Lemma 4.5 that guarantees that $D_1(e_f \times e_g) = D_1(e_f) \cdot D_1(e_g)$.

The proof that $N^*(H_{f\times g}^2, D_1) \ge N^*(H_f^2 \times H_g^2, D_1 \times D_1)$ is similar to the proof of Theorem 2.9 (first direction), together with the second part of Lemma 4.5 that guarantees that $D_1(e) = D_1(e_f) \cdot D_1(e_g)$. □

Using the last two theorems, we get

$$D(H_f^2, D_1) \cdot D(H_g^2, D_1) \ge D(H_{f\times g}^2, D_1) \ge \frac{D(H_f^2, D_1) \cdot D(H_g^2, D_1)}{cn^2},$$

for some constant $c$.

Let us now briefly discuss the case of computing general relations and not necessarily functions. The equality in Lemma 4.5 part (1) does not hold anymore. However, by [FKN91] the two sides cannot be too far. As a result, Theorem 4.6 is changed as well and it claims: let $R$ and $S$ be two relations. Then

$$\begin{aligned}\frac{N^*(H_R^2 \times H_S^2, D_1 \times D_1)}{\ln |V(H_{R\times S})|} &\le N^*(H_{R\times S}^2, D_1) \\ &\le N^*(H_R^2 \times H_S^2, D_1 \times D_1),\end{aligned}$$

which implies

$$D(H^2_R, D_1) \cdot D(H^2_S, D_1) \geq D(H^2_{R\times S}, D_1) \geq \frac{D(H^2_R, D_1) \cdot D(H^2_S, D_1)}{cn^4}.$$

## REFERENCES

[AUY83] A. V. Aho, J. D. Ullman, and M. Yannakakis, *On notions of information transfer in* VLSI *circuits*, Proc. 15th Annual ACM Symp. on Theory of Computing, 1989, pp. 133–139.

[BS90] R. Boppana and M. Sipser, *The Complexity of finite functions*, in Handbook of Theoretical Computer Science (Vol. A), J. van Leeuwen, ed., Elsevier Science Publishers, 1990, pp. 759–804.

[C79] V. Chvátal, *A greedy heuristic for set-covering problem*, Math. Oper. Res., 4 (1979), pp. 233–235.

[ES74] P. Erdos and J. Spencer, *Probabilistic Methods in Combinatorics*, Academic Press, New York, 1974.

[FKN91] T. Feder, E. Kushilevitz, and M. Naor, *Amortized communication complexity*, Proc. 32nd Annual Symposium on Foundations of Computer Science, 1991, pp. 239–248; SIAM J. Comput., 25 (1995), to appear.

[F87] M. Furer, *The power of randomness for communication complexity*, Proc. 19th Annual ACM Sympos. on Theory of Computing, (1987), pp. 178–181.

[GH89] M. Goldmann and J. Hastad, *A simple lower bound for monotone clique using a communication game*, Inform. Process. Lett., 41 (1992), pp. 221–226.

[K71] V. Khrapchenko, *A method of determining lower bounds for the complexity of π-schemes*, Math. Notes Acad. Sci. USSR, (1971), pp. 474–479.

[K89] M. Karchmer, *Communication Complexity: A New Approach to Circuit Depth*, The MIT Press, Cambridge, MA, 1989.

[KRW91] M. Karchmer, R. Raz, and A. Wigderson, *On proving super-logarithmic depth lower bounds via the direct sum in communication complexity*, in Proc. 6th IEEE Structure in Complexity Theory Annual Conference, (1991), pp. 299–304.

[KW88] M. Karchmer and A. Wigderson, *Monotone circuits for connectivity require super-logarithmic depth*, SIAM J. Discrete Math., 3 (1990), pp. 255–265.

[L75] L. Lovász, *On the ratio of optimal integral and fractional covers*, Discrete Math., 13 (1975), pp. 383–390.

[MS82] K. Mehlhorn and E. M. Schmidt, *Las Vegas is better than determinism in VLSI and distributive computing*, Proc. 14th Annual ACM Sympos. on Theory of Computing, (1982), pp. 330–337.

[N91] I. Newman, *Private vs. common random bits in communication complexity*, Inform. Process. Lett. 39 (1991), pp. 67–71.

[Y79] A. C.-C. Yao, *Some complexity questions related to distributive computing*, Proc. 11th Annual ACM Symp. on Theory of Computing, 1979, pp. 209–213.

SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 93–98, February 1995

# ON LOTTERIES WITH UNIQUE WINNERS[*]

EYAL KUSHILEVITZ[†], YISHAY MANSOUR[‡], AND MICHAEL O. RABIN[§]

**Abstract.** Lotteries with the *unique maximum* property and the *unique winner* property are considered. Tight lower bounds are proven on the domain size of such lotteries.

**Key words.** lotteries, lower bounds, symmetry breaking

**AMS subject classifications.** 68Q22, 68R99

**1. Introduction.** A *lottery* is a collection of discrete, independent random variables $\Pi_1, \ldots, \Pi_N$ defined over a set $\{1, \ldots, B\}$. Sometimes, we associate with each random variable $\Pi_i$ a player $P_i$ and think of a lottery as a subset of players choosing numbers, each player $P_i$ according to the corresponding $\Pi_i$. A lottery has the *unique maximum property* if for *every* subset of the random variables $\Pi_1, \ldots, \Pi_N$, with constant probability (say 2/3), the maximum value of the random variables is chosen by *exactly* one random variable. (Formally, for every non-empty subset $S \subseteq \{1, \ldots, N\}$, define the random variable $M_S = \max_{\{i \in S\}} \Pi_i$. Let $p_S$ be the probability that $|\{i \in S : \Pi_i = M_S\}| = 1$. The unique maximum property states that $p_S \geq 2/3$ for every $S$.)

A lottery has the *unique winner property* if for *every* subset of random variables, with constant probability, there *exists* a value that is chosen by *exactly* one random variable. (Formally, let $q_S$ be the probability that there exists a $j \in \{1, \ldots, B\}$ such that $|\{i \in S : \Pi_i = j\}| = 1$. The unique winner property states that $q_S \geq 2/3$ for every $S$.)

Lotteries with these properties have many applications in computer science, especially in cases where symmetry breaking is required. Examples include randomized mutual exclusion algorithms [7, 5], broadcast in radio networks [1], elections in anonymous networks [6], and various CRCW-PRAM algorithms [2]. [1]

A trivial way to achieve these properties is by letting the participants draw numbers uniformly in the set $\{1, \ldots, B\}$, where $B$ is "large enough" (compare to $N$). For $B = N$ with constant probability, the maximum is unique (and with much higher probability there exists a uniquely chosen value). Unfortunately, in the applications it is important that $B$ is as small as possible, as this value corresponds to important complexity measures such as *time* (in the case of radio broadcast) and *space* (in the case of mutual exclusion).

Rabin [7] described and analyzed the following *geometric* lottery: Let $B = \log_2 N + 4$. All players use the same probability distribution; for every $i$, $1 \leq i \leq B - 1$,

* Received by the editors January 14, 1994; accepted for publication (in revised form) January 21, 1994.

† Department of Computer Science, Technion, Haifa 32000, Israel (eyalk@cs.technion.il). The research of this author was supported by Office of Naval Research contract N0001491-J-1981 and National Science Foundation contract CCR-90-07677.

‡ Department of Computer Science, Tel-Aviv University, Israel (mansour@math.tau.ac.il).

§ Department of Computer Science, Aiken Computation Laboratory, Harvard University, Cambridge, Massachusetts 02138 (rabin@das.harvard.edu). The research of this author was supported by Office of Naval Research contact N0001491-J-1981 and National Science Foundation contract CCR-90-07677.

1 Some of these applications require the unique maximum property (e.g., the mutual exclusion) while for others the unique winner property is sufficient (e.g., the radio broadcast).

the value $i$ is chosen with probability $1/2^i$. The value $B$ is chosen with probability $1/2^{B-1}$. Rabin proved that this lottery has the unique maximum property.

This research was initiated with the motivation of discovering whether this construction can be improved or if it is optimal (in the sense of the number of values, i.e., $B$). The results of this note show that it is optimal (up to constants).

A critical point is that the number of *actual* participants, $t$, is not known in advance. If $t$ was known beforehand, we would be able to use the following lottery: choose the value 1 with probability $1 - 1/t$ and the value 2 with probability $1/t$. One can verify that if $t$ numbers are chosen according to this lottery, then with probability of about $1/e$ the maximum is unique. (This probability can easily be improved to 2/3.) This way we get a lottery whose number of values $B$ is independent of $N$. However, we prove that this cannot be the case when $t$ is not known in advance.[2] Namely, we show that every lottery with either the unique maximum property or the unique winner property requires $B = \Omega(\log N)$.[3]

A different line of research is to give lower bounds for the problems in which those lotteries are used. Following this research, a significant progress was made in this direction; in [4], a lower bound for randomized mutual exclusion is proven. From this lower bound, one can get a lower bound for lotteries with the unique maximum property, in which all players use the *same* random variable $\Pi$. In [3], a lower bound for broadcast in radio networks is proven. From this lower bound, the results of this note can be derived. However, the direct proofs in this note are much simpler and give a better understanding of the problem as well as much better constants than those that can be obtained indirectly by using the results of [3].

**2. Lotteries with the unique maximum property.** In this section we prove that any lottery with the unique maximum property requires $\Omega(\log N)$ values.

THEOREM 2.1. *Let $B$ be an integer. Let $\Pi_1, \dots, \Pi_N$ be a lottery for $N$ players $P_1, \dots, P_N$ over the set $\{1, 2, \dots, B\}$. If the lottery has the unique maximum property, then $B \geq \log_6 N$.*

*Proof.* We use the following notation: Let $A$ be a set of participants, and let $E$ be an event; then $Pr(E|A)$ denotes the probability that the event $E$ happens given that $A$ is the set of the participants in the lottery (and each participant $P_i \in A$ uses the corresponding random variable $\Pi_i$). We use the following definitions: Let $A \subseteq \{P_1, \dots, P_N\}$ be a non-empty set of participants. We define

$$m(A) \triangleq \max_{1 \leq j \leq B} \left[ Pr(\max \geq j|A) > \frac{1}{2} \right].$$

That is, $m(A)$ is the maximal value $j$, such that if $A$ is the set of participants in the lottery, the probability that the maximum number drawn is at least $j$ is greater than 1/2. We also define for every $1 \leq t \leq N$,

$$m(t) \triangleq \min_{A:|A|=t} m(A).$$

This definition satisfies the following trivial properties:

- For every $A$, $m(A)$ is well defined (as at least $j = 1$ satisfies the condition).

[2] Again, this is usually the case in the applications.

[3] Clearly, the $\Omega(\log N)$ bound for lotteries with the unique winner property implies the same bound for lotteries with the unique maximum property. Nevertheless, we present two different proofs as the proof for the case of unique maximum is much simpler and yields a better constant.

- For every $A$, $1 \leq m(A) \leq B$; and therefore for every $1 \leq t \leq N$, $1 \leq m(t) \leq B$.
- If $A' \subseteq A$, then $m(A') \leq m(A)$. This follows immediately from the definition of $m$ and by the fact that for every $j$, $Pr(\max \geq j|A) \geq Pr(\max \geq j|A')$. This implies that if $t' < t$, then $m(t') \leq m(t)$ (take $A$ to be a set that gives the minimum for $m(t)$ and $A'$ a subset of $A$ of size $t'$, then $m(t') \leq m(A') \leq m(A) = m(t)$).

The following claim says that $m(t)$ is not only non-decreasing but should be strictly increasing from time to time.

CLAIM. *Assume $t$ is divided by 6. Then $m(t/6) < m(t)$.*

*Proof of Claim.* Assume, by way of contradiction, that $m(t) = m(t/6) = j_0$. Let $A$ be a set of size $t$ such that $m(A) = j_0$ (i.e., $A$ gives the minimum for $m(t)$). Partition the set $A$ into six disjoint subsets, $A_1, \dots, A_6$ each of size $t/6$. For each of these $A_i$'s, since $A_i \subseteq A$, it follows that $m(A_i) \leq m(A) = j_0$. On the other hand, since $|A_i| = t/6$, $m(A_i) \geq m(t/6) = j_0$. Thus, $m(A_i) = j_0$. This in particular implies that $Pr(\max \geq j_0|A_i) > 1/2$.

Let $M$ be the random variable that is the number of $A_i$'s for which the maximum is at least $j_0$, and let $M'$ be the number of $A_i$'s for which the maximum is exactly $j_0$ or $M' = 0$ in the case that any of these maximum values is greater than $j_0$ (i.e., $M$ and $M'$ take values in $\{0, 1, \dots, 6\}$). Note that if $M' \geq 2$, then the lottery fails. Also note that

$$Pr[M' \geq 2|A] \geq Pr[M \geq 2|A] - Pr[\max \geq j_0 + 1|A].$$

Since $m(A) = j_0$, we have $Pr[\max \geq j_0 + 1|A] \leq 1/2$. By the independence of choices between the $A_i$'s and since for each of the $A_i$'s $Pr(\max \geq j_0|A_i) > 1/2$, we get $Pr[M \geq 2|A] > 57/64$. All together we get that $Pr[M' \geq 2|A] \geq 57/64 - 1/2 = 25/64 > 1/3$. This contradicts the assumption that the algorithm succeeds with probability 2/3 for any number of participants. $\square$

As $m(N) \leq B$ and $m(1) \geq 1$, the above claim implies $B \geq \log_6 N$. This completes the proof of the theorem.

## 3. Lotteries with the unique winner property.

In this section we prove that any lottery with the unique winner property requires $\Omega(\log N)$ values. We start by proving it for the case that all players use the same probability distribution. Then we prove the general case by reducing it to this special case.

THEOREM 3.1. *Let $B$ be an integer. Let $\Pi_1, \dots, \Pi_N$ be a lottery for $N$ players, over the set $\{1, 2, \dots, B\}$, such that $\Pi_1 = \cdots = \Pi_N \triangleq \Pi$. If the lottery has the unique winner property, then $B \geq (\frac{1}{2}\log_6 N) - 2$.*

*Proof.* Let $p_j$ be the probability, according to $\Pi$, of picking the number $j$. We consider the probabilities $p_1, p_2, \dots, p_B$ and prove that for every $t$ ($1 \leq t \leq N$) there must be one of the $p_j$'s that is "close" to $1/t$. Otherwise, if all the $p_j$'s are either "much bigger" than $1/t$ or "much smaller" than $1/t$ and there are $t$ participants that choose numbers according to these probabilities, then with a high probability each number is either picked at least twice or is not picked at all. In such a case there is no number that is chosen by a single participant (i.e., no unique winner). Therefore, for every $t$ there must be (at least) one of the $p_j$'s that is "close" to $1/t$, and this implies the result.

More formally, suppose that $B < (\frac{1}{2}\log_6 N) - 2$ (otherwise, we are done). Let $m = 2B + 3$ and $0 < \alpha < 1$ be some small enough constant (e.g., $\alpha = 1/6$). We associate with every probability $p_j$ $(1 \leq j \leq B)$ a subinterval $I_j = [\ell_j, u_j]$ of $[0,1]$ that contains $p_j$, in the following way: Let $i$ $(\geq 1)$ be the smallest integer such that $p_j \leq \alpha^i$ and such that $\alpha^i$ is not the right point of any $I_{j'}$, for $j' < j$. If such an $i$ exists, then $u_j = \alpha^i$; otherwise $u_j = 1$. Let $i$ $(\leq m)$ be the largest integer such that $p_j \geq \alpha^i$ and such that $\alpha^i$ is not the left point of any $I_{j'}$, for $j' < j$. If such an $i$ exists, then $\ell_j = \alpha^i$; otherwise $\ell_j = 0$. As $m = 2B + 3$, by the way of constructing the subintervals $I_j$ $(1 \leq j \leq B)$ there exists an index $1 < i < m$ such that $\alpha^i$ does not belong to any of these subintervals. Consider the case where $t = 1/\alpha^i$ numbers are chosen. In this case we prove that with a "high probability" each "big" $j$ (i.e., $j$ such that $p_j \geq \alpha^{i-1}$) is chosen at least once and each "small" $j$ (i.e., $j$ such that $p_j \leq \alpha^{i+1}$) is not chosen at all:

$$\begin{aligned}
Pr\,(\text{no "small" } j \text{ is picked} \mid t) &= 1 - Pr\,(\text{some "small" } j \text{ is picked} \mid t) \\
&\geq 1 - \sum_{j:p_j \leq \alpha^{i+1}} Pr\,(j \text{ is picked} \mid t) \\
&\geq 1 - \sum_{j:p_j \leq \alpha^{i+1}} t \cdot p_j \\
&= 1 - t \cdot \sum_{j:p_j \leq \alpha^{i+1}} p_j.
\end{aligned}$$

By the construction of the subintervals $I_j$ we can bound the $p_j$'s in the above sum by the corresponding $u_j$'s that form a geometric progression. Thus the sum is bounded by $\alpha^{i+1} \cdot 1/(1-\alpha)$. Therefore,

$$Pr\,(\text{no "small" } j \text{ is picked} \mid t) \geq 1 - \frac{t \cdot \alpha^{i+1}}{1-\alpha}.$$

By the choice of $t$, this is equal to $(1-2\alpha)/(1-\alpha)$. By the choice of $\alpha = 1/6$, this is at least $4/5$. Similarly, we have

$$\begin{aligned}
Pr\,(\text{every "big" } j \text{ is picked} \mid t) &= 1 - Pr\,(\text{some "big" } j \text{ is not picked} \mid t) \\
&\geq 1 - \sum_{j:p_j \geq \alpha^{i-1}} Pr\,(j \text{ is not picked} \mid t) \\
&= 1 - \sum_{j:p_j \geq \alpha^{i-1}} (1-p_j)^t.
\end{aligned}$$

By the construction of the subintervals $I_j$ we can bound the $p_j$'s in the above sum by the corresponding $\ell_j$'s. Thus we have

$$\sum_{j:p_j \geq \alpha^{i-1}} (1-p_j)^t \leq \sum_{j:p_j \geq \alpha^{i-1}} (1-\ell_j)^t.$$

In addition, all the $\ell_j$'s in the last sum are of the form $\alpha^k$, $k < i$, and $t = 1/\alpha^i$. Therefore, the last sum is less than

$$\sum_{j=1}^{i-1} e^{-(\frac{1}{\alpha})^{j-i}}.$$

By the choice of $\alpha = 1/6$ this sum is at most $1/5$; therefore, the above probability is at least $4/5$. Therefore, with probability at least $3/5$ each "big" $j$ is chosen at least once and each "small" $j$ is not chosen at all. Hence, when there are $2t$ participants, with probability $\geq 9/25 > 1/3$ each "big" $j$ is chosen at least twice and each "small" $j$ is not chosen at all. Therefore, with probability $> 1/3$ no number is chosen by a single participant—contradicting the requirement about the lottery. The only item remaining to be verified is that $2t \leq N$ (otherwise there are not enough players). This follows from our choice of parameters: as $t = 1/\alpha^i$, $\alpha = 1/6$, $i < m$, $m = 2B + 3$, and by assumption $B < (\frac{1}{2}\log_6 N) - 2$, it follows that $t = 1/\alpha^i = 6^i \leq 6^{m-1} = 6^{2B+2} = 6^{\log_6 N - 2} < N/2$. The theorem follows. □

In the following theorem we extend the result of the previous theorem to the case where each player $P_i$ may use a different distribution $\Pi_i$. The proof is by a reduction to the case where all players use the same distribution.

THEOREM 3.2. *Let $B$ be an integer. Let $\Pi_1, \ldots, \Pi_N$ be a lottery for $N$ players, over the set $\{1, 2, \ldots, B\}$. If the lottery has the unique winner property, then $B \geq d \cdot \log_6 N$, for some constant $d$.*

*Proof.* Assume toward a contradiction that there exist distributions $\Pi_1, \ldots, \Pi_N$ defined over the set $\{1, \ldots, B\}$, for $B = d \log_6 N$, such that the unique winner property, holds (and $d$ is some constant). We construct a distribution $\Pi$ over the same set that guarantees the unique winner property (with almost the same success probability[4]) for any $1 \leq \ell \leq N^{1/4}$ participants. By Theorem 3.1 the result follows. The distribution $\Pi$ is defined as follows:

*Choose, uniformly at random $i \in \{1, \ldots, N\}$.*
*Choose a number in $\{1, \ldots, B\}$ according to $\Pi_i$.*

Let $1 \leq \ell \leq N^{1/4}$ participants choose numbers according to $\Pi$. We say that the choice is *good* if each participant $P_j$ chooses a different distribution $\Pi_i$. The first claim says that this happens with a high probability.

CLAIM. *For any $1 \leq \ell \leq N^{1/4}$, the choice is good with probability at least $1 - 1/\sqrt{N}$.*

*Proof.* The probability that a pair of participants $P_{j_1}$ and $P_{j_2}$ choose the same $\Pi_i$ is exactly $1/N$. Therefore, the probability that among $\ell$ participants there exists a pair that choose the same $\Pi_i$ is no more than $\binom{\ell}{2} \cdot 1/N$. As $\ell \leq N^{1/4}$ it implies that the choice is good with probability at least $1 - 1/\sqrt{N}$. □

CLAIM. *For any $1 \leq \ell \leq N^{1/4}$, the probability of having a unique winner is at least $\frac{2}{3} \cdot (1 - 1/\sqrt{N})$.*

*Proof.* Clearly,

$$Pr(\text{unique winner}) \geq Pr(\text{unique winner}|\text{choice is good}) \cdot Pr(\text{choice is good}).$$

The probability that the choice is good is at least $1 - 1/\sqrt{N}$, by the previous claim. In such a case we are exactly in the same situation as in the original lottery. By assumption, this lottery guarantees a unique winner with probability at least $2/3$ for any set of $\ell$ participants; hence, this is certainly true for a random set of $\ell$ participants. The claim follows. □

We defined a lottery for $N^{1/4}$ *identical* players that has the unique winner property. Therefore, by Theorem 3.1, $B \geq c \log_6 N^{1/4}$, for some constant $c$, which completes the proof of the theorem. □

[4] Amplification of the success probability to $2/3$ can be done by picking pairs of numbers according to $\Pi$, which only slightly affects the constants.

## REFERENCES

[1] R. Bar-Yehuda, O. Goldreich, and A. Itai, *On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization*, J. Comput. System Sci., 45 (1992), pp. 104–126.

[2] J. Gil, Y. Matias, and U. Vishkin, *Toward a theory of nearly constant time parallel algorithms*, in Proc. 32nd IEEE Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1991, pp. 698–710.

[3] E. Kushilevitz, and Y. Mansour, *An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks*, in Proc. 12th ACM Symposium on Principles of Distributed Computing, ACM Press, August 1993, pp. 65–74.

[4] E. Kushilevitz, Y. Mansour, M. O. Rabin, and D. Zuckerman, *Lower bounds for randomized mutual-exclusion*, in Proc. 25th ACM Symposium on Theory of Computation, ACM Press, San Diego, CA, May 1993, pp. 154–163.

[5] E. Kushilevitz, and M. O. Rabin, *Randomized mutual exclusion algorithms revisited*, in Proc. 11th ACM Symposium on Principles of Distributed Computing, ACM Press, Vancouver, Canada, August 1992, pp. 275–283.

[6] Y. Matias, and Y. Afek, *Simple and efficient election algorithms for anonymous networks*, in Proc. of WDAG, Lecture Notes in Comput. Sci. 392, Springer-Verlag, New York, pp. 183–194.

[7] M. O. Rabin, *N-Process mutual exclusion with bounded waiting by $4 \log_2 N$-valued shared variable*, J. of Comput. System Sci., 25 (1) (1982), pp. 66–75.

SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 99–118, February 1995

# ALGORITHMS FOR SQUARE ROOTS OF GRAPHS*

YAW-LING LIN† AND STEVEN S. SKIENA‡

**Abstract.** The $n$th *power* ($n \geq 1$) of a graph $G = (V, E)$, written $G^n$, is defined to be the graph having $V$ as its vertex set with two vertices $u, v$ adjacent in $G^n$ if and only if there exists a path of length at most $n$ between them. Similarly, graph $H$ has an $n$th *root* $G$ if $G^n = H$. For the case of $n = 2$, $G^2$ is the *square* of $G$ and $G$ is the *square root* of $G^2$. This paper presents a linear time algorithm for finding the tree square roots of a given graph and a linear time algorithm for finding the square roots of planar graphs. A polynomial time algorithm for finding the square roots of subdivision graphs, which is equivalent to the problem of the inversion of total graphs, is also presented. Further, the authors give a linear time algorithm for finding a Hamiltonian cycle in a cubic graph and prove the NP-completeness of finding the maximum cliques in powers of graphs and the chordality of powers of trees.

**Key words.** square graphs, power graphs, tree square, planar square graphs

**AMS subject classifications.** 05C85, 05C50, 05C12

## 1. Introduction.

Given an undirected graph $G$, the *distance* between vertices $u$ and $v$, denoted by $d_G(u, v)$, is the length of the shortest path from $u$ to $v$ in $G$. The $n$th *power* ($n \geq 1$) of $G$, written $G^n$, is defined to be the graph having $V(G)$ as its vertex set with two vertices $u, v$ adjacent in $G^n$ if and only if there exists a path of length at most $n$ between them, i.e., $d(u, v) \leq n$. Similarly, graph $H$ has an $n$th *root* $G$ if $G^n = H$. For the case of $n = 2$, we say that $G^2$ is the *square* of $G$ and $G$ is the *square root* of $G^2$.

Mukhopadhyay [21] found that a connected undirected graph $G$ with vertices $v_1, \ldots, v_n$ has a square root if and only if $G$ contains a collection of $n$ complete subgraphs $G_1, \ldots, G_n$ such that for all $1 \leq i, j \leq n$:

1. $\bigcup_{1 \leq i \leq n} G_i = G$,
2. $v_i \in G_i$,
3. $v_i \in G_j$ if and only if $v_j \in G_i$.

Characterizations of squares of digraphs were given by Geller [13], and characterizations of $n$th power of graphs and digraphs were given by Escalante, Montejano, and Rojano [7]. Ross and Harary [23] showed that a graph has an unique tree square root (up to isomorphism) if it has a tree as its square root.

Although many properties of the square roots of graphs have been discovered, no efficient algorithms for finding the square root of a graph was known. One reason is that the mathematical characterization of the squares of graphs involve the concept of *clique*, which is an NP-complete problem in general. Furthermore, note that the characterization given by Mukhopadhyay deals with *some n* cliques of $G$ which will have exponentially many candidate cliques. Indeed, Motwani and Sudan [20] showed that recognizing the general square graphs is an NP-complete problem.

Fleischner [8] proved that the square of a biconnected graph is always Hamiltonian. Although the Hamiltonian cycle problem is NP-complete for general graphs

* Received by the editors June 3, 1991; accepted for publication (in revised form) January 31, 1994.

† Department of Information Science, Providence University, 200 Chung Chi Road, Sa-Lu, Taichung Shang, Taiwan (yllin@host1.pu.edu.tw).

‡ Department of Computer Science, State University of New York, Stony Brook, New York 11794-4400 (skiena@sbcs.sunysb.edu). This work was partially supported by National Science Foundation Research Initiation Award CCR-9109289.

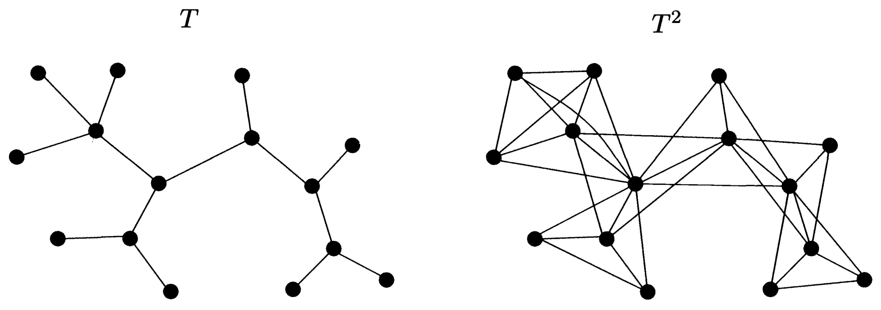


FIG. 1. *A tree and its square.*

[10], the biconnectivity of a graph can be tested in linear time [25]. Thus an efficient algorithm for finding the square root of some special classes of graphs could be useful for finding Hamiltonian cycles in these graphs. Unfortunately, Underground [27] had shown that finding a Hamiltonian cycle in $G^2$ given $G$ is also NP-complete.

In the first part of this paper, we concentrate on the problem of finding the square roots of graphs. We give an $O(m)$ time algorithm for finding the tree square roots of a graph in §2 and a linear time algorithm for finding the square roots of a planar graph in §3. In §4, we propose an $O(m^2)$ time algorithm for inversion of total graphs, which are the square of subdivided graphs.

Optimization problems concerning the square, and more generally the powers, of graphs are discussed in §5. We prove the following results:

- Finding the maximum clique, the minimum independent set, the minimum dominating set, the minimum clique partition, and the chromatic number in $G^k$ are all NP-complete for any fixed $k \geq 1$. Also, we show that when $G$ is a tree, $G^k$ is a chordal graph for any $k \geq 1$, implying these five optimization problems in $G^k$ can all be determined in linear time [14].
- A Hamiltonian cycle in $G^k$, $k \geq 3$, can be found in linear time given $G$. When $G$ is a tree, the existence of a Hamiltonian cycle in $G^2$ can be determined in linear time.

We conclude with open problems.

## 2. Tree square roots.

Tree square roots were first considered by Ross and Harary [23], who showed that they are unique up to isomorphism. Figure 1 presents a tree and its square. In this section, we present an $O(|V| + |E|)$ algorithm for finding the tree square root of a given graph $G = (V, E)$. We will show that the leaves of the tree square root can be readily identified, and, after all leaves of its square root have been found, we can trim all leaves from the graph, resulting in a graph that is the square of a smaller tree. If we repeat this process and build appropriate edges between the different levels of leaves, the ultimate structure of its tree square root follows.

Now consider a graph $T^2 = (V, E)$ which is known to be the square of some tree $T = (V, E')$. The *degree* of a vertex $v$ in $G$, written $\deg_G(v)$, is the size of its neighborhood. A vertex $v$ is a *leaf* (or *endpoint*) of a graph $G$ if $\deg_G(v) = 1$.

The longest distance among all vertices of a graph is called its *diameter*. A tree with diameter less than or equal to two is called a *star*. Note that the square of a star is a complete graph. Recall that a vertex $v$ is simplicial if its neighborhood forms a clique.

LEMMA 2.1. *$v$ is a simplicial point of $T^2$ if and only if $v$ is a leaf of $T$ or $T$ is a star.*

*Proof.* Assume that $v$ is a leaf of $T$. Since $N'_{T^2}(v) = \{u : d_T(u,v) \leq 2\}$, graph $\langle N'_{T^2}(v)\rangle_T$ can be obtained by starting from a leaf $V$ and walking at most two steps in $T$. The resulting graph has a diameter less than or equal to two, i.e., $\langle N'_{T^2}(v)\rangle_T$ is a star. The square of a star is a complete graph, so $\langle N'_{T^2}(v)T^2$ is complete, i.e., $v$ is simplicial. Now assume $v$ is not a leaf and $T$ is not a star. Then there must be a path $(x,v,y,z)$ in $T$. Since $d_T(x,z) = 3$ implies $xz \notin E(T^2)$ and $x,z \in N_{T^2}(v)$, $v$ is not simplicial. □

For each leaf $v$ of $T$, we can partition $N_{T^2}(v)$ into two sets $L_v$ and $M_v$, where $L_v$ denotes all leaves of $T$ in $N_{T^2}(v)$ and $M_v$ the set $N_{T^2}(v) - L_v$. According to Lemma 2.1, a leaf node has the lowest degree of any vertex in its neighborhood of $T^2$. The following lemma characterizes when the degree of a leaf will be strictly less than the degree of its neighboring internal node.

LEMMA 2.2. *Let $T$ be a nonstar tree and $v$ a leaf of $T$, then*

1. *For all $u \in L_v, \deg_{T^2}(u) = \deg_{T^2}(v)$.*
2. *For all $w \in M_v, \deg_{T^2}(w) > \deg_{T^2}(v)$.*

*Proof.* For part 1, let $u \in L_v$. Since $\langle N'_{T^2}(v)\rangle_T$ is a star, we know that $N'_{T^2}(v) = N'_{T^2}(u)$. So $\deg_{T^2}(u) = \deg_{T^2}(v)$. For part 2, let $w \in M_v$. Since $T$ is not a star, without loss of generality, there is either a path $(v,w,x,y)$ or a path $(v,x,w,y)$ in $T$. For either case, $y \in N_{T^2}(w)$ but $y \notin N_{T^2}(v)$. We conclude that $N'_{T^2}(w) \not\subset N'_{T^2}(v)$, Since, by Lemma 2.1, $N'_{T^2}(v) \subset N'_{T^2}(w)$, $\deg_{T^2}(w) > \deg_{T^2}(v)$. □

The *center* of star $T$ is the central vertex $v$ such that for all other vertices $u$ in $T$, $d_T(u,v) = 1$. If $|T| \neq 2$, since there is only one center $v$ in $T$, we can denote $v$ by center$(T)$. Let $K_n$ denote the complete graph of size $n$. The size of $M_v$, $|M_v|$, provides some valuable information.

LEMMA 2.3. *Let $v$ be a leaf of $T$, then*

1. *if $|M_v| \leq 1$, $T$ is a star;*
2. *if $|M_v| = 2$,* say *$M_v = \{x,y\}$, then $xy \in E(T)$;*
3. *if $|M_v| \geq 3$, $(v, \text{center}(M_v)) \in E(T)$.*

*Proof.* For part 1, either $|M_v| = 0$ meaning $T = K_2$ or $|M_v| = 1$, where again, $T$ is a star. For part 2, let $M_v = \{x,y\}$. Since $v$ is a leaf, Lemma 2.1 showed that $\langle N'_{T^2}(v)\rangle_T$ is a star. Since $|N_{T^2}(v)| \geq |M_v| = 2$ meaning $|N'_{T^2}(v)| \geq 3$, center$(\langle N'_{T^2}(v)T)$ is well defined. Further, center$(\langle N'_{T^2}(v)T) \notin L_v$. So, without loss of generality, we can let $x$ be the center of $\langle N'_{T^2}(v)\rangle_T$. Again, since $\langle N'_{T^2}(v)\rangle_T$ is a star, $y$ must be connected to $x$. So $xy \in E(T)$. For part 3, $|M_v| \geq 3$. Since $\langle N_{T^2}(v)\rangle_T$ is a star and $v$ is the center, we know that center$(\langle N_{T^2}(v)\rangle_T) =$ center$(\langle M_v\rangle_T)$. Clearly, $(v, \text{center}(\langle N_{T^2}(v)\rangle_T)) \in E(T)$. □

Given a tree $T$, we can delete all the leaves of $T$ resulting in a smaller tree $T'$. This trimming operation defines a function trim$(T) = T'$. The trimming operation can be repeated until the remaining subgraph $T'$ is empty. Since each vertex $v$ in $T$ will eventually be trimmed, we can associate an integer with $v$ specifying the number of trimming operation taken before $v$ becomes a leaf. This function from $V(T)$ to $[0..\lfloor(n-1)/2\rfloor]$ is recursively defined as follows:

$$\text{level}(v) = \begin{cases} 0 & \text{if } v \text{ is a leaf of } T \\ k & \text{if } v \text{ is a leaf of } \text{trim}^k(T) \end{cases}$$

Lemma 2.1 shows that a leaf has the lowest degree in its neighborhood. The following lemma shows the opposite direction. A vertex having the lowest degree in its neighborhood is not always a leaf but we can easily distinguish a low-degree internal node from a leaf.

LEMMA 2.4. *Let $T$ be a nonstar tree and $v$ a nonleaf vertex of $T$. If* $\deg_{T^2}(v) \leq \min\{\deg_{T^2}(u) : u \in N_{T^2}(v)\}$, *then* $\text{level}(v) > 2$ *and for all* $x \in N_{T^2}(v), N_{T^2}(v) \not\subset N'_{T^2}(x)$.

*Proof.* Since $v$ is not a leaf, $\text{level}(v) \geq 1$. Suppose $\text{level}(v) \leq 2$, then there must exist a $u$, a leaf of $T$, such that $uv \in E(T^2)$. By Lemma 2.2, we know that $\deg_{T^2}(u) < \deg_{T^2}(v)$, a contradiction. Let $x \in N_{T^2}(v)$. The second part has two cases:

*Case* 1. $d_T(x, v) = 1$. Since $T$ is not a star and $\text{level}(v) > 2$, there exists a path $(x, v, y, z)$ in $T$. Now, $z \in N_{T^2}(v)$ but $z \notin N'_{T^2}(x)$.

*Case* 2. $d_T(x, v) = 2$. Since $v$ is not a leaf, there must be a path $(x, y, v, z)$ in $T$. Again, $z \in N_{T^2}(v)$ but $z \notin N'_{T^2}(x)$. □

LEMMA 2.5. *Given a tree square $T^2 = (V, E)$, we can identify all leaves of $T$ by examining $T^2$ in a total time of $O(|E|)$ time.*

*Proof.* Assume that the graph $T^2$ is given in form of adjacency lists. Let $|V| = n$ and $|E| = m$. First, we can calculate the degree of each vertex totally in $O(m)$ time. Also, we will maintain a boolean array $B$, indexed by the vertices $V$, with size $n$, and initialize it by all zeros in $O(n)$ time. Then, for each vertex $v$ of $T^2$, first we will check whether $v$ has the lowest degree among its neighborhoods in $O(\deg_{T^2}(v))$ time. If not, $v$ cannot be a leaf in $T$; otherwise, we can mark $B_u = 1$ for each $u$ in $N_{T^2}(v)$ within $O(\deg_{T^2}(v))$ time, and choosing $u$ such that $u$ has the lowest degree in $N_{T^2}(v)$, we can check whether $N_{T^2}(v) \subset N_{T^2}(u)$ in $O(\deg_{T^2}(u))$ time, by polling each elements of $N_{T^2}(u)$ in $B_v$. If this condition does not hold, by Lemma 2.5, no vertex of $N'_{T^2}(v)$ is a leaf; otherwise, $v$ and those vertices in $N_{T^2}(v)$ with same degree as $v$ are all leaves of $T$. Note that the total time spent on admitting leaves is bounded by $\Sigma_{\text{level}(v) \in \{0,1\}} \deg_{T^2}(v)$. Also note that, since we are picking $u$, the lowest degree vertex in $N_{T^2}(v)$, the time spent on asking $N_{T^2}(v) \subset N_{T^2}(u)$ for those failed (nonleaf) $v$ is bounded by $\deg_{T^2}(x)$ for any vertex $x \in N_T(v)$. Further, if $v_1$ and $v_1$ both failed this test, it implies that $d_T(v_1, v_2) > 2$, and, for each leaf $y$ of $T$, both $d(y, v_1)$ and $d_T(y, v_2)$ are greater than two. That is, the amount of time failing the $N_{T^2}(v) \subset N_{T^2}(u)$ test is bounded by $\Sigma_{uv \in F} \deg_{T^2}(u) + \deg_{T^2}(v)$ for some $F \subset E(T)$, where $F$ is an independent set of edges, i.e., a matching. It follows that the total time is bounded by $\Sigma_{v \in T^2} \deg_{T^2}(v)$ or $O(|E|)$. □

THEOREM 2.6. *The tree square root of a graph can be found in $O(m)$ time, where $m$ denotes the number of edges of the given tree square graph.*

*Proof.* Let the given graph be $G = (V, E)$ with $|V| = n$ and $|E| = m$. Now assume that there is a tree $T$ such that $G = T^2$. By Lemma 2.5, we can identify all leaves of $T$ in $O(m)$ time. Then we repeat the following procedures.

Let $LF(T)$ denote the set of all leaves of tree $T$. Trimming $T^2$ by removing all vertices in $LF(T)$ results in a proper induced subgraph of $T^2$, say $T'^2$. This process takes $O(\sum_{v \in LF(T)} \deg_G(v))$ time. For all leaf $v$, we can partition its neighborhood into two sets $L_v$ and $M_v$ totally in $O(\sum_{v \in LF(T)} \deg_G(v))$ time. If for all $v$, $M_v < 2$, we will conclude that $T$ is a star and terminate the process. Else, if for all $v$, $M_v > 2$, then $G$ is not a square of a tree; otherwise, we have some $v \in LF(T)$ such that $M_v = \{x, y\}$, a set of size two. According to the second part of Lemma 2.3, we can report that edge $xy$ contains in $T$. Further, we claim that

$$LF(T') \subset \bigcup_{v \in LF(T) \wedge |M_v|=2} M_v.$$

To prove this claim, assume that $p$ is a leaf of $T'$, meaning there exists exactly one vertex, say $q$, in $T'$ such that $pq \in T'$. Since $p$ is a leaf, there must be a leaf $v$ in $T$ adjacent to $p$; otherwise, $p$ will be a leaf of $T$ meaning $p \notin T'$, a contradiction. That is, $\{p, q\} = M_v$. Our claim is proven.

Our claim means that we can find all the leaves of $T'$ in $O(|LF(T)|)$ time by Lemma 2.4, since we will now only need to check which vertex of $\{p, q\}$ has lower degree. Treat $T'$ as $T$ and repeat the process above until $T'^2$ becomes a complete graph, implying $T'$ a star. Each pass takes $O(\sum_{v \in LF(T)} \deg_G(v))$ to identify the leaves of $T$ and establish the set of possible leaves of $T'$. If $G$ is a tree square, eventually each vertex $v$ of $G$ becomes a leaf during the trimming process. That is, we will spend $O(\sum_{v \in V(G)} \deg_G(v))$ or $O(m)$ total time finding all *internal* edges (those edges which are not adjacent with leaves) of $T$.

At this point, only the original leaves have not yet been linked to other vertices. This problem can be solved by recording the internal nodes, $M_v$, of each original leaf $v$ of $T$. Along with the trimming process, the trimmed level of each vertex $v$ of $T$, level($v$), is recorded. In detail, we maintain a counter, say $k$, initialized by zero. Every time the trimming process is repeated, $k$ is increased by one. Each time a vertex $v$ is eliminated from the tree, we will record level($v$) as $k$. Note that this fulfills our definition of *level* function and takes a total of $O(n)$ time.

Recall that the structure of the internal tree has already been determined at this point. If $G$ is complete, $T$ is just a star, which we have already dealt with when we tried to find the leaves. Now we may assume that $G$ is not complete, meaning $|M_v| \geq 2$. More precisely, the size of $M_v$ can be as follows.

*Case* 1. $|M_v| = 2$ or $M_v = \{x, y\}$. If level($x$) = level($y$) = 1, there will be exactly two set of leaves, say $L_1$ and $L_2$, and we can either link $x$ to $L_1$ and $y$ to $L_2$ or $y$ to $L_1$ and $x$ to $L_2$. Otherwise, exactly one of $x$ and $y$ is in the level 1. Without loss of generality, we say level($x$) = 1, then $v$ and $x$ are linked. This case takes constant time.

*Case* 2. $|M_v| \geq 3$. By Lemma 2.3, we shall link $v$ to center($M_v$). Now we show how the center($M_v$) is found. First choose two arbitrary $x, y$ from $M_v$. If $xy \in T'$, pick another $z$ from $M_v$, and since $\langle M_v \rangle_{T'}$ is a star, $z$ is adjacent to, say, $x$. That is, $x$ is the center, if $xy \notin T'$. Keep picking $z$ from $M_v$ and eventually, since $\langle M_v \rangle_{T'}$ is a star, we will find a $z$ such that $zx$ and $zy \in T'$. $z$ is the center and the search takes $|M_v|$ or $O(\deg_G(v))$ time.

The process of linking the leaves of $T$ to $T'$ takes $O(\sum_{v \in LF(T)} \deg_G(v))$, also bounded by $O(m)$. After the tree $T$ has been found, we will now double check whether $G = T^2$. This can be done by examining each vertex $v$ of $T$ and seeing whether $\bigcup_{u \in N_T} N_T(u) \cup N_T(v) = N_G(v)$ in $O(\deg_G(v))$ time, since the time needed for finding $\bigcup_{u \in N_T} N_T(u)$ is bounded by $2 \deg_G(v)$ using a boolean array, as we have shown in Lemma 2.5. So totally, it will take $O(m)$ time to finish the last check.

Now we have completed the analysis and description of the algorithm. In summary, it takes $O(m)$ time to identify all leaves of $T$, and another $O(m)$ time to trim the tree and find the structure of $T$. Finally, to make sure $T$ is indeed the square root of $G$, it needs another $O(m)$. We now conclude that the tree square root of a square graph can be found in linear time. $\square$

**3. Square roots of planar graphs.** In this section, we present an $O(n)$ algorithm for finding the square root of a given planar graph based on the characterization of planar squares found by Harary, Karp, and Tutte [16] and the linear time triconnected components algorithm given by Hopcroft and Tarjan [18].

A vertex $v$ is an *articulation vertex* (or *cut point*) of a connected graph $G$ if the removal of $v$ from $G$ results in a disconnected graph. A *bridge* of a connected graph $G$ is an edge $e \in E(G)$ whose removal disconnects $G$. A *block* or *biconnected component* of $G$ is a maximal connected induced subgraph of $G$ without articulation vertices. An *$n$-connected* graph $G$ is a graph with at least $n+1$ vertices such that the removal of any $n-1$ vertices does not disconnect $G$ [26]. Note that the complete graph $K_n$ is $(n-1)$-connected but not $n$-connected. A graph is *biconnected* (*triconnected*) if it is 2-connected (3-connected.)

THEOREM 3.1 (Harary, Karp, and Tutte [16] ). *A graph $G$ has a planar square if and only if*

1. *every point of $G$ has degree less than or equal to three,*
2. *every block of $G$ with more than four points is a cycle of even length, and*
3. *$G$ does not have three mutually adjacent articulation vertices.*

The following discussion concerns the definition of *triconnected components* given by Hopcroft and Tarjan [18]. Let $G = (V, E)$ be a biconnected graph. A pair of vertices $\{a, b\}$ in $G$ is called a *separation pair* of $G$ if the removal of $\{a, b\}$ disconnect $G$. Let $S = \{H_i : H_i$ is a connected component of $\langle V - \{a, b\}\rangle_G\}$. and $\mathcal{S} = \{G_i = \langle V(H_i) \cup \{a, b\}\rangle_G$ with $ab$ linked: for each $H_i \in S\}$. Each element $G_i$ of $\mathcal{S}$ is called the *split graph* of $G$ with respect to $\{a, b\}$. The newly added edge $ab$, called *virtual edge*, is not necessarily in $G$. Given a biconnected but not triconnected or complete graph $G$, we can find one of its separation pairs, $\{a, b\}$ and split it into split graphs. If any split graph $G_i$ of $G$ is not triconnected or complete, the splitting process is repeated until each split graph is either triconnected or complete. The ultimate split graphs are called the *split components* of $G$. The split components are not necessarily unique. Given a set of split graphs, partition it into two classes: $\mathcal{A} = \{G_i : G_i$ a triconnected split component$\}$ and $\mathcal{T} = \{G_i : G_i$ a split component and $G_i = K_3\}$. *Merge* those split components in $\mathcal{T}$ that share edges into a polygon. Let $\mathcal{P}$ denote the collection of the resulting in components. $\mathcal{A} \cup \mathcal{P}$ is called the *triconnected components* of $G$. Triconnected components are unique and the following theorem applies.

THEOREM 3.2 (Hopcroft and Tarjan [18] ). *The triconnected components of a graph can be found in $O(m + n)$ time.*

An *endline* of $G$ is an edge $uv$ of $G$ such that either $u$ or $v$ is a leaf. A *burr* of $G$ is a maximal connected (induced) subgraph of $G$ in which every bridge is an end line. By removing all leaves of a burr $B$ in $G$, the remaining subgraph $B'$ is called the *central block* of $B$. Since $B$ does not contain inner bridge, $B'$ must be a block or a single vertex. As shown in Theorem 3.1, if $G^2$ is a planar graph, then for each burr $B$ of $G$, either of the following cases is true.

*Case* 1. The central block $B'$ is an even length cycle with more than four vertices. Let $C_n$ denote a chordless cycle of length $n$. That is, $B' = C_{2n}$, for some $n$ greater than 2. Further, each vertex of $B'$ is adjacent to at most one endline.

*Case* 2. The central block $B'$ is a block of size at most four. Each vertex of $B'$ can be linked with at most one leaf, providing it does not violate the conditions of Theorem 3.1. Such kind of burr has at most eight vertices. That is, a burr consists of a central block $C_4$ with each vertex adjacent to exactly one leaf.

*Case* 3. The central block $B'$ consists of a single vertex $v$, which can be adjacent to at most three other leaves. That is, $B$ is a star of size at most four.

LEMMA 3.3. *The square of a burr is triconnected or complete. Further, each triconnected component of a planar square $G^2$ is the square of a burr in $G$.*

*Proof.* Given a graph $G = (V, E), U \subset V, v \in V$, and $x \notin V$, let $G - U$ denote $\langle V - U \rangle_G$ and $G \cup ax$ denote the graph $(V \cup \{x\}, E \cup ax)$. First we prove that the square of a burr is triconnected or a complete graph with at most four vertices. Given a burr $B$, we analyze the cases of different burrs based on the size of its central block $B'$.

*Case* 1. $B'$ is a block of size greater or equal to three. We claim that:
(1) *$B'^2$ is triconnected or $K_3$.* If $|B'| = 3$, $B'^2 = K_3$. If $|B'| = 4$, $B'^2 = K_4$ is triconnected. Otherwise, $B' = C_{2n}$ for some $n > 2$. Consider each pair of vertices, $\{u, v\}$, of $C_{2n}$. If $uv \in E(C_{2n})$, then $C_{2n} - \{u, v\}$ is still connected. If $uv \notin E(C_n)$, without loss of generality, let $C_{2n} = (P_1, u, P_2, v)$, where $P_1$ and $P_2$ represent two different paths in $C_{2n}$. Note that in $C_{2n}^2$, there exists at least one edge that connects $P_1$ and $P_2$. That is, $C_{2n}^2 - \{u, v\}$ is connected or $B'^2$ is triconnected.
(2) *For each end line $vx$ of $B$, $(B' \cup xv)^2$ is triconnected.* Let $x$ be the leaf of $B$ meaning $v \in B'$. For each $w \in B'$, delete $\{x, w\}$ from $(B' \cup xv)^2$. This results in a graph $B'^2 - \{w\}$, which is still connected. Otherwise delete $\{u, v\} \subset V(B')$ from $(B' \cup xv)^2$. $B'^2 - \{u, v\}$ is still connected as shown in (1). Note that $x$ is adjacent to at least three vertices in $B'^2$. So there must exist another vertex $w \in V(B') - \{u, v\}$ that is adjacent to $x$ at the $(B' \cup vx)^2$.

By claims (1) and (2) stated above, we can easily see that if $B \neq K_3$, $B$ is triconnected. The only case we have not mentioned is picking the vertices pair $\{u, v\}$ such that both of them are the leaves, but clearly $B - \{u, v\}$ is still connected.

*Case* 2. The central block is a single vertex. Such a burr must be a star of size at most four. The square of stars are complete as shown in Lemma 2.1.

Now we proceed to prove that each triconnected component of $G^2$ is the square of a burr. First we note that vertex $\{u, v\}$ of $G^2$ is a separation pair of $G^2$ if and only if $uv$ is the nonendline bridge of $G$. So the split process will be repeated until the resulting graphs are triconnected or complete graphs with at most three vertices. Suppose there is a triconnected component $H^2$ in $G^2$, such that $H$ is not a burr of $G$. By the definition of burr, there exists a nonendline bridge $uv$ in $H$. Because the removal of vertices pair $\{u, v\}$ disconnects $H^2$, which means that the splitting process has not completed, we conclude that $H^2$ is not the triconnected component of $G^2$. □

Theorem 3.2 shows that $O(m+n)$ time suffices to find all triconnected components of a given graph. For our case of planar graph, because the number of edges $m \leq 3n - 6$, we can find all triconnected components in $O(n)$ time.

For $G^2$ to be planar, each large burr in $G$ must be an even-length cycle plus some endlines in general. Otherwise, the central block will just be a finite graph with size less than five. The following lemma characterizes the leaves of a large burr.

LEMMA 3.4. *Given a planar square graph $G^2$ and a burr $B$ of $G$ such that $|B| > 5$, $v$ is a leaf of $B$ if and only if* $\deg_{B^2}(v) = 3$.

*Proof.* Since $|B| > 5$ and $B^2$ is planar, the central block $B'$ must have at least four vertices. Otherwise the condition of Theorem 3.1 will be violated. First we show that for each nonleaf $v$ of $B$, $\deg_{B^2}(v) > 3$. If $B'$ is an even-length cycle $C_{2n}$ such that $n \geq 3$, then $\deg_{B^2}(v) \geq 4$. Otherwise, the central block is a block of size four. There are at least two leaves outside this size-four block. Again, $\deg_{B^2}(v) \geq 4$.

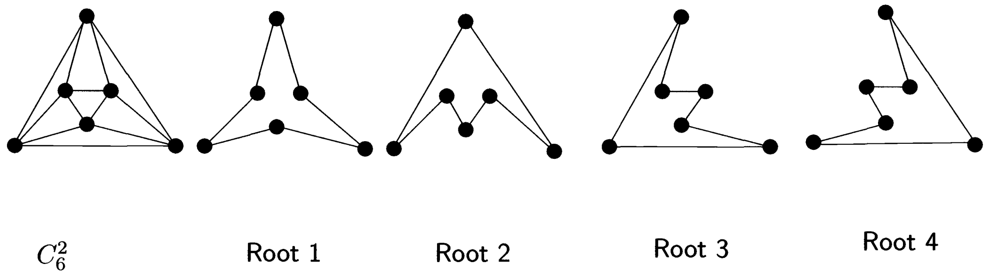


FIG. 2. *Four different roots of* $C_6^2$.

Now we show the each leaf $v \in B$ has degree three in $B^2$. Let $uv \in B$ meaning $u \in B'$. Since $B^2$ is planar, $\deg_B(u) \leq 3$. Since $B'$ is a block, $\deg_{B'}(u) \geq 2$, meaning $\deg_B(u) \geq 3$. Since $\deg_B(u) = 3$, $\deg_{B^2}(v) = 3$. □

Now we show that we can efficiently find the original burr by given its square.

LEMMA 3.5. *Given a planar graph* $B^2$, *where* $B$ *is a burr, the structure of* $B$ *can be computed in* $O(|B|)$ *time.*

*Proof.* Consider various sizes of $|B|$.

*Case* 1. $|B| > 5$. By Lemma 3.4, we can identify all leaves of $B$ within $O(|B|)$ time by examining the degree of each vertex. Let $L$ be the set of leaves in $B$. There can be two possibilities.

*Case* 1.1. $|L| = 0$. Then $B = C_{2n}$ for some $n \geq 3$. If $B = C_6$, there are four distinct labeled graphs whose square is equal to $C_6^2$ as shown in Fig. 2. For $n > 3$, the identification of $C_{2n}$ is unique since edge $uv \in C_{2n}$ if and only if $|N'_{C_{2n}^2}(u) \cap N'_{C_{2n}^2}(v)| = 3$. In linear time we can trace the exact labeling of $C_{2n}$

*Case* 1.2. $|L| > 0$. Choose an arbitrary vertex $v$ from $L$ and delete all vertices $L - \{v\}$ from $B$. The resulting graph will just be a graph of structure $(v \cup C_{2n})^2$ for some $n \geq 2$. Since we know that the vertex $v$ is its leaf, one of vertices $N_B(v) = \{a, b, c\}$ is adjacent to $v$ in $B$. It is easy to verify that $av \in B$ if and only if $|N_{B^2}(a) \cap N_{B^2}(b)| = |N_{B^2}(a) \cap N_{B^2}(c)| = 3$. Say $av \in B$, meaning $ab$ and $ac \in B$. Since for $x \neq v$ or $c$ and $x \in N_{B^2}(b), bx \in B$ if and only if $ax \in B$, once we know $ab, ac \in B$, we can trace the cycle out by this scheme. This process can be repeated until the whole structure of $B$ is revealed. It can be done in $O(|B|)$.

*Case* 2. $|B| = 5$. For this finite graph, all possible structures of $B$ are as follows.

*Case* 2.1. $B =$ ($C_4$ plus an endline). This case there are six possible arrangements of vertices to constitute $B$ as shown in Fig. 3.

*Case* 2.2. $B =$ ($K_3$ plus two endlines). Again, there are six possible arrangements of vertices to constitute $B$ as shown in Fig. 4. One thing interesting about this case is that there is only one possible (up to isomorphism) square burr for size five. We have shown it has 12 distinct labeled roots and two different (up to isomorphic) roots.

*Case* 3. $|B| = 4$. All possible structures of $B$ are a star with size four and possibly adding any edges or $C_4$.

*Case* 4. $|B| = 3$. $B$ is a path of length two or $K_3$.

*Case* 5. $|B| \leq 2$. $B$ is $K_1$ or $K_2$.

It has been shown that for a large burr, the structure of $B$ can be revealed in $O(|B|)$, and for the finite size of $B$, the structure of $B$ can be found in constant time. We conclude the structure of $B$ can be determined in $O(|B|)$ time. □

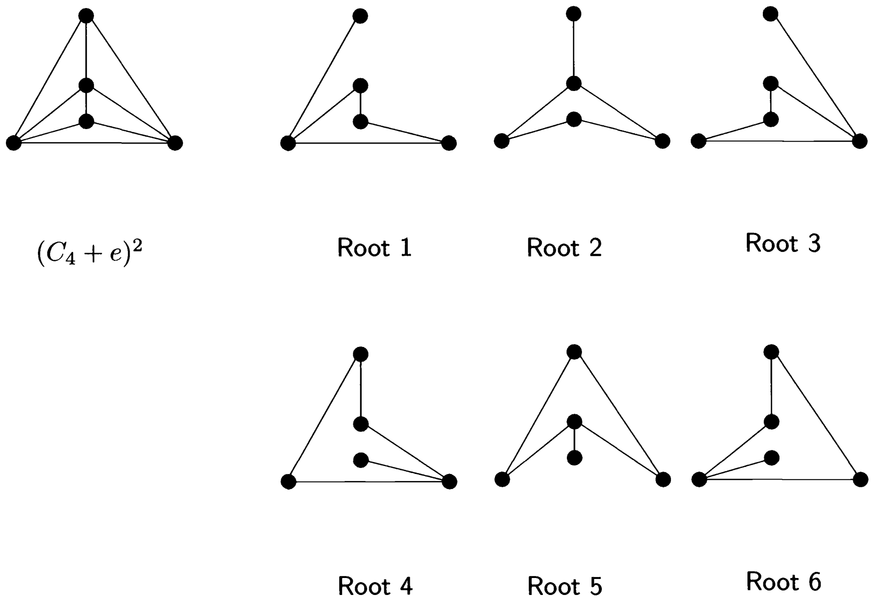


FIG. 3. $(C_4 + e)^2$ *and its six different roots.*

Given a square burr $B^2$, we have just shown that the structure of $B$ is not necessarily unique. Also note that not all combinations of arbitrary burr roots constitute the root of $G^2$. So to complete the determination of the square root of $G^2$, we must identify all inner bridges of $G$. The following lemma shows that it can also be done efficiently.

LEMMA 3.6. *Given $G^2$ a planar graph, all inner bridges of $G$ can be found within $O(n)$ time.*

*Proof.* Without loss of generality, we assume that $G$ is connected. Say $G^2$ consists of $k$ square burrs, $B_i^2, 1 \leq i \leq k$. By Lemma 3.3 and Theorem 3.2, these square burrs can be identified in $O(n)$ time. If $k = 1$, then there is no inner bridge in $G$. We will now only consider the case that $k > 1$.

Let $\mathcal{B} = \{B_i^2 : B_i \text{ is a burr of } G\}$ and $\mathcal{E} = \{(B_i^2, B_j^2) : |B_i^2 \cap B_j^2| = 2\}$. We claim $(\mathcal{B}, \mathcal{E})$ defines a tree. Note that $(\mathcal{B}, \mathcal{E})$ is connected. Suppose there exists a cycle $(B_1^2, B_2^2, \ldots, B_m^2)$ in $(\mathcal{B}, \mathcal{E})$ where $m$ is the length of the cycle . Then $\cup_{1 \leq i \leq m} B_i$ is a burr, which violates the assumption that $B_i$ is a burr. Therefore $(\mathcal{B}, \mathcal{E})$ must be a tree. It implies that $k \leq n$.

Recall that, for each two burrs $B_i$ and $B_j$ of $G$, either $B_i^2 \cap B_j^2 = \emptyset$ or $B_i^2 \cap B_j^2 = \{u, v\}$, which implies $uv$ is the inner bridge between $B_i^2$ and $B_j^2$. Otherwise, $|B_i^2 \cap B_j^2| = 1$, and there is an intermediate burr $B_l^2$, which is either a $K_3$ or $K_4$, such that $|B_i^2 \cap B_l^2| = |B_l^2 \cap B_j^2| = 2$. Now, repeat the following process for each $B_i^2$.

First, for each burr $B_i^2$, insert the number $i$ to each of its vertices, $v$. Note that each vertex can be associated with at most four burrs since the degree of each vertex in the root can be at most three. Denote the list of burrs associated with vertex $v$ by $A_v$.

Now, for each vertex $v$, if the number of square burrs associated with $v$ is at least two, visit every vertex $u$ in the neighborhood of $v$. If $|\{A_u \cap A_v\} = \{i, j\}| = 2$, then

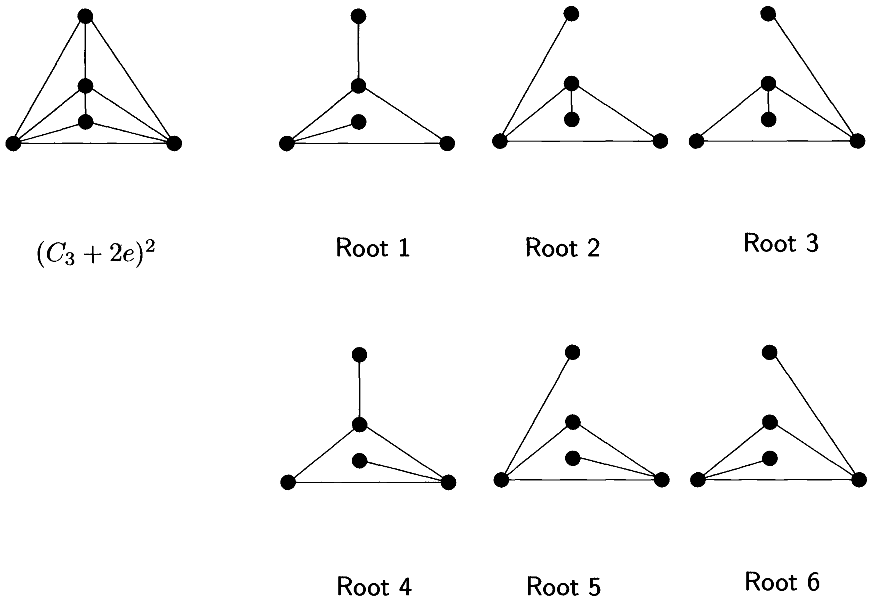


FIG. 4. *$(C_3 + 2e)^2$ and its six different roots.*

we can claim that $uv$ is the inner bridge between two burrs $B_i$ and $B_j$.

The first step of the strategy just mentioned can be done in $O(|B_i|)$ for each $B_i^2$. Recall that, for each $v$ in $G$, $|A_v| \leq 4$. Thus the second step can be done in constant time for each vertex $v$. Totally, we will need $\sum_{1 \leq i \leq k} |B_i| + \sum_v 1$, which is exactly $O(n + 2k - 2) + O(n) = O(n)$ operations (remember that $(\mathcal{B}, \mathcal{E})$ is a tree). This completes our proof. □

THEOREM 3.7. *The square root of a planar graph $G^2$ can be found in $O(n)$ time.*

*Proof.* By Lemma 3.3, the square burrs of $G^2$ can be found in $O(n)$ time. By Lemma 3.6, the inner connection between those square burrs can be found in $O(n)$ time. By Lemma 3.5, the structure of each burr $B_i$ can be found in $O(|B_i|)$ time. Since we have shown in Lemma 3.6 that $\sum_i |B_i| < 3n$, the result follows. □

**4. Inversion of total graphs.** Given a graph $G = (V, E)$, its *total graph* [15], $T(G)$, has vertex set $V \cup E$ with two vertices of $T(G)$ adjacent whenever they are neighbors in $G$. If $uv$ is an edge of $G$, and $w$ not a vertex of $G$, then $uv$ is *subdivided* when it is replaced by two edges $uw$ and $wv$. If every edge of $G$ is subdivided, the resulting graph is the *subdivision graph* $S(G)$ [15]. Behzad [2] showed that, given a graph $G$, the total graph $T(G)$ is isomorphic to the square of subdivision graph $S(G)$. Behzad and Radjavi [3] showed that the two graphs $G, H$ are isomorphic to each other if and only if $T(G) \cong T(H)$. We give a polynomial time algorithm for the inversion of total graphs by reducing it into finding square roots of the squares of subdivision graphs. Recently, we have learned of a similar result by Gavril [12].

Given a graph $G = (V, E)$ and its subdivision graph $S(G)$, we call the set of newly added vertices $W$, such that $V(S(G)) = V \cup W$. By Behzad's result, we know that $T(G) = [S(G)]^2$. Given $v \in V$, we call $I(v) = N_{S(G)}(v) \subset W$, the *inner neighborhoods* of vertex $v$, and $O(v) = N_G(v) \subset V$, the *outer neighborhoods*. Note that $N_{T(G)}(v) = N_{[S(G)]^2}(v) = I(v) \cup O(v)$. We observe the following.

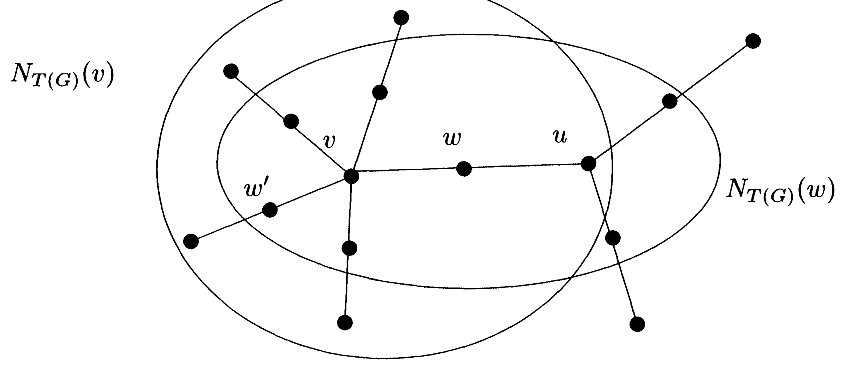


FIG. 5. $I(v) = N_{T(G)}(v) \cap N_{T(G)}(w) - \{u\}$ *in total graph.*

LEMMA 4.1. *Let $uv$ be an edge of $G$ and $w$ the subdivided vertex of $uv$ implying $uw, wv \in S(G)$. Then*

(i) $I(v) = N_{T(G)}(v) \cap N_{T(G)}(w) - \{u\}$.

(ii) *For each pair $(w', u') \in I(v) \times O(v), w'u' \in S(G)$ if and only if $\{u'\} = N_{T(G)}(w') \cap N_{T(G)}(v) - I(v)$.*

*Proof.* (i) Since, for each $x \in O(v) - \{u\}, d_{S(G)}(w, x) = 3$, it follows that $I(v) = N_{T(G)}(v) \cap N_{T(G)}(w) - \{u\}$ as illustrated in Fig. 5.

(ii) The only if part is trivial by (i). Now assume that $\{u'\} = N_{T(G)}(w') \cap N_{T(G)}(v) - I(v)$. Note that $d_{S(G)}(u', w') \leq 2$ and $d_{S(G)}(u', v) = 2$. It follows that $u'v' \in S(G)$. □

Given a graph $G$, recall that a *maximal clique* of $G$ is an induced complete subgraph of $G$, such that no other vertex in $G$ can be added in to form a larger clique. We find a way to tell whether an edge $uv$ is in the square root graph by examining these maximal cliques in the square containing both vertices $u$ and $v$.

PROPOSITION 4.2. *Given a graph $G = (V, E)$ and an edge $uv$ of $G$, there are at most two maximal cliques of size three containing both $u$ and $v$ in $G^2$.*

*Proof.* Let $K = \{u, v, w\}$ be a maximal clique in $G^2$. Note that it is not possible that $d_G(u, w) = d_G(v, w) = 2$ in $G$ because that will imply that $K$ is not maximal. So, without loss of generality, we can assume that $uw$ is an edge of $G$. Note that $v$ and $w$ are the only vertices allowed to be adjacent to $u$, since otherwise $K$ will not be maximal. It is possible that $v$ can be adjacent to a vertex $w' \neq w$ such that $N_G(v) = \{u, w'\}$ implying $\{u, v, w'\}$ also being a maximal clique, but that is all possible maximal cliques of size three we can have containing both $u$ and $v$. □

PROPOSITION 4.3. *Given a graph $G = (V, E)$, an edge $uv$ of $G$, and $w$ the subdivided vertex of $uv$ in $S(G)$, then* $\deg_{T(G)}(v) = 2$, *and* $\deg_{T(G)}(w) = \deg_{T(G)}(u) + \deg_{T(G)}(u)$ *both $u$ and $v$ in $G^2$.*

*Proof.* The proof of this proposition is based on the fact that $T(G) = [S(G)]^2$, and we will leave it to the reader. □

THEOREM 4.4. *The inversion of total graphs can be done in $O(m^2)$ time where $n$ and $m$ denote the number of vertices and edges of the given graph, respectively.*

*Proof.* Given a graph $H$, our goal is to present an algorithm for finding a graph $G$ such that $H = T(G) = [S(G)]^2$ in $O(|E(H)|^2)$ time. Without loss of the generality, we will assume $H$, and thus $G$, is connected. Our strategy is to pick a triple$\{u, v, w\}$

in $H$ such that $vw$ and $wu$ are in $S(G)$, so that we can use Lemma 4.1 to trace the structure of $S(G)$, which is the square root of $H$, and then the graph $G$ is easily found from $S(G)$.

Assume that $H$ contains $n$ vertices and $m$ edges. First we can construct the adjacency matrix of $H$ in $O(n^2)$ time. Now consider the graph $G$ we are looking for. By Proposition 4.3, if a vertex $v$ has the lowest degree in $G$, so does $v$ in $T(G)$, that is, the given graph $H$. So we begin by sorting all vertices of $H$ according to their degrees in $H$. Since the maximum degree of all vertices in $H$ is bounded by $n$, by using the bucket sort method, the sorting process can be done in $O(m)$. Denote the lowest degree of all vertices in $H$ by $\delta$. It is clear that some vertex $v$ in $H$ with degree $\delta$ will be also a vertex of $G$ (with degree $\delta/2$), and there is at least a vertex $w \in N_{T(G)}(v)$ such that $vw$ is also in $S(G)$.

Let $\alpha$ denote the number of degree $\delta$ vertices in $T(G)$. It is clear that the number of possible candidates of $vw$ in $T(G)$ will be $\alpha\delta = O(m)$. To apply the result of Lemma 4.1, we also need to find a vertex $u$ in $T(G)$ such that $vw$ and $wu$ are all in $G$. Fortunately, Proposition 4.3 assures us that the number of candidates of such $u$ is at most two. To find all maximal cliques of size three containing $vw$ in $T(G)$, we can examine the set $A = N_{T(G)}(v) \cap N_{T(G)}(w)$ in $O(\delta)$ time and eliminate any pair $\{x, y\} \subset A$ from it resulting a smaller subset $A'$ if $xy \in T(G)$, which can be done in $O(\delta^2)$ time, which is still bounded by $O(m)$. Again, note that the size of $A'$ is at most two by Proposition 4.3.

Now we have a triple $\{u, v, w\}$ forming a maximal clique in $T(G)$ and suggesting that $vw$ and $wu$ are two edges of $S(G)$. So we can apply Lemma 4.1 to trace the structure of $S(G)$ in $O(m+n)$ time by a breadth-first-searching method described as follows. First, for each vertex $w'$ in the inner neighborhoods, $I(v)$ that can be found in $\deg_{T(G)}(v)$ time, we know that $vw' \in E(S(G))$ by part (i) of Lemma 4.1. Also, for each $u' \in N_{T(G)}(w')$, we know that $u'w' \in E(S(G))$ if $u'$ is also in $N_{T(G)}(v)$ but not in $I(v)$, which we can tell in $O(\deg_{T(G)}(w'))$ time. By propagating this procedure to each vertex in $T(G)$ by using a breadth-first-search method, it takes a total of $O(\sum_{v \in T(G)} O(\deg_{T(G)}(v))) = O(m)$ time.

To avoid estimating an incorrect $\{u, v, w\}$ triple, we will need to double check whether $H = [S(G)]^2$. This can be done by examining each vertex $v$ of $S(G)$ and see whether $\bigcup_{u \in N_{S(G)}} N_{S(G)}(u) \cup N_{S(G)}(v) = N_H(v)$ in $O(\deg_H(v))$ time, since the time needed for finding $\bigcup_{u \in N_{S(G)}} N_{S(G)}(u)$ is $2\deg_G(v)$ if $v \in G$, or $\deg_G(x) + \deg_G(y)$ if $v$ is the subdivided vertex of an edge $xy$ of $G$, as shown in Proposition 4.3.

Now we have completed the analysis and description of the algorithm. In summary, it will spend at most $O(m)$ time for each possible triple $\{u, v, w\}$, whereas the number of candidate triples is bounded by $O(\alpha\delta)$. Since we need the time to initialize the adjacency matrix of $H$, the total time bound will be $O(n^2 + \alpha\delta m)$. In the worst case, it will take $O(m^2)$ time, although in most of the cases that $\alpha\delta$ will be much smaller than $m$ thus the time complexity will collapse into $O(n^2)$. □

## 5. Optimization problems on power graphs.

Here we present a number of our results concerning the square, and more generally, the powers of graphs. In §5.1, we give a linear time algorithm for finding a Hamiltonian cycle in cubic graphs. In §5.2, we prove the NP-completeness of finding the maximum cliques in powers of graphs by transformations from the general maximum cliques problem. In §5.6, we prove the chordality of powers of trees by showing that they have the intersection model of subtrees of a tree.

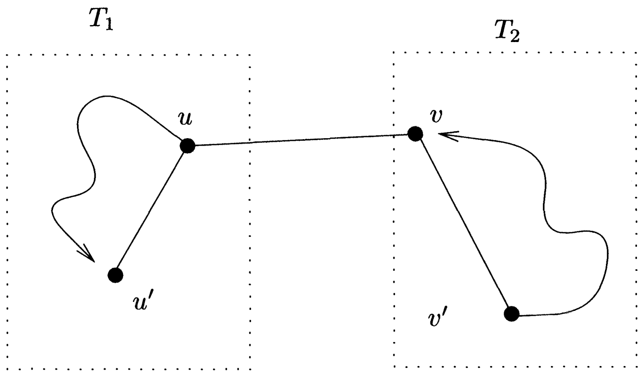


FIG. 6. *Finding a Hamiltonian cycle in a cubic tree.*

**5.1. Hamiltonian cycles in powers of graphs.** Fleischner [8] proved that the square of a biconnected graph is always Hamiltonian. Harary and Schwenk [17] proved that the square of a tree $T$ is Hamiltonian if and only if $T$ does not contain $S(K_{1,3})$ as its induced subgraph. Here $S(K_{1,3})$ denotes the subdivision graph of the complete bipartite graph $K_{1,3}$ or the size four star.

A graph $G$ is *Hamiltonian connected* if every two distinct vertices are connected by a Hamiltonian path. Sekanina [24] proved $G^3$ is Hamiltonian connected if $G$ is connected. It implies that if the size of $G$ is greater or equal to three, then $G^3$ is Hamiltonian. However, we have found no analysis of the complexity of finding Hamiltonian cycles in cubic graphs in the literature. Here we present a linear time algorithm (in terms of the vertices and edges number of $G$) for finding a Hamiltonian cycle in $G^3$ if $G$ is given.

THEOREM 5.1. *Given a connected graph $G = (V, E)$ with size at least three and an integer $k \geq 3$, we can find a Hamiltonian cycle in $G^k$ within $O(|V| + |E|)$ time.*

*Proof.* We will prove this theorem by giving a linear time algorithm for finding a Hamiltonian cycle of $G^k$. First, we will find a spanning tree, $T$, of $G$ while linear time suffices by using a simple depth-first-search algorithm, assuming the given graph is represented in form of adjacency lists. Our goal is to find a Hamiltonian path $h$ in $T^3$; it follows that $uv + h$ will be a Hamiltonian cycle of $T^3$ and thus a Hamiltonian cycle of $G^3$. Since $E(G^i) \subset E(G^{i+1})$ for any positive integer $i$, $uv + h$ is also a Hamiltonian cycle of $G^k$.

We start by selecting an arbitrary edge $uv$ of $T$. After deleting $uv$ from $T$, we separate $T$ into two smaller trees, $T_1$ and $T_2$, containing $u$ and $v$ respectively, as shown in Fig. 6.

If $u$ is also adjacent to some vertex $u'$ in $T_1$, we proceed to find a Hamiltonian path $P_1$ from $u$ to $u'$ in $T_1$, by recursively applying the algorithm; otherwise, $u$ itself is the Hamiltonian path we want. The same procedure will be applied in $T_2$, only this time we construct a Hamiltonian path $P_2$ from $v'$ (a vertex adjacent to $v$ in $T_2$) to $v$. Since the last vertex of $P_1$ ($u'$ or $u$) has distance at most three from the first vertex of $P_2$ ($v'$ or $v$), it follows that $P_1$ concatenating with $P_2$ is a Hamiltonian path from $u$ to $v$ in $T^3$.

To demonstrate that this algorithm can be done in linear time, we must show that the deletion of $uv$ from $T$ can be implemented in constant time. Note that the conventional adjacent lists representation might take $O(n)$ to delete an arbitrary edge, thus resulting an $O(n^2)$ algorithm. Therefore, we use a double linked list $N(v)$, forming a ring, to represent the set of all neighbors of each vertex $v$ in $T$. Each element of $N(v)$ has a pointer that points to an edge $e = uv$, for some $u$. For each vertex $v$ of $T$, we use a cell containing a pointer that pointing to the double-linked list $N(v)$. For each edge $e = uv$, we use four pointers pointing to $u, v, N(u)$, and $N(v)$. Thus, to delete an edge $e = uv$, we start from $e$, following its link to locate the position of $e$ in $N(v)$, delete it in constant time, then locate $v$, and link it to any element of this newly modified $N(v)$. The part of $u$ and $N(u)$ can be handled symmetrically.

Let $|T| = |V| = n, |T_1| = n_1$, and $|T_2| = n_2$, and let time$(T)$ define the time needed to perform our algorithm. Since $n = n_1 + n_2$ and

$$\text{time}(T) = \text{time}(T_1) + \text{time}(T_2) + \text{constant time for deleting } uv,$$

by induction, time$(T) = \text{time}(T_1) + \text{time}(T_2) + c = O(n_1) + O(n_2) + c = O(n)$. So we now conclude that finding a Hamiltonian cycle in a cubic (or higher powers) graph can be done in linear time. □

**5.2. Maximum cliques of power graphs.** Recall that a clique in a graph $G$ is *maximum* if it is the largest induced complete subgraph of $G$. Here we prove that finding maximum cliques in powered graphs is NP-hard by transformation from the general problem of finding the maximum cliques in arbitrary graphs.

THEOREM 5.2. *Let $G = (V, E)$ be a graph. Then, for any fixed integer $k \geq 1$, finding the maximum clique of $G^k$ is NP-complete.*

*Proof.* Note that, for the case of $k = 1$, this problem is to find the maximum clique in $G$, which is NP-complete as shown in [10]. Let $|E(G)| = m$. Now we will prove the theorem for the case that $k > 1$. We divide the problem depending upon whether $k$ is even or odd.

*Case* 1. $k = 2p + 2$, for some $p \geq 0$. Construct a graph $G'$ according to the following transformation. For each edge $uv \in E$, construct a set of $2p + 1$ new vertices, $V_{uv} = \{[uv]_0, [uv]_1, \ldots, [uv]_{2p}\}$. Let $W = \cup_{uv \in E} V_{uv}$, the new vertices in $G'$. Link these vertices of $V_{uv}$ with $u$ and $v$ to form a path by adding the edges $E_{uv} = \{([uv]_i, [uv]_{i+1}) : 0 \leq i < 2p\} \cup \{(u, [uv]_0), ([uv]_{2p}, v)\}$. Note that $[uv]_p$ is the unique center vertex in $E_{uv}$. Link these center vertices in $G'$ by defining $C = \{([uv]_p, [u'v']_p) :$ for all $uv, u'v' \in E\}$. Figure 7 illustrates this transformation.

Now construct the graph $G' = (V', E')$ by defining

$$V' = V \cup W,$$
$$E' = \cup_{uv \in E} E_{uv} \cup C.$$

Since, for each pair of vertices $x \in V_{uv}, y \in V_{u'v'}$, the path from $x$ to $[uv]_p$, to $[u'v']_p$, and, finally, to $y$ has length at most $2p + 1 = k - 1$. It follows that for each pair $v \in V$ and $w \in W$, $d_{G'}(v, w) \leq k$, implying that $W \cup v$ forms a clique in $G'^k$ for any $v \in V$. Also notice that, for each two vertices, $u$ and $v$, of $V$, $uv \in E$ if and only if $d_{G'}(u, v) = k$. Thus, a subset of $V$, $Q$ of size $q$, is a clique of $G$ if and only if $X \cup W$ is a clique of $(G')^k$. That is, $G$ has a clique of size $q$ if and only if $G'^k$ has a clique of size $(k - 1)m + q$. Since clearly $G'$ can be constructed in polynomial time, finding the maximum clique in $G^k$ is NP-hard for the case of even $k$.

*Case* 2. $k = 2p+1$, for some $p \geq 1$. The construct of the graph $G'$ is similar to the even case except that we now add an additional kernel vertex $c$ to $G'$. For each edge $uv \in E$, construct a set of $2p$ new vertices, $V_{uv} = \{[uv]_1, [uv]_1, \ldots, [uv]_{2p}\}$. The new

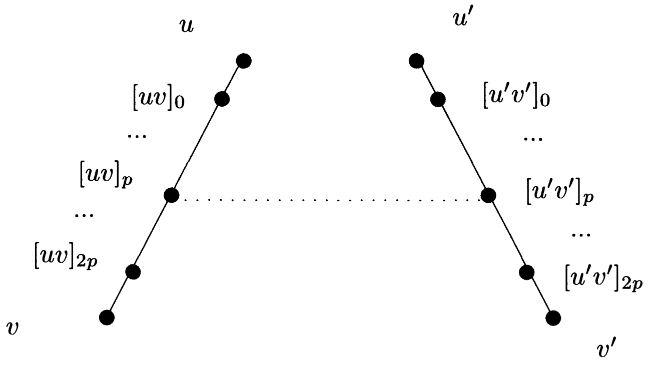


FIG. 7. *The transformation of even powers of a graph.*

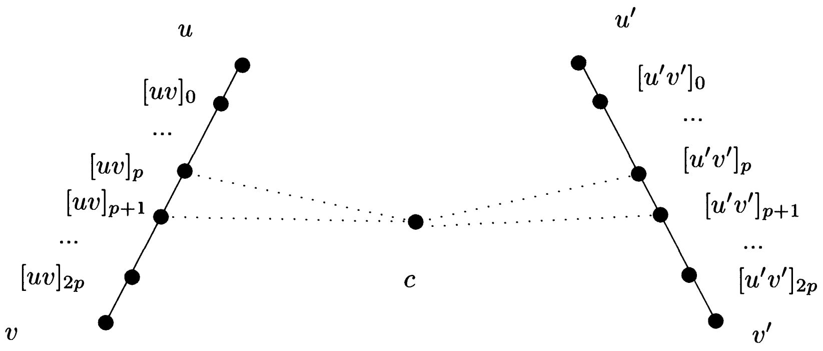


FIG. 8. *The transformation of odd powers of a graph.*

vertices in $G'$ will be $W = \cup_{uv \in E} V_{uv} \cup \{c\}$. Link vertices of $V_{uv}$ with $u$ and $v$ to form a path by adding the edges $E_{uv} = \{([uv]_i, [uv]_{i+1}) : 1 \le i < 2p\} \cup \{(u, [uv]_0), ([uv]_{2p}, v)\}$. Note that $[uv]_p$ and $[uv]_{p+1}$ are two center vertices in $E_{uv}$. Link these center vertices with the kernel $c$ by defining $C = \{(c, [uv]_i) : \text{for all } uv \in E, i \in [p, p+1]\}$. Figure 8 illustrates this transformation.

Again, construct the graph $G' = (V', E')$ by defining

$$V' = V \cup W,$$
$$E' = \cup_{uv \in E} E_{uv} \cup C.$$

Since for each pair of vertices $x \in V_{uv}, y \in V_{u'v'}$ the path from $x$ to $y$ defined by concatenating the shortest path from $x$ to $c$ and from $c$ to $y$ has a length at most $2p = k - 1$, it follows that, for each pair $v \in V$ and $w \in W$, $d_{G'}(v, w) \le k$, implying that $W \cup v$ forms a clique in $G'^k$ for any $v \in V$. Also notice that, for each two vertices, $u$ and $v$, of $V$, $uv \in E$ if and only if $d_{G'}(u, v) = k$. Thus, given $X$, a subset of $V$, $X$ is a clique of $G$ if and only if $X \cup W$ is a clique of $(G')^k$. That is, $G$ has a clique of size $q$ if and only if $G'^k$ has a clique of size $(k-1)m + 1 + q$. It is not difficult to verify that $G'$ can be constructed in polynomial time; thus, the problem is NP-hard for the case of odd $k$.

Since the problem is clearly in NP, we now conclude that the problem of finding the maximum clique of $G^k$ is NP-complete for any fixed integer $k \geq 1$. □

**5.3. Maximum independent sets of power graphs.** Given a graph $G$ and a positive integer $k > 1$, the transformed graph $G'$ described in the previous proof is uniquely defined. We can denote this transformation by $G' = \Phi(G, k)$. The most interesting property of this transformation is that, for every two distinct vertices $u, v$ of $G$, we have

$$uv \notin G \iff d_{\Phi(G,k)}(u,v) = k+1 \tag{1}$$

and

$$uv \in G \iff d_{\Phi(G,k)}(u,v) = k. \tag{2}$$

Further, for every two new vertices $x, y \in W$ of $\Phi(G,k)$ and each original vertex $v$ of $G$, we have

$$d_{\Phi(G,k)}(v,x) \leq k \quad \text{and} \quad d_{\Phi(G,k)}(x,y) < k \tag{3}$$

Thus, $\Phi(G,k)^k$, the $k$th power of the transformed graph $\Phi(G,k)$, still contains the information of the original clique. That is, assuming $Q \subset V(G)$, $Q$ is a clique in $G$ if and only if $Q$ is a clique in $\Phi(G,k)^k$. This observation essentially constitutes the proof of Theorem 5.2. Interesting enough, we will find that $\Phi(G,k)^k$ also contains information of the original *independent set.* Formally, we have the following theorem.

THEOREM 5.3. *Let $G = (V, E)$ be a graph. Then, for any fixed integer $k \geq 1$, finding the maximum independent set of $G^k$ is NP-complete.*

*Proof.* For the case of $k = 1$, this problem is to find the maximum independent set in $G$, which is NP-complete as shown in [10]. Otherwise, we can reduce the problem of finding the maximum independent set in arbitrary graphs to this problem for any fixed $k > 1$.

Given an arbitrary graph $G = (V, E)$, we can construct the graph $\Phi(G,k)$ in polynomial time as shown in the proof of Theorem 5.2. We will now show that $G$ has an independent set of size $\alpha$ if and only if $\Phi(G,k)^k$ has an independent set of size $\alpha$. Let $I$, $|I| = \alpha$ be an independent set of $G$. By equation (1), $I$ is clearly an independent set of $\Phi(G,k)^k$.

On the other direction, let $I$, $|I| = \alpha$, be an independent set of $\Phi(G,k)^k$. Without loss of generality, we can assume $\alpha > 1$. By equation (3), $I$ can not contain any new vertex ($W$); thus $I \subset V$. Again, by equation (1), $I$ is an independent set of $G$. □

**5.4. Coloring and cliques partition of power graphs.** Given a graph $G$, a *clique partition* of $G$ is a partition $\{V_1, \ldots, V_k\}$ of the vertices of $G$ such that each $V_i$, $1 \leq i \leq k$, is a clique in $G$. The *minimum* clique partition of $G$ is a clique partition with the smallest possible $k$. A *vertex coloring* of $G$ is a partition $\{V_1, \ldots, V_k\}$ of the vertices of $G$ such that each $V_i$, $1 \leq i \leq k$, is an independent set in $G$. The *minimum* vertex coloring of $G$ is a coloring with the smallest possible $k$. Once again, the transformation $\Phi$ can be used to prove the the hardness of finding a minimum clique partition or finding a minimum coloring of in a power graph.

THEOREM 5.4. *Let $G = (V, E)$ be a graph. Then, for any fixed integer $k \geq 1$, finding the minimum clique partition of $G^k$ is NP-complete.*

*Proof.* For the case of $k = 1$, this problem is to find the minimum clique partition in $G$, which is NP-complete as shown in [10]. Otherwise, we can reduce the problem of finding the minimum clique partition in general graph to this problem for any fixed $k > 1$.

Given an arbitrary graph $G = (V, E)$, we can construct the graph $\Phi(G, k)$ in polynomial time as shown in the proof of Theorem 5.2. We will now show that $G$ has a clique partition of size $\leq \beta$ if and only if $\Phi(G, k)^k$ has a clique partition of size $\leq \beta$. Let $P = \{V_1, \dots, V_\alpha\}$, $\alpha \leq \beta$, be a clique partition of $G$. By equation (2), for each $V_i \in P$, $1 \leq i \leq \alpha$, $V_i$ is again a clique in $\Phi(G, k)^k$. Further, by equation (3), $V_1 \cup W$ is still a clique. Thus $\Phi(G, k)^k$ has a clique partition of size $\alpha \leq \beta$.

On the other direction, let $P = \{V_1, \dots, V_\alpha\}$, $\alpha \leq \beta$, be a clique partition of $\Phi(G, k)^k$. For each vertices subset $V_i$, $1 \leq i \leq \alpha$, let $V_i' = V_i \cap V$. For each $i$, clearly $V_i'$ (if not empty) is still a clique in $\Phi(G, k)^k$. Further, by equation (2), $V_i'$ is also a clique in $G$. It follows that $\{V_1', \dots, V_\alpha'\}$ is a clique partition of $G$. That is, $G$ has a clique partition of size $\leq \beta$. □

THEOREM 5.5. *Let $G = (V, E)$ be a graph. Then, for any fixed integer $k \geq 1$, finding the minimum vertex coloring of $G^k$ is NP-complete.*

*Proof.* For the case of $k = 1$, this problem is to find the minimum vertex coloring in $G$, which is NP-complete as shown in [10]. Otherwise, we can reduce the problem of finding the minimum vertex coloring in general graph to this problem for any fixed $k > 1$.

Given an arbitrary graph $G = (V, E)$, we can construct the graph $\Phi(G, k)$ in polynomial time as shown in the proof of Theorem 5.2. Let $w = |W|$ denote the size of the newly added vertices of $\Phi(G, k)$. We will now show that $G$ has a vertex coloring of size $\alpha$ if and only if $\Phi(G, k)^k$ has a vertex coloring of size $\alpha + w$. Let $C = \{V_1, \dots, V_\alpha\}$ be a vertex coloring of $G$. By equation (1), each $V_i \in P$, $1 \leq i \leq \alpha$, is again an independent set in $\Phi(G, k)^k$. For each $v \in W$, singleton $\{v\}$ certainly constitutes an independent set. Thus $\{V_1, \dots, V_\alpha, \{v\}_{v \in W}\}$ is a coloring of size $\alpha + w$.

In the other direction, let $C = \{V_1, \dots, V_{\alpha+w}\}$ be a vertex coloring of $\Phi(G, k)^k$. By equation (3), each new vertex $v \in W$ is adjacent to every other vertex in $\Phi(G, k)^k$. Thus each vertex $v \in W$ must form a singleton in the coloring $C$. Deleting those singletons from $C$, we have $C' = C \setminus \{\{v\} : v \in W\}$. Note that $|C'| = \alpha$, and $C'$ is a vertex coloring of $G$ since each vertex set of $C'$ is still an independent set in $G$ by equation (1). That is, $G$ has a vertex coloring of size $\alpha$. □

**5.5. Minimum dominating sets of power graphs.** Given a graph $G = (V, E)$, a *dominating set*, $D$, is a subset of vertices of $G$ such that for each vertex $v \in V \setminus D$, there is some vertex $u \in D$ such that $uv \in E$. In other words, $\bigcup_{v \in D} N^+(v) = V$. A *minimum dominating set* is a dominating set with the smallest cardinality. It would be nice if the same transformation can also be used to prove that the minimum dominating set problem in $G^k$ is also NP-complete. Unfortunately, since the size of the minimum dominating set in $\Phi(G, k)^k$ is always 1, we must use another technique.

THEOREM 5.6. *Let $G = (V, E)$ be a graph. Then, for any fixed integer $k \geq 1$, finding the minimum dominating set of $G^k$ is NP-complete.*

*Proof.* For the case of $k = 1$, this problem is to find the minimum dominating set in $G$, which is NP-complete as shown in [10]. Otherwise, we can reduce the problem

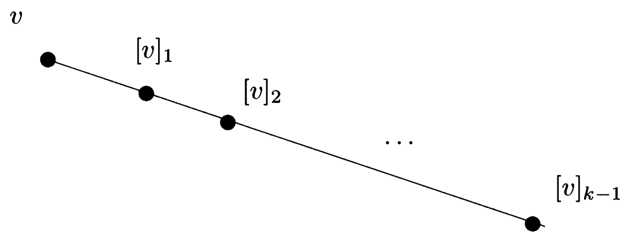


FIG. 9. *Adding a length $k-1$ tail to a vertex $v$.*

of finding the minimum dominating set in arbitrary graphs to this problem for any fixed $k > 1$.

Given $k$ and an arbitrary graph $G = (V, E)$, we will construct the following graph, denoted $\Psi(G,k)$, in polynomial time. For each vertex $v$ of $G$, we will attach a *tail* of length $k-1$ to it. That is, for each $v \in V$, let

$$V_v = \{[v]_1, \dots, [v]_{k-1}\},$$
$$E_v = \{v[v]_1\} \cup \{[v]_i[v]_{i+1} : i \in [1..k-2]\}.$$

Figure 9 illustrates this transformation. Now we can define $\Psi(G,k) = (V', E')$ as follows:

$$V' = \cup_{v \in V} V_v \cup V,$$
$$E' = \cup_{v \in V} E_v \cup E.$$

Note that $\Psi(G,k)$ has $kn$ vertices and $m + (k-1)n$ edges. $\Psi(G,k)$ can certainly be constructed in polynomial time. Note that graphs $G$ and $\Psi(G,k)$ are related by the following equation. For $v, x \in V, i \in [1..k-1]$,

$$(4) \qquad d_{\Psi(G,k)}(v, [x]_i) = d_G(v,x) + i.$$

We will now show that $G$ has a dominating set of size $\alpha$ if and only if $\Psi(G,k)^k$ has a dominating set of size $\alpha$. Let $D = \{v_1, \dots, v_\alpha\}$ be a dominating set of $G$. That is, for each $x \in V' \setminus D$, there exists a $v \in D$ such that $d_G(v,x) = 1$. By equation (4), for each $[x]_i \in V \setminus D$, $d_G(v, [x]_i) \le k$ since $i \in [1..k-1]$. That is, $D$ is still a dominating set of $\Psi(G,k)^k$.

In the other direction, let $D = \{v_1, \dots, v_\alpha\}$ be a dominating set of $\Psi(G,k)^k$. Without loss of generality, we can assume that $D \subset V$. Otherwise, for each $[v]_j \in D$, replace $[v]_j$ by the vertex $v$. Call the modified set $D'$. By equation (4), $D'$ is still a dominating set of $\Psi(G,k)^k$. We claim that $D'$ is also a dominating set of $G$. Suppose $D'$ is not. Then there must be a vertex $v$ in $G$ such that for each $x \in D'$, $d_G(v,x) > 1$. By equation (4), that means for each $x \in D'$, $d_{\Psi(G,k)}(v,x) > k$, but it is contradicted by the fact that $D'$ is a dominating set of $\Psi(G,k)^k$. Since $D'$ is a dominating set of $G$ and $D'$ has the same cardinality of $D$, it follows that $G$ has a dominating set of size $\alpha$. □

**5.6. The chordality of powers of trees.** A graph in which every simple cycle of length strictly greater than three possesses a chord is called a *chordal graph.* In the

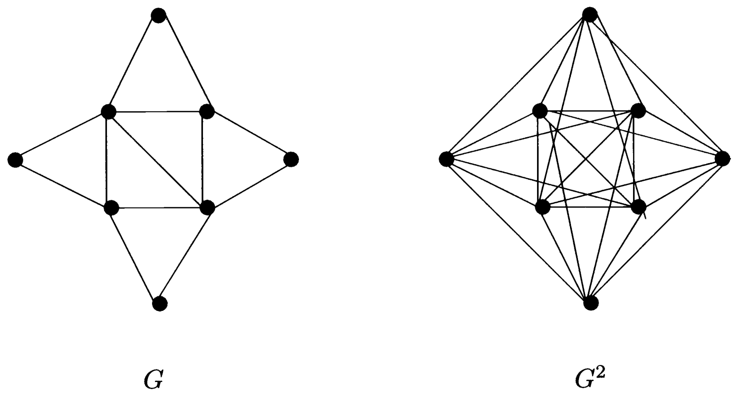


FIG. 10. *A chordal graph and its nonchordal square.*

literature, chordal graphs have also been called *triangulated, rigid-circuit, monotone transitive,* and *perfect elimination* graphs [6], [19], [9], [22]. Rose, Tarjan, and Lueker [22] presented an algorithm for recognition of a chordal graph in linear time. Along with their algorithm, the maximum clique of a chordal graph can also be found in linear time. Walter [28], Gavril [11], and Buneman [4] proved that a graph $G$ is chordal if and only if $G$ is the intersection graph of a family of subtrees of a tree. Observe that the square of a chordal graph is not necessarily chordal, as shown in Fig. 10. However, we will show that any arbitrary powers of trees are always chordal.

THEOREM 5.7. *Powers of trees are chordal.*

*Proof.* Let $T$ be a tree. We want to prove that $T^k$ is chordal for any positive integer $k$. Let $S(G)$ be the subdivision graph of a graph $G$ as defined in §4 and $N^k_G(v)$ denote the set $\{u \in G : d(u,v) \leq k\}$ where $d(u,v)$ is the distance between vertices $u$ and $v$ in $G$. Note that $uv \in G^k$ if and only if $N^k_{S(G)}(u) \cap N^k_{S(G)}(v) \neq \emptyset$, implying that $G^k$ is the intersection graph of the family $S = \{N^k_{S(G)}(v) :$ for all $v \in G\}$. It follows that $T^k$ is the intersection graph of the family $\{N^k_{S(T)}(v) :$ for all $v \in T\}$. Since $S(T)$ is a tree and $\langle N^k_{S(T)}(v)\rangle_{S(T)}$ is a subtree of $S(T)$ with center vertex $v$ and radius $k$, $T^k$ is the intersection graph of a family of subtrees of the tree $S(T)$. Thus $T^k$ is chordal. □

## 6. Conclusions.

We have presented efficient algorithms for finding the square roots of graphs in three interesting special cases: tree squares, planar graphs, and subdivided graphs. Further, we have studied the complexity of several optimization problems on powers of graphs. Two interesting open problems remain.

- Let $A$ be the adjacency matrix of a graph, made reflexive by adding a self-loop to each vertex. Given $A^2$, can we determine one of its $(0,1)$-matrix square roots in polynomial time? This problem is potentially easier than finding the square root of a graph, since we are also given the number of paths of length at most two between each pair of vertices.
- What is the complexity of recognizing the squares of directed-acyclic graphs (DAGs)? Clearly, the square of a DAG is a DAG. All square roots of a DAG $G^2$ contain the transitive reduction of $G^2$ as a subgraph. The time complexity of finding both the transitive closure and reduction of a digraph is equivalent to boolean matrix multiplication [1].

**Acknowledgments.** We thank Gene Stark for providing a simpler proof of Theorem 5.7 and Alan Tucker for valuable discussions.

## REFERENCES

[1] A. Aho, M. Garey, and J. Ullman, *The transitive reduction of a directed graph*, SIAM J. Comput., 1 (1972), pp. 131–137.

[2] M. Behzad, *A criterion for the planarity of a total graph*, Proc. Cambridge Philos. Soc., 63 (1967), pp. 679–681.

[3] M. Behzad and H. Radjavi, *The total group of a graph*, Proc. Amer. Math. Soc., 19 (1968), pp. 158–163.

[4] P. Buneman, *A characterization of rigid circuit graphs*, Discrete Math., 9 (1974), pp. 205–212.

[5] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, in Proc. 19th ACM Symposium Theory of Computing, 1987, pp. 1–6.

[6] G. A. Dirac, *On rigid circuit graphs*, Abh. Math. Sem. Univ. Hamburg, 25 (1961), pp. 71–76.

[7] F. Escalante, L. Montejano, and T. Rojano, *Characterization of n-path graphs and of graphs having nth root*, J. Combin. Ser. Theory B, 16 (1974), pp. 282–289.

[8] H. Fleischner, *The square of every two-connected graph is Hamiltonian*, J. Combin. Theory Ser. B, 16 (1974), pp. 29–34.

[9] D. R. Fulkerson and O. A. Gross, *Incidence matrices and interval graphs*, Pacific J. Math., 15 (1965), pp. 835–855.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.

[11] F. Gavril, *The intersection graphs of subtrees in trees are exactly the chordal graphs*, J. Combin. Theory Ser. B, 16 (1974), pp. 47–56.

[12] ———, *A recognition algorithms for the total graph*, Networks, 8 (1978), pp. 121–133.

[13] D. P. Geller, *The square root of a digraph*, J. Combin. Theory, 5 (1968), pp. 320–321.

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

[15] F. Harary, *Graph Theory*, Addison-Wesley, Massachusetts, 1972.

[16] F. Harary, R. M. Karp, and W. T. Tutte, *A criterion for planarity of the square of a graph*, J. Combin. Theory, 2 (1967), pp. 395–405.

[17] F. Harary and A. Schwenk, *Trees with hamiltonian square*, Mathematika, 18 (1971), pp. 138–140.

[18] J. E. Hopcroft and R. E. Tarjan, *Dividing a graph into triconnected components*, SIAM J. Comput., 2 (1973), pp. 135–158.

[19] C. G. Lekkerkerker and J. Ch. Boland, *Representation of a finite graph by a set of intervals on the real line*, Fund. Math., 51 (1962), pp. 45–64.

[20] R. Motwani and M. Sudan, *Square is NP-complete*, manuscript, September 1991.

[21] A. Mukhopadhyay, *The square root of a graph*, J. Combin. Theory, 2 (1967), pp. 290–295.

[22] D. J. Rose, R. E. Tarjan, and G. S. Lueker, *Algorithmic aspects of vertex elimination of graphs*, SIAM J. Comput., 5 (1976), pp. 266–283.

[23] I. C. Ross and F. Harary, *The square of a tree*, Bell System Tech. J., 39 (1960), pp. 641–647.

[24] M. Sekanina, *On an ordering of the set of vertices of a connected graph*, Tech. Report No. 412, Publ. Fac. Sci. Univ. Brno, 1960.

[25] R. E. Tarjan, *Depth-first search and linear graph algorithms*, SIAM J. Comput., 1 (1972), pp. 146–159.

[26] W. T. Tutte, *Connectivity in Graphs*, University of Toronto Press, London, 1966.

[27] P. Underground, *On graphs with Hamiltonian squares*, Discrete. Math., 21 (1978), p. 323.

[28] J. R. Walter, *Representations of Rigid Cycle Graphs*, Ph.D. thesis, Wayne State University, Detroit, MI, 1972.

SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 119–132, February 1995

# SEARCH PROBLEMS IN THE DECISION TREE MODEL*

LÁSZLÓ LOVÁSZ†, MONI NAOR‡, ILAN NEWMAN§, AND AVI WIGDERSON¶

**Abstract.** The relative power of determinism, randomness, and nondeterminism for search problems in the Boolean decision tree model is studied. It is shown that the gaps between the nondeterministic, the randomized, and the deterministic complexities can be arbitrarily large for search problems. An interesting connection of this model to the complexity of resolution proofs is also mentioned.

**Key words.** Boolean functions, decision trees

**AMS subject classifications.** 68R05, 94A15, 68Q10

## 1. Introduction.

Ramsey's theorem asserts that every graph on $n$ vertices has either a complete graph or an independent set of size $\frac{1}{2}\log n$. A natural search problem associated with this theorem is to find such a subgraph. Many other problems, like the ones below, have a similar flavor. Given an assignment of $n$ pigeons into $n-1$ pigeon holes, find two pigeons assigned to the same hole. Given a $k$-chromatic graph and a coloring of its nodes with fewer than $k$ colors, find two neighbors that have the same color. Given an unsatisfiable 3-CNF formula and an assignment to its variables, find a clause that is not satisfied.

How hard is it to solve such search problems? The answer depends of course on their representation and the computational model. We assume that the input is encoded in binary, and that we are only allowed to probe input bits. This gives the familiar Boolean decision tree model, adapted to solving search problems rather than computing Boolean functions. We study the relationship between the standard nondeterministic, probabilistic, and deterministic variants of this model and discover that it is drastically different from the case of function computation, where all three measures are polynomially related (see [2], [7], [19], [14]). In all the examples just listed it is easy to guess and verify the solution; hence the nondeterministic decision tree complexity is small (a constant or polylog). If the decision tree was computing a function, this would imply that both the randomized and deterministic complexities are small, because the deterministic complexity (and thus the randomized too) is at most the square of the nondeterministic complexity [2], [7], [19]. It turns out that for search problems these gaps can be arbitrarily large.

Our investigation is partly motivated by a similar study of search problems in the communication complexity setting in the work of Karchmer and Wigderson [11] and Raz and Wigderson [18], where a similar phenomenon occurs but not to the same extent. Another study of search problems was carried out by Papadimitriou [15]

* Received by the editors July 9, 1992; accepted for publication (in revised form) December 6, 1993.

† Department of Computer Science, Princeton University, Princeton, New Jersey 08544, and Eötvos Loránd University, Budapest, Hungary.

‡ Department of Applied Mathematics, Weizmann Institute, Rehovot 76100, Israel. Part of this work was completed while the author was visiting the IBM Almaden Research Center, San Jose, California 95120.

§ Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, New Brunswick, New Jersey, 08903, and Department of Computer Science, Hebrew University, Jerusalem 91904, Israel.

¶ Department of Computer Science, Princeton University, Princeton, New Jersey 08544, and Department of Computer Science, Hebrew University, Jerusalem 91904, Israel.

where complexity classes defined by search problems were investigated. Some of our examples are inspired by this work and [8], [10].

The examples above may remind the readers of resolution proofs. Indeed, resolution proofs viewed top down yield Boolean decision trees to the search problems above (this fact seems to be folklore and is elaborated in the Appendix).

Thus, the exponential length lower bounds on resolution proofs (see, e.g., [4]) provide linear deterministic lower bounds even when the nondeterministic complexity is a constant. However, the distinction between the resolution and the decision tree points of view becomes clear when trying to make sense out of the probabilistic model. In resolution, one proves simultaneously that every assignment falsifies some clause in an unsatisfiable formula. This has no natural probabilistic analogue. The decision tree approach, where one must find for a given input assignment a clause that it falsifies, has a natural randomized version. Indeed, we will be primarily concerned with the power of probabilistic computation for such problems.

We prove that the probabilistic complexity can be at both ends of the spectrum. We give an explicit search problem for which the probabilistic (and nondeterministic) complexity is constant, but the deterministic is linear. On the other hand we provide two explicit problems for which there is a large gap between the nondeterministic and probabilistic complexities: one in which the first is constant and the second is $\Omega(n^{\frac{1}{6}})$, and another in which the first is $O(\log n)$ while the second is nearly linear. Finally, we present an explicit problem for which there is a simultaneous exponential gap between the nondeterministic versus the randomized and the randomized versus the deterministic complexities. This last example uses an upper bound due to Irani [9] on online coloring algorithms to provide a lower bound on the deterministic complexity. We note here that as far as we know, no such simultaneous gaps are known for any other model of computation.

The special case of nondeterministic complexity 2 deserves a special interest. It corresponds to unsatisfiable 2-CNF formulae. We characterize the deterministic complexity by the structure of the formula and note that it can never exceed $1 + \log n$. We show that here too the gap between the randomized complexity and the nondeterministic and the gap between the randomized complexity and the deterministic can be arbitrarily large.

The paper is organized as follows. In §2 we give the formal definitions of search problems and decision trees and show that the CNF search problem is "complete" for all the variants of decision trees. In §3 we construct search problems with a large gap between the deterministic and the randomized complexity, randomized and nondeterministic complexity, and simultaneous nondeterministic—randomized—deterministic complexity gaps. In §4 we discuss the special case of nondeterministic complexity 2. The exact relationship to the resolution problem appears in the Appendix.

**2. Definitions.** A search problem is specified by $n$ variables and a collection of "witnesses." In addition, this collection must have the property that every assignment to the $n$ variables is associated with at least one witness. Given an input string which is an assignment to the $n$ variables, the goal is to find a witness consistent with it. We state the definition formally.

DEFINITION 2.1. *A search problem on n variables is a relation $F \subseteq \{0,1\}^n \times W$, such that $\forall x \in \{0,1\}^n$, $\exists w \in W$ for which $(x, w) \in F$. $W$ is a finite set, called the set of witnesses. The goal of the search for input $x \in \{0,1\}^n$ is to find a $w \in W$ such that $(x, w) \in F$ (we call such a $w$ a valid witness for $x$).*

A special class of search problems comes from DNF tautologies. A monomial $m$

is a conjunction of literals (over the variable set $X$). A DNF tautology $g$ with a set of monomials $M$ is a Boolean formula $g = \vee_{m \in M} m$, and so $g$ is true for every assignment to the variables. $g$ naturally defines a search problem on $\{0,1\}^X \times M$. Each input (a truth assignment to the variables) is associated with all monomials $m \in M$ that are satisfied by it.

The class of search problems defined by DNF tautologies is quite general. Indeed as we shall see, every search problem $F$ gives rise to a search problem on an associated DNF tautology, $g(F)$, that will turn out to be equivalent to $F$.

With every monomial $m$ we associate the subcube $C_m$ of all the points in $\{0,1\}^X$ that are consistent with $m$ (i.e., evaluate to "1" on $m$).

DEFINITION 2.2. *Let $F \subseteq \{0,1\}^n \times W$ be a search problem. The set of monomials associated with $F$, denoted by $M_F$, is defined by $m \in M_F$ if $\exists w \in W$ such that $\forall x \in C_m$, $(x,w) \in F$. In words, $m \in M_F$ if all inputs $x$ that are consistent with $m$ share a mutual witness $w$.*

CLAIM 2.3. *Let $F \subseteq \{0,1\}^n \times W$ be a search problem and let $g(F)$ be the formula defined by $g(F) = \vee_{m \in M_F} m$, then $g(F)$ is a* DNF *tautology.*

*Proof.* Every input $x$ has a witness in $W$. In particular, the subcube that contains the single point $x$ defines a monomial $m \in M_F$ that is satisfied by $x$. □

Thus $g(F)$ defines a valid DNF search problem $F' \subseteq \{0,1\}^n \times M_F$, where $F'$ is the set of pairs $(x,m)$ such that $x$ satisfies $m$. It will be convenient though, for historical reasons, to consider $f(F) = \overline{g(F)}$ which is an unsatisfiable CNF formula with the associated search problem. For each input (an assignment to the variables) find an unsatisfied clause. Observe that there is a natural correspondence between witnesses of $F$ and $F'$.

**2.1. Decision tree complexity for search problems.** Let $F \subseteq \{0,1\}^n \times W$ be a search problem. A deterministic decision tree for $F$ is an algorithm that may query the value of an input bit at each step. The goal is to find a consistent witness. Formally, such an algorithm is a rooted binary tree in which every internal node is labeled by a variable and the two outgoing edges are labeled by the two possible values to that variable. Each leaf is labeled with a witness $w \in W$. Every assignment of the variables determines a path from the root to a leaf in a natural way. The tree is a valid decision tree, if for every assignment this path ends in a leaf labeled by a valid witness.

The *deterministic* complexity of $F$, $D(F)$, is the minimum depth of any decision tree for $F$.

The *nondeterministic* complexity of $F$, $N(F)$, is the minimum number of variables that must be probed in order to find a valid witness for the worst case input. Alternatively, it is exactly the maximum size, over all inputs $x$, of the smallest monomial in $M_F$ that is satisfied by $x$.

A randomized decision tree for $F$ is a distribution over deterministic decision trees for $F$. The complexity of a randomized decision tree is the expected path length for the worst case input. The *randomized* complexity of $F$, $R(F)$, is the minimum over all randomized decision trees for $F$.

*Facts.* 1. Let $F \subseteq \{0,1\}^n \times W$ be a search problem and $F' \subseteq \{0,1\}^n \times M_F$ the associated search problem. Then $D(F) = D(F')$, $R(F) = R(F')$ and $N(F) = N(F')$. Furthermore, there is a natural correspondence between any decision tree for $F$ (deterministic, randomized, or nondeterministic) and a (corresponding) decision tree for $F'$. Thus $F$ and $F'$ are indeed restatements of essentially the same problem. Throughout the paper we will not distinguish between $F$ and its associated $F'$.

2. For every decision problem $F$ : $N(F) \leq R(F) \leq D(F)$.

An observation of Chvatal and Szemerédi [5] is that for a search problem $F$, lower bounds on the (regular) resolution process for the unsatisfiable formula $f(F)$ imply lower bounds on the deterministic decision tree complexity for $F$. We elaborate on that point in the Appendix.

## 3. The relative power of determinism and randomization versus nondeterminism.

In this section we present some explicit search problems for which there are large gaps between the different decision tree complexity measures. Our main task is to construct search problems for which $N(F) << D(F)$, $R(F) << D(F)$, or $N(F) << R(F) << D(F)$ simultaneously. Another parameter to consider in each case is $D(F)$ versus the number of variables $n$, which is the obvious upper bound for all the three measures of complexity.

### 3.1. Gaps between $R(F)$ and $D(F)$.

We present here an *explicit* search problem for which $R(F) = O(1)$ and $D(F) = \Omega(n)$. Note that the existence of such a problem follows from [4] by probabilistic arguments.

Let $G(U, V, E)$ be a bipartite graph. Define the search problem DEG($G$) on $|E|$ variables in the following way. Each 0-1 assignment to the variables is interpreted as a subgraph $G'$ of $G$, defined by those edges that are assigned "1." The search problem is to find a vertex $r$ whose degree in $G'$ is not one. Clearly if the sides of the graph are not equal ($|V| \neq |U|$) such a vertex exists for every subgraph $G'$; thus as long as the sides are not equal, DEG($G$) is a valid search problem.

LEMMA 3.1. *Let $G = (U, V, E)$ be a bipartite graph with maximum degree $d$ and $|U| = 2n$, $|V| = n$, then $N(\mathrm{DEG}(G)) \leq d$, and $R(\mathrm{DEG}(G)) \leq 2d(d+1)$.*

*Proof.* The nondeterministic complexity $N(\mathrm{DEG}(G)) \leq d$ since for every input (subgraph $G'$) one must only check the incident edges of the guessed vertex.

Consider the following random decision tree. Pick at random a vertex $u \in U$ and independently a vertex $v \in V$ and ask for all edges that are incident to each of the two vertices (i.e., $2d$ edges are being checked). If $u$ or $v$ produces a witness you should stop; otherwise repeat this process until done.

We claim that the probability that a witness is discovered in each iteration is at least $\frac{1}{d+1}$. If there are more than $\frac{2nd}{d+1}$ edges in the subgraph $G'$ defined by the "1"-edges, then at least $\frac{n}{d+1}$ of the vertices in $V$ are of degree at least 2. In this case the fact that $v \in V$ is chosen at random proves the claim. If, on the other hand, $G'$ has less than $\frac{2nd}{d+1}$ edges, then at least $\frac{2n}{d+1}$ of the vertices in $U$ are of degree 0 in $G'$. Thus, the fact that $u \in U$ is chosen at random proves the claim in this case. We conclude that the expected number of iterations is $d+1$; in each of them $2d$ edges are probed which yields the above upper bound. $\square$

We will show now an infinite sequence of bipartite graphs for which the deterministic complexity is $\Omega(n)$.

Let $G = (U, V, E)$ be a bipartite graph with maximum degree $d$, $|U| = 2n$, $|V| = n$ as before, and with the additional "expansion" property: for every $S \subset U$, $|S| \leq n/4 \Rightarrow |N(S)| \geq 2|S|$. ($N(S) = \{v| \ (u, v) \in E \text{ and } u \in S\}$).

Such a graph exists for large enough $d$ and infinitely many $n's$ and can be efficiently constructed using expander graphs [13] ($d$ can be taken to be 30).

THEOREM 3.2. *Let $G$ be a graph as defined above then $R(\mathrm{DEG}(G)) = O(1)$ and $D(\mathrm{DEG}(G)) = \Omega(n)$.*

*Proof.* The fact that $R(\mathrm{DEG}(G)) = 2d(d+1) = O(1)$ was already proved in Lemma 3.1.

We show an adversary strategy that is going to cause any deterministic decision tree to probe $\Omega(n)$ edges. The adversary will be limited to produce a subgraph for which $\forall v \in V$, $deg_{G'}(v) = 1$ and $\forall u \in U$, $deg_{G'}(u) \leq 1$. Thus, the answer the decision tree has to find is a vertex in $U$.

We need some definitions. For any $S \subseteq U$ and subgraph $G'$ of $G$, $N_{G'}(S) = \{v \in V \mid (u,v) \in G', u \in S\}$. For stage $i$ (after $i$ edges were probed) let $E_i' = \{e | e$ was assigned "0" $\}$ and $E_i'' = \{e |\ e$ was not probed and $\exists e'$, $e'$ assigned "1" and $e \cap e' \neq \emptyset\}$. Define $G_i = G - (E_i' \cup E_i'')$. In words, $G_i$ contains all the edges that are still possible for the adversary to use in its final subgraph without violating the above limitation.

For each $S \subseteq U$, define $N_i(S) = N_{G_i}(S)$. For any subgraph $G'$ of $G$, define $S \subset U$ to be unmatchable if $N_{G'}(S) < |S|$. Let $S^*_{G'}$ denote a minimum cardinality unmatchable set in $G'$. Finally call $S_i^* = S^*_{G_i}$ a minimal unmatchable set in step $i$.

By the above limitation on the adversary, at step $i$ the subgraph $G_i$ contains a partial matching from $U$ to $V$. Thus the decision tree cannot know the answer as long as there is no isolated vertex in $G_i$. Obviously such a vertex is, by itself, a minimum unmatchable set. Initially, by the definition of the graph, $|S_0^*| \geq n/4$. The strategy of the adversary is to make sure that the minimum unmatchable set size does not decrease too fast. Formally, in step $i$ an edge $e = (u,v)$, $u \in U$, $v \in V$ is probed. The adversary computes the following.

1. $S^0(e) = S^*(G_i - e)$, i.e., the minimum unmatchable set that occurs on "0" answer on $e$.

2. $S^1(e) = S^*(G_i - \{f = (x,v) |\ f$ was not probed and $x \in U\})$, i.e., the minimum unmatchable set that occurs on "1" answer on $e$.

He then chooses the answer on $e$ so as to make $S_{i+1}^*$ the larger of $S^0(e), S^1(e)$.

The heart of the argument is the following claim.

CLAIM 3.3. *If $|S_{i+1}^*| = s$ then there is a minimal unmatchable set $S_i^*$ with $|S_i^*| \leq 2s$.*

*Proof.* Assume $e$ is asked in step $i+1$. By the above strategy, $|S_{i+1}^*| = \max(|S^0(e)|, |S^1(e)|)$. It is easy to see that $S = S^0(e) \cup S^1(e)$ cannot be matched into $V$ in $G_i$. Thus, $S$ contains an unmatchable set for step $i$ of cardinality no more than $|S^0(e) \cup S^1(e)| \leq 2 \cdot \max(|S^0(e)|, |S^1(e)|) = 2s$. □

We can now complete the proof of Theorem 3.2 by the following argument. At the beginning $|S_0^*| \geq n/4$, at the end $|S_i^*| = 1$ and by the claim the cardinality of the minimal unmatchable set does not decrease by more than a factor of 2. We conclude that at some step $j$, $n/16 \leq |S_j^*| \leq n/8$, with $|N_j(S_j^*)| < |S_j^*|$. However, by the expansion property of $G$, $|N_G(S_j^*)| \geq 2|S_j^*|$. Since at each step $N_i(S)$ can drop by at most $d$ for any set $S$, at least $|S_j^*|/d = \Omega(n)$ edges were probed up to step $j$. □

**3.2. Gaps between $N(F)$ and $R(F)$.** In this section we construct two search problems for which the randomized complexity is large while the nondeterministic complexity is small. The nondeterministic complexity of the first problem is constant while its randomized complexity is $\Omega(n^{\frac{1}{6}})$. The second problem has $O(\log n)$ nondeterministic complexity and its randomized complexity is $\Omega(n/\log n)$. The proof of the lower bound on the randomized complexity of the first problem is by proving a lower bound on the distributional complexity. (Yao [21] has shown that this is sufficient.) The proof for the second is by an indirect reduction to a communication complexity game.

**3.2.1. A problem with $N(F) = 3$ and $R(F) \in \Omega(n^{\frac{1}{6}})$.** Let GRID be the following problem on $n = m^2 - 1$ variables. Consider an $(m+1) \times (m+1)$ matrix where the entry at the bottom left corner contains a one and the top row and rightmost column are all zero. The $n$ input bits determine the rest of the entries of the matrix.

For any 0-1 assignment to the rest of the matrix, the goal is to find an entry that is one and its upper and right neighbors are zero. It is not hard to see that such a configuration always exists.

This example is inspired by the lower bound argument of Hirsch, Papadimitriou, and Vavasis [8] for finding Brouwer fixed points and discussions with Noga Alon on extending it to the random case.

THEOREM 3.4. *$N(\text{GRID}) = 3$, $R(\text{GRID}) = \Omega(n^{\frac{1}{6}})$, and $D(\text{GRID}) = \Theta(\sqrt{n})$.*

*Proof.* The fact that $N(\text{GRID}) = 3$ is clear. The theorem is established by the following lemmata.

LEMMA 3.5. *$R(\text{GRID}) = \Omega(n^{\frac{1}{6}})$.*

*Proof.* A basic result of Yao [21] asserts that in order to prove lower bounds on randomized decision tree complexity it is sufficient to show a distribution on the inputs such that any deterministic algorithm requires a high expected number of queries. The distribution for which we claim the lower bound is defined as follows: a random upward and rightward path starting from the bottom left corner of the matrix and ending at the top row or right column is picked uniformly from all such paths. The entries along the path receive the value "1" and the rest receive the value "0."

We claim that any deterministic algorithm requires $\Omega(m^{\frac{1}{3}}) = \Omega(n^{\frac{1}{6}})$ queries on the average to discover the end point of the path, which is the only point where the desired configuration occurs. We need the following claim.

CLAIM 3.6. *Let $A, B, C_1, \ldots, C_k$ be points in the matrix such that $B$ is of Manhattan distance at least d from $A$ in the downward and left direction, and each $C_i$, $1 \le i \le k$ is of Manhattan distance at least d from $B$. For the above distribution on paths and $1 \le k \le \frac{\sqrt{d}}{2}$, the probability that a path passes through $A$ given that it passes through $B$ and avoids $C_1, \ldots, C_k$ is at most $\frac{2}{\sqrt{d}}$.*

*Proof.* Let $\mathcal{A}$ be the event that the path passes through point $A$, $\mathcal{B}$ the event that it passes through $B$ and $\mathcal{C}_i$, $1 \le i \le k$ the event that it passes through $C_i$.

$$\text{Prob}(\mathcal{A}|\mathcal{B}, \overline{\mathcal{C}_1}, \ldots, \overline{\mathcal{C}_k}) = \frac{\text{Prob}(\mathcal{A} \cap \overline{\mathcal{C}_1} \cap, \ldots, \cap \overline{\mathcal{C}_k}|\mathcal{B})}{\text{Prob}(\overline{\mathcal{C}_1} \cap, \ldots, \cap \overline{\mathcal{C}_k}|\mathcal{B})} \le \frac{\text{Prob}(\mathcal{A}|\mathcal{B})}{1 - \text{Prob}((\mathcal{C}_1 \cup, \ldots, \cup \mathcal{C}_k)|\mathcal{B})}$$

$$\le \frac{\text{Prob}(\mathcal{A}|\mathcal{B})}{1 - k \cdot \max_{1 \le i \le k}\{(\text{Prob}(\mathcal{C}_i|\mathcal{B})\ \}} \le \frac{\frac{1}{\sqrt{d}}}{1 - \frac{k}{\sqrt{d}}} \le \frac{2}{\sqrt{d}}$$

($\text{Prob}(\mathcal{A}|\mathcal{B})$ and $\text{Prob}(\mathcal{C}_i|\mathcal{B})$ are bounded by $\frac{1}{\sqrt{d}}$ since they correspond to $\max_{1 \le j \le d'} \binom{d'}{j}/2^{d'}$). □

We continue now with the proof of Lemma 3.5. Let $T$ be any deterministic decision tree for GRID. We will give additional information to queries of $T$. At any stage $i$ we provide $T$ with a prefix of the path. Informally, the intuition is the following: at each stage a contiguous initial segment of the path will be known to $T$. If the next query is to a point of distance not too far from that known prefix, we provide $T$ with an additional segment of the path, a segment that is long enough to make sure that knowing it will determine the value of the query. Thus, the conditional probability of future queries will not depend on the answer to that query. If on the other hand, $T$ queries a point "far away" from the prefix of the path that he already knows, then there is a good chance that he will get a "0" answer and learn very little (by Claim 3.6).

Formally, at each stage $i$ we provide $T$ with the prefix of the path of length $j_i m^{2/3}$, i.e., with all the points on the path up to Manhattan distance $j_i m^{2/3}$ from the origin (the bottom left corner). The length of the prefix is determined as follows. We start with $j_0 = 0$. If by the $i$th stage $\ell \geq 0$ is the largest number such that there exists a set of $\ell$ queries $q_1, q_2, \ldots, q_\ell$ so that for all $1 \leq h \leq \ell$ $q_h$ is to a point of distance $D_h$ with $(j_{i-1} + h - 1)m^{2/3} + 1 \leq D_h \leq (j_{i-1} + h)m^{2/3}$ from the origin, then we provide all the points on the path up to distance $(j_{i-1} + \ell)m^{2/3}$ and set $j_i$ to be $j_{i-1} + \ell$. In such a case we say that $q_1, \ldots, q_\ell$ contribute to $j_i$. Note that each query can contribute to $q_r$ for at most one $r$. It follows that for all $i$ we have that $j_i \leq i$.

Consider now what happens whenever an execution of $T$ discovers the end point of the path with less than $\frac{1}{4}m^{1/3}$ steps. There must be the first step $k$ for which the $k$th query is at distance larger than $(j_{k-1} + 1) \cdot m^{2/3}$ and was answered "1." Let $B$ be the end point of the prefix at stage $k - 1$ (i.e., of distance $j_{k-1} \cdot m^{2/3}$ from the origin), let $A$ be the $k$th query and let $C_i$, $i \leq k - 1$ be all the previous queries that were answered by "0" and were to distance further than $(j_{k-1} + 1) \cdot m^{2/3}$. By Claim 3.6, we have that for any $1 \leq k \leq 1/4m^{1/3}$ the probability that such an event will occur is at most $\frac{2}{\sqrt{m^{2/3}}} = \frac{2}{m^{1/3}}$. Therefore the probability that such an event will occur for some $1 \leq k \leq 1/4m^{1/3}$ is at most $1/2$ and therefore the expected number of queries is at least $\Omega(m^{1/3})$. □

By replacing the grid with an expander we can get a problem with sharper bounds. Let $G$ be a 3-regular expander with $n$ edges and let $u$ be some node in $G$. Associate the $n$ inputs with the edges of $G$. Thus every assignment to the inputs defines a subgraph $G'$ by the edges that are assigned "1." The problem ODD is to find a node other than $u$ with an odd degree in $G'$ or find that $u$ has even degree in $G'$.

ODD is a valid search problem by the fact that every graph has an even number of nodes with odd degree. Therefore $N(\text{ODD}) = 3$. Using the fact that expanders are rapidly mixing, i.e., that a random walk in an expander gets to a node that is almost random after $O(\log n)$ steps, we can show that $R(\text{ODD})$ is $\Omega(n^{\frac{1}{3}})$. The intuition is that as in the grid, we concentrate the probability on walks of length $n^{2/3}$ that start from $u$. A query to a "far enough" point is unlikely to be on the walk, thus the tree would have to ask many vertices along the walk in order to find its end. The advantage here is that "far enough" becomes quite small ($O(\log n)$ due to the mixing property).

We conjecture however that $R(\text{ODD})$ is $\Theta(n)$.

LEMMA 3.7. $D(\text{GRID}) = O(m) = O(\sqrt{n})$.

*Proof.* There is a deterministic decision tree of complexity $O(m)$ that solves the problem. It asks all entries in the $\lfloor \frac{m}{2} \rfloor$-row. If there is any "1" entry, there must be an answer in the upper half of the matrix. Otherwise, there must be an answer in the lower part of the matrix. The decision tree probes next the relevant half of the $\lfloor \frac{m}{2} \rfloor$-column and recurses respectively. □

LEMMA 3.8. $D(\text{GRID}) = \Omega(m)$.

*Proof.* There is a simple adversary strategy that can force any deterministic algorithm to query at least $m - 1$ locations. The adversary maintains a contiguous path of 1's from the bottom left location (initially this path contains just the bottom left point). It also maintains a direction for the path which is either horizontal or vertical. Given a query, if it is not in the same row or column as the end point of the path or it is on the same row (column) and the direction is vertical (horizontal), the adversary answers 0. If it is, say, in the same row as the end point of the path and the direction is horizontal, then if in all the columns between the end point and current query point a query has been made, then the adversary answers 1 and gives away 1's for all the locations between the end point and query point. If not, he answers the query by 0, finds the first column in which a query has not been yet made, fills the row with 1's up to that point and switches the direction to vertical. The other case is treated similarly.

It is easy to see that this strategy maintains the invariant that, at any step, the current path can be augmented to the top row or right column. Moreover, the adversary answers 1 in a row (column) only if all columns (rows) between the current end point and the query have points that had already been queried. So, every 1 in $(i, j)$ position is discovered after at least $\max(i, j) - 1$ steps. □

This concludes the proof of Theorem 3.4. □

**3.2.2. A problem with $N(F) = O(\log n)$ and $R(F) = \Omega(n/\log n)$.** Our next example is of nondeterministic complexity $O(\log n)$ and randomized complexity of $\Omega(n/\log n)$. The lower bound on the randomized complexity is based on a reduction from a problem in communication complexity.

Let $K_{3m}$ be the complete graph on $3m$ vertices. Let $P_m$ be the set of all $m$-matchings in $K_{3m}$, i.e., the set of all $m$ pairwise disjoint edges. Let $Q_m$ denote the set of all $(m-1)$-subsets of vertices of $K_{3m}$. Note that for every member $p \in P_m$ and every member $q \in Q_m$ there is an edge $e \in p$ such that $e \cap q = \emptyset$. Our search problem is essentially to find such an edge on an input $(p, q) \in P_m \times Q_m$. However, we use some Boolean encoding of the problem.

We encode the sets by permutations (as explained below) and permutations by permutation networks. A permutation network is a graph that "realizes" permutations. Formally, let $G$ be a directed acyclic graph with $k$ sources called input nodes, $k$ sinks, called output nodes, and some other nodes called switches. Each input node, has one out-going edge and each output node has one in-going edge. Each switch has two in-going and two out-going edges and can be assigned to select one of the two permutations that map the two in-going edges to the two out-going edges. Clearly, a setting to each switch defines $k$ paths in $G$, each from an input node to an output node. Thus it defines a mapping from the $k$ inputs to the $k$ outputs. It is easy to see that this mapping is always a permutation.

Let $S_k$ be the symmetric group on $k$ elements. A graph $G$ as above is called a "$k$-permutation network" if for every $\pi \in S_k$ there is a setting to the switches so that the mapping defined by it is $\pi$. For details of construction of some permutation networks see [12], [16].

*Fact.* For every $k$, a $k$-permutation network of size $O(k \log k)$ and depth $O(\log k)$ can be constructed efficiently (e.g., the shuffle-exchange network is a simple construction).

We can formally define now our $n$ variables search problem MATCH: let $k = 3m$. Fix two disjoint $k$-permutation networks and let $n$ be the total number of switches ($n = O(m \log m)$). The input is an assignment to the switches of each network, interpreted as two permutations $\pi_1, \pi_2$ on $k$ elements. The first permutation encodes an $m$-matching $p$; $\pi_1(i)$ is matched to $\pi_1(i+m)$ for every $1 \leq i \leq m$. The second encodes a set $q$ of size $m-1$ by $q = \{\pi_2(1), \ldots, \pi_2(m-1)\}$. The search problem is to find an edge as above.

THEOREM 3.9. *$N(\text{MATCH}) = O(\log n)$ and $R(\text{MATCH}) = \Omega(n/\log n)$.*

*Proof.* $N(F) = O(\log m)$ since all one has to do is to "guess" $i$ and find $j, r$ such that $\pi_1(i) = j$, $\pi_1(i+m) = r$ (this is an edge in $p$). In addition, check that $j$, $r \notin q$ by "guessing" $s$, $t > m-1$ such that $\pi_2(s) = j$, $\pi_2(t) = r$. This takes $O(\log m)$ probes.

The lower bound on the randomized complexity follows from (i) the result of Raz and Wigderson [18] on the complexity of the problem of finding the desired edge in the communication complexity setting where one party has $p \in P_m$ as its input and the other party has $q \in Q_m$ as its input. They showed that $\Omega(k)$ bits must be transmitted. (ii) The fact that any lower bound in the communication complexity model is also a lower bound in the decision tree model, since the players can simulate the decision tree for each other (transmit the current bit being probed). We don't give a detailed definition of the communication complexity model here. For further information see [11], [18], and [1]. □

**3.3. Simultaneous large gaps; $N(F) << R(F) << D(F)$ .** In this section we construct a problem with simultaneous exponential gaps among $N(F)$, $R(F)$, and $D(F)$. We remark here that the deterministic lower bound is based on an interesting application of an upper bounds for an online coloring algorithm.

Let $r$ be an integer. Let $G = (V, E)$ be an $m$ vertex graph that is not $r$-colorable and $n = m \log r$. The $r$-coloring search problem for $G$, denoted by $\mathrm{COL}_r(G)$, is the following $n$ variable problem: every assignment of the $n = m \log r$ variables is interpreted as an $r$-coloring of $G$. The goal is to find two neighbors with the same color. Clearly such a configuration always exists; we call such a configuration a "monochromatic edge."

THEOREM 3.10. *Let $r = (\log m)^2$, $d = 16r^2$, and let $G = (V, E)$ be a $d$-regular $m$-vertex Ramanujan expander as constructed by Lubotzky, Phillips, and Sarnak* [13]. *Then,*

1. $N(\mathrm{COL}_r(G)) = O(\log r) = O(\log \log m)$
2. $R(\mathrm{COL}_r(G)) = \Omega(\sqrt{r})$ *and* $R(\mathrm{COL}_r(G)) = O(r \log r)$
3. $D(\mathrm{COL}_r(G)) = \Omega(2^{\frac{1}{6}\sqrt{r}}) = \Omega(m^{1/6})$

*Proof.* We first have to show that the search problem is indeed valid, that is, to prove that $G$ is not $r$-colorable. This as well as the rest of the proof will follow from the lemmata below.

LEMMA 3.11. *For every $r$-coloring of $G$ there exist at least $8 \cdot r \cdot m$ monochromatic edges.*

*Proof.* For a set $S \subseteq V$ let $E(S) = \{(u, v) |\ u, v \in S\}$. Let $S_i$, $1 \le i \le r$ be the color classes under a coloring of $V$ with $r$ colors. The number of monochromatic edges is thus $\sum_{i=1}^{i=r} |E(S_i)|$. However, by the expansion properties of $G$, for every $S' \subset V$, $|E(S')| \ge \frac{d \cdot |S'|^2}{m} - 2\sqrt{d}|S'|$. Thus, the number of monochromatic edges is at least

$$\sum_{i=1}^{i=r} \left( \frac{d \cdot |S_i|^2}{m} - 2\sqrt{d}|S_i| \right) \ge \frac{d}{m} \left( \sum_{i=1}^{i=r} |S_i|^2 \right) - 2\sqrt{d}m$$
$$\ge \frac{d}{mr} (\sum_i |S_i|)^2 - 2\sqrt{d}m = \frac{dm}{r} - 2\sqrt{d}m = 8 \cdot r \cdot m. \qquad \square$$

LEMMA 3.12. $R(\mathrm{COL}_r(G)) = O(r \log r)$ *and* $R(\mathrm{COL}_r(G)) = \Omega(\sqrt{r})$.

*Proof.* The upper bound follows from Lemma 3.11. There are at least $8 \cdot r \cdot m$ "monochromatic" edges for every $r$-coloring; thus selecting an edge at random and probing the $2 \log r$ bits that define its end points' colors results in a witness with probability at least $8mr/dm \in \Omega(1/r)$. Therefore we get that the expected number of queries is $O(r \log r)$.

The lower bound follows by showing a "hard" distribution (as in Yao [21]). The distribution is uniform on all $r$-colorings. It is easy to see (by the "birthday paradox"), that any deterministic tree must probe at least $\sqrt{r}$ vertices in order to hit the same color twice with constant probability (in particular to find two neighbors with the same color). $\square$

LEMMA 3.13. *Let $k = \frac{2}{3}\log_{d-1} m$. For every integer $s$, every induced subgraph $G'$ of $G$ on at most $t = s^{k-1}$ vertices has a vertex of degree at most $s$.*

*Proof.* By [13] $G$ has no cycles of length less than $2k = \frac{4}{3}\log_{d-1} m$. Let $G'$ be the smallest subgraph for which every vertex has degree at least $s+1$. For a vertex $v \in G'$ let $S_0 = \{v\}$ and $S_i = \{u \mid (u,v) \in G',\text{ and } v \in S_{i-1}\}$. Every $u \in S_i$, $i < k$ has just one neighbor in $S_{i-1}$ and no neighbors in $S_i$ since otherwise $G'$ has a cycle of length smaller than $2k$. However, since the degree of every such $u$ is at least $s+1$ we get that $|S_i| \geq s \cdot |S_{i-1}|$ for all $i < k$ which gives that $G'$ contains at least $s^{k-1}$ vertices. $\square$

LEMMA 3.14. $D(\mathrm{COL}_r(G)) = \Omega(2^{\frac{1}{6}\sqrt{r}}) = \Omega(m^{1/6})$.

*Proof.* Let $s = \sqrt{r}$. By Lemma 3.12 every induced subgraph $G'$ of size at most $t = s^{k-1} = \Omega(2^{\frac{1}{6}\sqrt{r}})$ has a vertex of degree of at most $s$. It follows that it can be *online* colored by $s \cdot \log t < r$ colors, since Irani [9] has shown that the greedy algorithm has this performance. This means that as long as the decision tree probes no more than $t$ nodes, the adversary can correctly online color the induced subgraph of the probed nodes so that no monochromatic edge occurs. $\square$

This concludes the proof of Theorem 3.10 $\square$

## 4. The case of $N(F) = 2$.

In this section we investigate decision problems for which $N(F) = 2$, i.e., those which correspond to unsatisfiable 2-CNF formulae. It turns out that in that case the situation is different from the general case. Namely, $D(F)$ can be nearly characterized. It follows also that for $n$-variable problems $D(F) = O(\log n)$ for any such $F$. However, $R(F)$ may still be small in comparison to $D(F)$.

Let $f$ be an unsatisfiable CNF formula. We say that a subformula $f'$ of $f$ is critical if it is unsatisfiable but deletion of any clause makes it satisfiable.

THEOREM 4.1. *Let $F$ be an $n$-variable search problem represented by a 2-CNF formula $f(F)$. Let $f'$ be a critical subformula with minimum number of clauses. Let $k$ be the number of variables in $f'$ and $m$ be the number of clauses in $f'$. Then $k \geq m/2$ and $\log m \leq D(F) \leq 2 + \log m$.*

*Proof.* Let $T$ be a decision tree for $F$. Look at the set of clauses in its leaves. Every input reaches one of those clauses and falsifies it. So, the subformula $F'$ defined by the clauses of the tree is unsatisfiable, i.e., it has at least $m$ clauses. We conclude that the size of $T$ is at least $m$ and its depth is at least $\log m$ which proves the lower bound on $D(F)$.

To prove the upper bound define for any unsatisfiable 2-CNF formula $F$ the (standard) directed graph $G(F)$, associated with $F$ : $V(G)$ is the set of $2n$ literals. For every clause $(\alpha \vee \beta)$, $(\overline{\alpha} \rightarrow \beta)$, and $(\overline{\beta} \rightarrow \alpha)$ are edges in $E(G(F))$. For every single variable clause $x$, $(\overline{x} \rightarrow x)$ is an edge of $G(F)$.

CLAIM 4.2. *For any unsatisfiable 2-CNF formula $F$ let $G(F)$ be its graph, then there is a variable $x$ such that there is a directed path from $x$ to $\overline{x}$ and a directed path from $\overline{x}$ to $x$ in $G(F)$.*

*Proof.* The proof is by induction on the number of variables of $F$. The claim is easily checked for 2 variable formulae. Assume $F$ is unsatisfiable. Choose any variable $x$ in $F$ and for any possible two clauses $(x \vee y)$, $(\overline{x} \vee z)$ produce the "resolvant" $(y \vee z)$. The formula $F_1$ obtained by deleting every clause that contains $x$ or $\overline{x}$ and adding the new clauses is unsatisfiable. By induction hypothesis there is some $y$ such that $G(F_1)$ has a path $P_1$, from $y$ to $\overline{y}$ and a path $P_2$, from $\overline{y}$ to $y$. However, every edge $(u,v)$ in

$G(F_1)$ is either an edge in $G(F)$ or is the result of resolving two clauses of the form $(x \vee \overline{u})$ and $(\overline{x} \vee v)$. But then, the associated edges $(u, x), (x, v)$ are a path from $u$ to $v$ in $G(F)$. Thus every path from $a$ to $b$ in $G(F_1)$ has a corresponding path from $a$ to $b$ in $G(F)$, in particular so do $P_1$, $P_2$. □

Let $f'$ be a critical subformula of $F$ and let $G(f')$ be its associated directed graph. By the claim there is a variable $x$ for which there is a directed path $P_1$ from $x$ to $\overline{x}$ and a directed path $P_2$ from $\overline{x}$ to $x$. This leads to the following decision tree for $F$. First $x$ is probed. If $x = 1$ the decision tree will find an edge in $P_1$ which is directed from "1" to "0" by binary search along $P_1$. If $x = 0$ it will do the same thing on $P_2$. Such an edge $(u, v)$ corresponds to a clause $(\overline{u} \vee v)$ which is falsified (since $u = 1, v = 0$). Every clause contributes two edges to the graph so the length of each of the paths is no more than $2m$. We get the bound of $2 + \log m$ on the number of probes in the binary search.

We may take $P_1$, $P_2$ above to be simple paths, in particular the length of the paths is bounded by $k$ too. So, one gets an upper bound of $1 + \log k$ as well, and so $k \geq m/2$ by the lower bound. This concludes the proof of Theorem 4.1. □

COROLLARY 4.3. *For every 2-CNF search problem $F$ on $n$ variables, $D(F) = O(\log n)$.* □

We show now that the randomized complexity can be much smaller than the deterministic, and in other cases the largest possible.

Let $G_1$ be a constant degree Ramanujan expander on $n$ vertices of the type constructed by [13]. $\mathrm{COL}_2(G)$ is the 2-coloring search problem as defined in §3.3.

THEOREM 4.4. $N(\mathrm{COL}_2(G_1)) = 2$, $R(\mathrm{COL}_2(G_1)) = O(1)$, $D(\mathrm{COL}_2(G_1)) = \Omega(\log \log n)$.

*Proof.* If indeed $G_1$ is not two-colorable then clearly $N(\mathrm{COL}_2(G_1)) = 2$ from the definition of the problem. The fact that $G_1$ is not two-colorable and $R(\mathrm{COL}_2(G_1)) = O(1)$ follows from the same arguments as in the proof of Lemmas 3.11, 3.12 with $r = 2$. The proof that $D(\mathrm{COL}_2(G_1)) = \Omega(\log \log n)$ follows from Theorem 4.1 above since a critical subformula for $\mathrm{COL}_2(G_1)$ corresponds to the edges of a nontwo-colorable subgraph of $G_1$ (in particular it must contain an odd cycle). However the cycles of $G_1$ are of length $\Omega(\log n)$ [13]. □

Let $G_2$ be an odd cycle of length $n$.

THEOREM 4.5. $N(\mathrm{COL}_2(G_2)) = 2$, $R(\mathrm{COL}_2(G_2)) = \Omega(\log n)$.

*Proof.* $N(\mathrm{COL}_2(G_2)) = 2$ is obvious. Following Yao's technique, the distribution will be uniformly concentrated on the $n$ different inputs coloring, the $i$th being the one that colors correctly all edges except the $i$th edge.

A deterministic tree of average depth $d$ must have at least $1/2$ of the inputs reaching leaves of depth no more than $2d$ (from our special set of $n$ inputs). Since there are no more than $2^{2d}$ leaves of depth $2d$, at least $\frac{n}{2^{2d+1}}$ inputs arrive at the same leaf. However, no two inputs from our special set can arrive at same leaf since every such input has a different witness. So, $n \leq 2^{2d+1}$ or $d = \Omega(\log n)$. □

## 5. Appendix.

**5.1. The connection to the resolution problem.** Here we present an observation of V. Chvatal and E. Szemerédi [5] that relates the complexity of the resolution process for an unsatisfiable formula $F$ and its deterministic decision tree, via a generalization of decision trees (branching programs).

A *resolution* for an unsatisfiable CNF formula $F$ is a process that proves that the formula is unsatisfiable. It generates additional clauses which should be satisfiable if $F$ is satisfiable until a contradiction is obtained (the empty clause). The resolution process was first defined by Blake [3] and became popular as a theorem-proving technique by Robinson [17] and Davis and Putnam [6]. Formally, let $F$ be an unsatisfiable CNF formula with clauses $\mathcal{C} = \{C_1, \ldots, C_m\}$. The resolution is a straight line program. In each step $l$, a clause $C^l$ is produced if either $C^l \in \mathcal{C}$ or there exist $i, j < l$, with $C^i = (x \vee \alpha)$, $C^j = (\overline{x} \vee \beta)$, and $C^l = (\alpha \vee \beta)$ where $\alpha$ and $\beta$ are disjunctions of literals. In that case we say that $C^l$ was obtained by resolving on $x$. The resolution is indeed a proof for $F$ if it ends with the empty clause. The size of the resolution is the smallest number of steps to reach the empty clause, denoted here by $\mathrm{RES}(F)$. The resolution process for $F$ may be described as a directed acyclic graph $G$ of in degree 0 or 2. The vertices are the clauses that are generated by the resolution process. If $C^i, C^j$ are resolved to obtain $C^l$ then the two corresponding nodes are connected by directed edges into $C^l$. The "output" node is the empty clause.

A resolution is said to be *regular* if in every directed path form input to the output node, every variable is resolved at most once.

THEOREM 5.1 ([6]). *For every unsatisfiable* CNF *formula $F$ there is a finite regular resolution that proves $F$.*

We denote by $\mathrm{RRES}(F)$ the minimum size of a regular resolution for $F$.

A *branching program* for an unsatisfiable CNF formula $F$ is a directed acyclic graph $G$ with outdegree 0 or 2 and a special source vertex $s$. Each vertex of out degree 2 is labeled by a variable. The two out-going edges are labeled by the two possible values the variable can take. Every 0-1 assignment of the variables defines a path from the source to a sink in a natural way. Each sink is labeled by a clause of $F$ such that for every assignment of the variables, the path from the source ends in a sink labeled with a clause that is unsatisfied by the assignment.

The size of branching program for an unsatisfiable CNF formula $F$, $\mathrm{BP}(F)$, is the smallest size (number of vertices) of a branching program for $F$.

A branching program is said to be *read-once* (ROBP) if in any source-sink path every variable appears at most once. Ths size of the smallest ROBP for $F$ is denoted by $\mathrm{ROBP}(F)$.

*Note.* A decision tree for $F$ is also a ROBP for $F$.

THEOREM 5.2 ([5]). *Let $F$ be an unsatisfiable* CNF *formula, then* $\mathrm{RRES}(F) = \mathrm{ROBP}(F)$.

COROLLARY 5.3. $D(F) \geq \log(\mathrm{RRES}(F))$.

We will present here the ideas of the proof of Theorem 5.2 for the sake of completeness.

*Proof.* Assume first that $T$ is a ROBP for $F$. We will associate a clause to every vertex of $T$ such that $T$ becomes a resolution graph for $F$. A vertex $v$ labeled by a variable $x$ will be associated with a clause $C(v)$ with the property that every input that reaches $v$ in $T$ falsifies $C(v)$. We associate clauses inductively from the sinks backwards. To each sink we associate the clause it is labeled with by $T$. Let $v$ be the next vertex to be associated with a clause. $v$ is labeled with a variable $x$ in $T$ and has edges $(v, u_0)$ for $x = 0$ and $(v, u_1)$ for $x = 1$. We assume, by induction, that $u_0, (u_1)$ is labeled with $C_0, (C_1)$ respectively.

CLAIM. $C_0$ does not contain $\overline{x}$ and $C_1$ does not contain $x$. □

We conclude that either $C_0 = (x \vee \alpha)$, $C_1 = (\overline{x} \vee \beta)$, or one of $C_0, C_1$ does not contain $x, \overline{x}$ at all. In the first case label $v$ with $C(v) = (\alpha \vee \beta)$. In the second case label $v$ with the clause that does not contain $x, \overline{x}$.

By the definition of the labeling, it is clear that the source node is labeled by the empty clause (since every input reaches it). Thus, the tree represents a regular resolution process for $F$ (possibly with some redundant steps that correspond to the second case of the labeling above).

Assume now that we have a resolution graph $G$ for $F$. $G$ can be transformed to be a branching program for $F$ by reversing the direction of the edges, and labeling each node by the variable used to resolve the clause it is associated with (except the sinks which are labeled by their clauses).

It can be seen that this gives a branching program for $F$ by asserting the following property. For every node $v$ originally associated with a clause $C(v)$, all inputs that arrive to $v$ (from the root) falsify $C(v)$. Furthermore, if $G$ represents a regular resolution, then it results in a ROBP since along a path each variable appears at most once. $\square$

Note that the above also proves $\mathrm{BF}(F) \leq \mathrm{RES}(F)$. We remark that in general there can be an exponential gap between these two measures (since any unsatisfiable CNF has a BP of the size of $F$ while $\mathrm{RES}(F)$ might be exponential). It is an interesting question to find a concrete model of computation that is polynomially equivalent to resolution.

## REFERENCES

[1] L. Babai, P. Frankl, and J. Simon, *Complexity classes in communication complexity theory*, in Proc. 27th Annual IEEE Symp. on Foundations of Computer Science, IEEE Computer Society Press, 1986, pp. 337–347.

[2] M. Blum and R. Impagliazzo, *Generic oracles and oracles classes*, Proc. 28th Annual IEEE Symp. on Foundations of Computer Science, IEEE Computer Society Press, 1987, pp. 118–126.

[3] A. Blake, *Canonical expressions in Boolean algebra*, Ph.D. thesis, Department of Mathematics, Chicago University, Chicago, 1937.

[4] V. Chvatal and E. Szemerédi, *Many hard examples for resolution*, J. ACM, 35 (1988), pp. 759–768.

[5] E. Szemerédi, *Personal communication*, 1991.

[6] M. Davis and H. Putnam, *A computing procedure for quantification theory*, J. ACM, 7 (1960), pp. 210–215.

[7] J. Hartmanis and L. A. Hemachandra, *One-way functions, robustness and the non-isomorphism of* NP-*complete sets*, Proc. 2nd Structure in Complexity Theory Conference, IEEE Computer Society Press, 1987, pp. 160–173.

[8] M. D. Hirsch, C. H. Papadimitriou, and S. Vavasis, *Exponential lower bound for finding Brouwer fixed points*, J. Complexity, 5 (1989), pp. 379–416.

[9] S. Irani, *On-line coloring inductive graphs*, Proc. 31 IEEE Annual Symp. of Foundations of Comput. Science, IEEE Computer Society Press, 1990, pp. 470–479.

[10] R. Impagliazzo and M. Naor, *Decision trees downward closure*, Proc. 3rd Structure in Complexity Theory Conference, IEEE Computer Society Press, 1988, pp. 29–38.

[11] M. Karchmer and A. Wigderson, *Monotone circuits for connectivity require super-logarithmic depth*, Proc. 20th Annual ACM Symp. on Theory of Computing, (1988), pp. 539–550.

[12] F. T. Leighton, *Introduction To Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann Publishers, 1992, pp. 439–510.

[13] A. Lubotzky, R. Phillips, and P. Sarnak, *Explicit expanders and the Ramanujan conjecture*, Proc. 18th Annual ACM Symp. on Theory of Computing, ACM Press, 1986, pp. 240–246.

[14] N. Nisan, CREW PRAMS *and decision trees*, Proc. 21st ACM Symp. on Theory of Computing, ACM Press, 1989, pp. 327–335.

[15] C. H. Papadimitriou, *Graph theoretic lemmata and complexity classes*, Proc. 31st IEEE Symp. on Foundations of Computer Science, IEEE Computer Society Press, 1990, pp. 794–801.

[16] N. Pippenger, *Communication Networks*, in Handbook Of Theoretical Computer Science, MIT Press, Cambridge, MA, 1990, pp. 807–833.

[17] J. A. Robinson, *A machine-oriented logic based on the resolution principle*, J. ACM, 12 (1965), pp. 23–41.

[18] R. RAZ AND A. WIGDERSON, *Monotone circuits for matching require linear depth*, Proc. 22th ACM Symp. on Theory of Computing, ACM Press, 1990, pp. 287–292.

[19] G. TARDOS, *Query complexity, or why is it difficult to separate* $NP^A \cap \mathrm{CO}NP^A$ *from* $P^A$ *by a random oracle?*, Combinatorica, 9 (1989), pp. 385–392.

[20] L. G. VALIANT, *General purpose parallel architectures*, in Handbook of Theoretical Computer Science, MIT Press, Cambridge, MA, 1990, pp. 945–971.

[21] A. C. YAO, *Probabilistic computation, towards a unified measure of complexity*, Proc. 18th IEEE Symp. on Foundations of Computer Science, IEEE Computer Society Press, 1977, pp. 222–227.

SIAM J. DISC. MATH.
Vol. 8, No. 1, pp. 133–153, February 1995

# SEQUENTIAL AND SIMULTANEOUS LIFTINGS OF MINIMAL COVER INEQUALITIES FOR GENERALIZED UPPER BOUND CONSTRAINED KNAPSACK POLYTOPES [*]

HANIF D. SHERALI[†] AND YOUNGHO LEE[‡]

**Abstract.** A family of facets for the GUB (Generalized Upper Bound) constrained knapsack polytope that are obtainable through a lifting procedure is characterized. The sequential lifting procedure developed herein computes lifted coefficients of the variables in each GUB set simultaneously, in contrast with the usual sequential lifting procedure that lifts only one variable at a time. Moreover, this sequential lifting procedure can be implemented in polynomial time of complexity $O(nm)$, where $n$ is the number of variables and $m(\leq n)$ is the number of GUB sets. In addition, a characterization of the facets obtainable through a simultaneous lifting procedure is derived. This characterization enables us to deduce lower and upper bounds on the lifted coefficients. In particular, for the case of the ordinary knapsack polytope, a known lower bound on the coefficients of lifted facets derived from minimal covers was further tightened.

**Key words.** GUB knapsack polytope, minimal GUB covers, reformulation–linearization technique, lifting, facets

**AMS subject classifications.** 90C10, 90C27

## 1. Introduction.

Consider the GUB (Generalized Upper Bound)-constrained, or multiple-choice, knapsack problem defined as follows:

$$\text{(GKP)} \qquad \text{minimize} \left\{ \sum_{j\in N} c_j x_j : \sum_{i\in M}\sum_{j\in N_i} a_j x_j \geq b,\ \sum_{j\in N_i} x_j \leq 1\ \forall\, i \in M,\right.$$
$$\left. x_j \in (0,1)\ \ \forall\, j \in N \right\},$$

where the data are all integers, $N = \{1, ..., n\}$, $M = \{1, ..., m\}$, and where $\cup_{i\in M} N_i \equiv N$, with $N_i \cap N_j = \phi$ for $i,\ j \in M,\ i \neq j$. Johnson and Padberg [7] show that any GUB knapsack problem with arbitrarily signed coefficients $b$ and $a_j,\ j \in N$, can be equivalently transformed into a form with $b > 0$ and with $0 < a_j \leq b\,\forall\, j \in N$, and they relate facets of the transformed problem with those of the original problem. Hence, without loss of generality, we will also assume that $b > 0$ and that $0 < a_j \leq b\,\forall\, j \in N$. Note that if $|N_i| = 1\,\forall\, i \in M$, then problem GKP is the *ordinary* 0-1 *knapsack problem.*

There are many useful applications of model GKP. As suggested in Sinha and Zoltners [13], this model is appropriate for capital budgeting problems having a single resource and where the investment opportunities are divided into disjoint subsets. Balintify et al. [3] identify another application in menu planning for determining what food items should be selected from various daily menu courses to maximize an individual's food preference, subject to a calorie constraint. More importantly, model GKP frequently arises as a subset of large-scale real-world 0-1 integer programming

[*] Received by the editors May 6, 1992; accepted for publication (in revised form) January 13, 1994. This material is based upon work supported by National Science Foundation grant DDM-9121419 and Air Force Office of Scientific Research 2304/B1.

[†] Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0118.

[‡] US WEST Advanced Technologies, Applied Research, 4001 Discovery Drive, Boulder, Colorado 80303.

problems. As demonstrated in the results of Crowder, Johnson, and Padberg [4] and Hoffman and Padberg [6], even a partial knowledge of the polyhedral structure of ordinary and GUB-constrained knapsack polytopes can significantly enhance the overall performance of branch-and-cut algorithms. Moreover, Martin and Schrage [8] and Hoffman and Padberg [6] present logical implications that can be derived from GUB-constrained knapsack polytopes in the context of coefficient reductions for 0-1 integer programming problems. In the same spirit, model GKP can also be used to generate classes of valid inequalities for certain scheduling polytopes (see Lee and Sherali [12] and Wolsey [14]) to tighten their underlying linear programming relaxations. In this regard, Wolsey [14] defines a "GUB cover" inequality for problem GKP and presents some implementations of GUB cover inequalities for solving machine sequencing problems, generalized assignment problems, and variable-upper-bounded flow problems with GUB constraints. He also shows that for a special case of a GUB-constrained knapsack problem, this class of inequalities is sufficient to describe the entire convex hull. However, we will be concerned in this paper with the polyhedral properties of the convex hull of feasible solutions to problem GKP through an extension of the well-known minimal cover inequalities for the ordinary knapsack polytope. Johnson and Padberg [7] also briefly treat a partial characterization of the facets defining the convex hull of feasible solutions to GKP. In particular, they identify conditions under which facet coefficients would be zero, ordered in magnitude within sets, or have the same magnitude within sets. We will, however, be concerned with the actual generation of facets via a lifting process, and we will provide a complete characterization of facets obtainable in this fashion.

In §2 below, we present a class of valid inequalities for problem GKP obtained by a generalization of the minimal cover inequalities for the ordinary knapsack polytope. We also develop a necessary and sufficient condition for such an inequality to define a facet of a lower dimensional polytope. Subsequently, in §3, we develop a sequential lifting procedure to obtain a family of facets. The sequential lifting procedure developed herein computes lifted coefficients of the variables in each GUB set simultaneously, in contrast with the usual sequential lifting procedure that lifts only one variable at a time. Moreover, we show that this sequential lifting procedure can be implemented in polynomial time of complexity $O(nm)$. In §4, we use the reformulation–linearization technique of Sherali and Adams [11] to easily characterize facets obtainable through a simultaneous lifting procedure. This characterization enables us to derive lower and upper bounds on the lifted coefficients. Finally, in §5, for the special case of the ordinary knapsack polytope, we use this analysis to further tighten a known lower bound on the coefficients of lifted facets derived from minimal covers.

**2. Valid inequalities from minimal GUB covers.** Denote the constraint set of model GKP as

$$X \equiv \left\{ x \in (0,1)^n : \sum_{i\in M}\sum_{j\in N_i} a_j x_j \geq b,\ \sum_{j\in N_i} x_j \leq 1\ \forall\, i \in M \right\}.$$

We start by introducing some notation. For $K \subseteq N$ let $M_K = \{i \in M : j \in N_i$ for some $j \in K\}$. Also, for $k \in N$, we denote $M_{\{k\}}$ simply as $M_k$. For each $i \in M$, define a *key index* $j(i)$ such that $j(i) \in \arg\max_{j\in N_i}(a_j)$. Similarly, given any $B \subseteq N$, for each $i \in M_B$, define a *key index* $j_B(i)$ such that $j_B(i) \in \arg\max_{j\in N_i\cap B}(a_j)$. For $A \subseteq M$, denote $A_+ = \{j(i) : i \in A\}$. Similarly, for $B \subseteq N$, denote $B_+ = \{j_B(i) : i \in M_B\}$, and let $B_- = B - B_+$.

Let us suppose that for each $k \in N$

$$a_k + \sum_{i \in (M - M_k)} a_{j(i)} \geq b. \tag{1}$$

Otherwise, $x_k = 0$ in every feasible solution to $X$. Denoting the convex hull operation by conv $(\cdot)$, let GUBKP $\equiv$ conv$(X)$, and let dim(GUBKP) be the dimension of GUBKP, which is the maximum number of affinely independent points in GUBKP minus one.

PROPOSITION 2.1. dim (GUBKP) $= n - |M_0|$, *where* $M_0 = \{i \in M : \sum_{p \in (M-i)} a_{j(p)} < b\}$.

*Proof.* By the definition of $M_0$, we must have $\sum_{j \in N_i} x_j = 1$ for each $i \in M_0$. Hence, it follows that dim(GUBKP) $\leq n - |M_0|$. To prove that dim(GUBKP) $= n - |M_0|$, it suffices to show that there exist $n - |M_0| + 1$ affinely independent points in GUBKP.

For each $i \in (M - M_0)$, we construct a set of feasible points in GUBKP as follows. For each $k \in N_i$, construct $x^{(k,i)} \equiv \{x_k = 1,\ x_j = 1$ for $j = j(p),\ \forall\, p \in (M-i),\ x_j = 0$ otherwise$\}$, and let $x^{(0,i)} \equiv \{x_j = 1$ for $j = j(p),\ \ \forall\, p \in (M-i),\ x_j = 0$ otherwise$\}$. Similarly, for each $i \in M_0$, we construct a set of feasible points in GUBKP as follows. For each $k \in N_i$, construct $x^{(k,i)} \equiv \{x_k = 1,\ x_j = 1$ for $j = j(p),\ \ \forall\, p\ \in M - i,\ x_j = 0$ otherwise$\}$. Then, the total number of distinct feasible points thus constructed is $n - |M_0| + 1$. Let $\widetilde{X}$ be the set of these distinct points $x^j$, indexed by $j = 1, \ldots, n - |M_0| + 1$. Without loss of generality, let $x^{n-|M_0|+1} \equiv \{x_j = 1$ for $j \in N_+,\ x_j = 0$ otherwise$\}$. Construct a matrix $D$ whose row vectors are $x^j - x^{n-|M_0|+1}$, $j = 1, \ldots, n - |M_0|$. Then the matrix $D$ can be readily seen to possess a block-diagonal structure, with the rows corresponding to each block being linearly independent. Hence, $x^j$, $j = 1, \ldots, n - |M_0| + 1$, are affinely independent. This completes the proof. □

Now, if dim(GUBKP) is less than $n$, then by writing each inequality constraint $i \in M_0$ as an equality constraint and using this equation to eliminate the variable that has the smallest $a_j$ coefficient, we get a full-dimensional subpolytope of dimension of $n - |M_0|$. Hence, without loss of generality, we can assume henceforth that GUBKP is a full-dimensional polytope. The following results are readily evident.

COROLLARY 2.2. *For a given* $H \subseteq N$, *define* $X(H) = X \cap \{x \in (0,1)^n : x_{j(i)} = 1,\ \forall\, i \in M_H\}$. *Then, for any* $A \subseteq M$, $\dim(\text{conv}(X(\cup_{i \in A} N_i))) = |\cup_{i \in (M-A)} N_i|$.

PROPOSITION 2.3. *For each* $j \in N_-$, *the inequality* $x_j \geq 0$ *is a facet of GUBKP.*

PROPOSITION 2.4. *The GUB constraints* $\sum_{j \in N_i} x_j \leq 1$, $i \in M$, *are facets of GUBKP.*

Balas [1] discusses the well-known classes of minimal cover inequalities for the ordinary knapsack polytope, along with its tightened variants, namely the extended and the strong minimal cover inequalities. We generalize these definitions and derive related results for the GUB-constrained knapsack polytope below. In addition, an alternate tightened variant that is peculiar to the GUB-constrained situation is derived. Note that these inequalities are different from the valid inequalities treated by Wolsey [14].

We will say that a set $K = \cup_{i \in Q} N_i$, for some $Q \subseteq M$, is called a *GUB cover* of $X$, if $\sum_{i \in M_{\overline{K}}} a_{j(i)} \leq b - 1$, where $\overline{K} \equiv N - K$. A GUB cover $K$ is called a *minimal GUB cover* of $X$ if $\sum_{i \in M_{\overline{K}}} a_{j(i)} + \min_{i \in M_K}(a_{j(i)}) \geq b$. Accordingly, we define a *minimal*

*GUB cover inequality* as the valid inequality

$$\sum_{j\in K} x_j \geq 1. \tag{2}$$

For a minimal GUB cover $K$, we define $R = \{j \in \overline{K} : a_j \geq \max_{j\in K}(a_j)\}$. An *extension of the minimal GUB cover* $K$ of $X$, denoted by $E(K)$, is defined as $E(K) = K \cup S$, where $S = \cup_{i\in M_R} N_i$.

PROPOSITION 2.5. *If* $K$ *is a minimal GUB cover of* $X$, *then the inequality defined as*

$$\sum_{j\in E(K)} x_j \geq 1 + |M_R| \tag{3}$$

*is a valid inequality for GUBKP. Moreover, this inequality dominates the minimal GUB cover inequality if* $R \neq \phi$.

*Proof.* The validity of (3) follows by verifying that if a binary $\overline{x}$ satisfies the GUB constraints and is such that $\sum_{j\in E(K)} \overline{x}_j \leq |M_R|$, then $\overline{x} \notin X$. The dominance statement is evident by rewriting (3) as $\sum_{j\in K} x_j \geq 1 + \sum_{i\in M_R}(1 - \sum_{j\in N_i} x_j)$. This completes the proof. □

The idea of using strong minimal covers to generate nondominated extensions as for the case of the ordinary knapsack problem (see Balas [1]) can be readily extended to the GUB-constrained situation as follows. We will call a minimal GUB cover $K$ *strong* if either $E(K) = N$, or else no set of the form $S = \cup_{i\in(M_K - M_{j_1})\cup\{p\}} N_i$ for any $p \in M_{\overline{E(K)}}$ is a cover, where $j_1 \in \arg\max_{j\in K}(a_j)$. That is, a set $K \subseteq N$ is a *strong minimal GUB cover* of $X$ if $K$ is a minimal GUB cover for which, if $E(K) \neq N$, then $a_{j_1} + \sum_{i\in(M_{\overline{E(K)}} - p)} a_{j(i)} \geq b \;\; \forall\, p \in M_{\overline{E(K)}}$. Hence, a minimal GUB cover $K$ is strong if there exists no minimal GUB cover of the same size as $K$ whose extension strictly contains that of $K$.

We now consider another strengthening procedure for the minimal GUB cover inequality.

PROPOSITION 2.6. *If* $K$ *is a minimal GUB cover and* $(K_1, K_2)$ *is a partition of* $K$ *with* $K_2 \neq \phi$ *such that*

$$\sum_{i\in M_{K_2}} \max_{j\in N_i\cap K_2}(a_j) + \sum_{i\in M_{\overline{K}}} a_{j(i)} < b,$$

*then the inequality*

$$\sum_{j\in K_1} x_j \geq 1 \tag{4}$$

*is valid for GUBKP and dominates the minimal GUB cover inequality* $\sum_{j\in K} x_j \geq 1$. *Moreover, if* $\min_{j\in K_1}(a_j) + \sum_{i\in M_{\overline{K}}} a_{j(i)} \geq b$, *then the inequality* (4) *is a facet of* $\operatorname{conv}(X(K_2, \overline{K}))$ *where* $X(K_2, \overline{K}) = X \cap \{x \in (0,1)^n : x_j = 0 \;\; \forall\, j \in K_2,\ x_{j(i)} = 1 \;\; \forall\, i \in M_{\overline{K}}\}$.

*Proof.* It is readily shown that the inequality (4) is valid for GUBKP. Since $\min_{j\in K_1}(a_j) + \sum_{i\in M_{\overline{K}}} a_{j(i)} \geq b$, the unit vectors $e_j$, for $j \in K_1$, are feasible to

$\mathrm{conv}(X(K_2, \overline{K}))$, and moreover, they satisfy (4) as an equality and are linearly independent. Hence, the inequality (4) is a facet of $\mathrm{conv}(X(K_2, \overline{K}))$. This completes the proof. □

Furthermore, if no such partition with $K_2 \neq \phi$ exists, i.e., if $\min_{j\in K}(a_j) + \sum_{i\in M_{\overline{K}}} a_{j(i)} \geq b$, then we have the following result.

PROPOSITION 2.7. *For a minimal GUB cover $K$, the minimal GUB cover inequality is a facet of* $\mathrm{conv}(X(\overline{K}))$ *if and only if* $\min_{j\in K}(a_j) + \sum_{i\in M_{\overline{K}}} a_{j(i)} \geq b$.

*Proof.* The proof follows easily by examining the $|K|$ linearly independent unit vectors $e_j$, one for each $j \in K$, which belong to $\mathrm{conv}(X(\overline{K}))$, and which satisfy the minimal GUB cover inequality as an equality. □

Moreover, canonical facets of GUBKP are related to inequalities (4) as follows.

PROPOSITION 2.8. *If for some $H \subseteq N$, the inequality $\sum_{j\in H} x_j \geq 1$ is a facet of GUBKP, then $K = \cup_{i\in M_H} N_i \supseteq H$ is a GUB cover such that within $K$, $H$ is a minimal set satisfying $\sum_{i\in M_H} \max_{j\in(N_i - H)}(a_j) + \sum_{i\in M_{\overline{K}}} a_{j(i)} < b$.*

*Proof.* Since $\sum_{j\in H} x_j \geq 1$ is valid for GUBKP, we have that $\sum_{j\in\overline{H}_+} a_j \leq b-1$, which can be restated as $\sum_{i\in(M-M_H)} a_{j(i)} + \sum_{i\in M_H} \max_{j\in(N_i-H)}(a_j) \leq b-1$. Since $\sum_{i\in(M-M_H)} a_{j(i)} \leq b-1$, $K = \cup_{i\in M_H} N_i$ is a GUB cover set that contains $H$ and satisfies the condition of the proposition. If $H$ is not minimal, then there exists an $s \in H$ such that $H' = H - s$ satisfies the condition of the proposition. Then, by Proposition 2.6, $\sum_{j\in H'} x_j \geq 1$ is valid for GUBKP. Since $\sum_{j\in H'} x_j \geq 1$ strictly dominates $\sum_{j\in H} x_j \geq 1$, we have a contradiction, and this completes the proof. □

*Example* 2.1. Consider the following example, where $X \equiv \{x \in (0,1)^8 : x_1 + 5x_2 + x_3 + 5x_4 + x_5 + 3x_6 + x_7 + 3x_8 \geq 9,\ x_1 + x_2 \leq 1,\ x_3 + x_4 \leq 1,\ x_5 + x_6 \leq 1,\ x_7 + x_8 \leq 1\}$.

Since for all $i \in M$, $\sum_{p\in(M-i)} a_{j(p)} \geq 9$, the convex hull of $X$ is a full-dimensional polytope by Proposition 2.1. A GUB cover is given by $K = \{1,2,3,4,5,6\}$, where $K_+ = \{2,4,6\}$ and $K_- = \{1,3,5\}$. A minimal GUB cover is given by $K = \{1,2,3,4\}$, and a minimal GUB cover inequality is $x_1 + x_2 + x_3 + x_4 \geq 1$. This minimal GUB cover does not admit any extension. However, consider a partition of $K$ such that $K_1 = \{2,4\}$ and $K_2 = \{1,3\}$. Then a strengthened minimal GUB cover inequality of the form (4) is $x_2 + x_4 \geq 1$, which is a facet of $\mathrm{conv}(X(K_2, \overline{K}))$, where $X(K_2, \overline{K}) \equiv X \cap \{x \in (0,1)^8 : x_1 = x_3 = 0,\ x_6 = x_8 = 1\}$. Note that the foregoing minimal cover is not strong, since an extension of the minimal GUB cover $K = \{3,4,5,6\}$ is $E(K) = \{1,2,3,4,5,6\} \supset \{1,2,3,4\}$, and the extended inequality of the type (3) is $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 2$. This minimal GUB cover $K = \{3,4,5,6\}$ can be verified to be strong. Moreover, since $\min_{j\in K}(a_j) + \sum_{i\in M_{\overline{K}}} a_{j(i)} = 9 = b$, by Proposition 2.7, the corresponding minimal GUB cover inequality $x_3 + x_4 + x_5 + x_6 \geq 1$ is a facet of $\mathrm{conv}(X(\overline{K})) \equiv X \cap \{x \in (0,1)^8 : x_2 = 1,\ x_8 = 1\}$.

*Remark* 2.1. In concluding this section, we remark that within the context of 0-1 programming problems that contain GUB constraints, given a fractional solution to the continuous relaxation, one can set up a separation problem using individual problem constraints along with (a subset of) the GUB constraints to possibly generate a minimal GUB cover that deletes this fractional solution. Such an approach would be similar to that used by Crowder, Johnson, and Padberg [4] (also, see Hoffman and Padberg [6]), except that this separation problem would be a GUB-constrained knapsack problem. Note that the transformation suggested by Johnson and Padberg [7] can be used to put this GUB-constrained knapsack problem in the standard form considered herein. Having generated such a minimal cover, this can either be tightened using the foregoing discussion, or it can be possibly lifted into a facet of GUBKP as discussed in §3 below.

**3. Sequentially lifted facets from minimal GUB covers.** We now consider a polynomial-time strengthening procedure that sequentially lifts a given minimal GUB cover inequality. For the case of the ordinary knapsack polytope, Balas and Zemel [15] exhibit that the sequential lifting procedure of Padberg [10], which lifts one variable at a time, obtains a facet when applied to a minimal cover inequality that is a facet of a certain lower-dimensional polytope. However, as we shall show, in the presence of GUB constraints, we need to lift all the variables in $N_p$ for each GUB constraint $p \in M_{\overline{K}}$ *simultaneously*, where $K$ is a minimal GUB cover of $X$, to obtain a lifted inequality. In particular, if the condition of Proposition 2.7 holds, then we show that the resulting inequality is a facet of GUBKP. Moreover, in the spirit of the procedure proposed by Zemel [16], we show that this type of a sequential-simultaneous lifting can also be conducted in polynomial time.

Let us begin our analysis by defining some notation. For a given $p \in M_{\overline{K}}$, define $\eta^t$, for $t \equiv j(p)$, as

$$\eta^t \equiv \min\left\{\sum_{j\in K} x_j \ : \ x \in X,\ x_j = 0 \ \ \forall\, j \in N_p\right\}$$

$$(5) \qquad \equiv \min\left\{\sum_{i\in M_K} x_{j(i)} \ : \ \sum_{i\in M_K} a_{j(i)}x_{j(i)} \ge b - \sum_{i\in(M_{\overline{K}}\,-\,p)} a_{j(i)},\right.$$
$$\left. x_{j(i)} \in (0,1)\ \forall\, i \in M_K\right\},$$

and define $\zeta^s$, for each $s \in (N_p - j(p)) \equiv (N_p - t)$, as

$$\zeta^s \equiv \min\left\{\sum_{j\in K} x_j : x \in X,\ x_s = 1\right\}$$

$$(6) \qquad \equiv \min\left\{\sum_{i\in M_K} x_{j(i)} : \sum_{i\in M_K} a_{j(i)}x_{j(i)} \ge b - a_s - \sum_{i\in(M_{\overline{K}}\,-\,p)} a_{j(i)},\right.$$
$$\left. x_{j(i)} \in (0,1)\ \forall\, i \in M_K\right\}.$$

Since $K$ is a minimal GUB cover of $X$ and GUBKP is full-dimensional and by condition (1), note that $1 \le \eta^t \le |M_K|$ and $1 \le \zeta^s \le \eta^t \ \ \forall\, s \in (N_p - t)$.

PROPOSITION 3.1. (i) *For a given minimal GUB cover $K$ of $X$, the following inequality, defined for any particular $p \in M_{\overline{K}}$ as*

$$(7) \qquad \sum_{j\in K} x_j + \sum_{s\in(N_p\,-\,t)} \alpha_s x_s + \alpha_t x_t \ge 1 + \alpha_t,$$

*where $t \equiv j(p)$, is a valid inequality for GUBKP for any $\alpha_t \le \eta^t - 1$ and for any $\alpha_s \ge -\zeta^s + \alpha_t + 1\ \forall\, s \in (N_p - t)$.*

(ii) *Moreover, if* $\alpha_t = \eta^t - 1$, $\alpha_s = \eta^t - \zeta^s \ \forall \ s \in (N_p - t)$ *and* $\min_{j\in K}(a_j) + \sum_{i\in M_{\overline{K}}} a_{j(i)} \geq b$, *then the inequality* (7) *is a facet of* $\operatorname{conv}(X(\overline{K} - N_p))$.

*Proof.* (i) Since the minimal GUB cover inequality is valid for GUBKP and (7) coincides with the minimal GUB inequality when $x_t = 1$ and $x_s = 0 \, \forall \, s \in (N_p - t)$, it is sufficient to show for establishing the validity of (7) that if $\bar{x} \in X$ and has $\bar{x}_s = 1$ for any $s \in (N_p - t)$, or if $\tilde{x} \in X$ and has $\tilde{x}_j = 0 \ \ \forall \ j \in N_p$, then such $\bar{x}$ and $\tilde{x}$ satisfy (7). That is, $\sum_{j\in K} \bar{x}_j + \alpha_s \geq 1 + \alpha_t$, and $\sum_{j\in K} \tilde{x}_j \geq 1 + \alpha_t$. By the definition of the quantities $\alpha_t$ and $\alpha_s$, in the first case, we have, using (6), that $\sum_{j\in K} \bar{x}_j + \alpha_s \geq \zeta^s - \zeta^s + \alpha_t + 1 = 1 + \alpha_t$, and in the second case, we have, using (5), that $\sum_{j\in K} \tilde{x}_j \geq \eta^t \geq 1 + \alpha_t$. Hence, the inequality (7) is valid for GUBKP.

(ii) Since the minimal GUB cover inequality is a facet of $\operatorname{conv}(X(\overline{K}))$ by Proposition 2.7, and since $\operatorname{conv}(X(\overline{K}))$ is of full dimension $|K|$ by Corollary 2.2, there exist $|K|$ linearly independent vertices of $\operatorname{conv}(X(\overline{K}))$, indexed by $x^j$, $j = 1, \ldots, |K|$, that satisfy the minimal GUB cover inequality as an equality. Since $x^j_{j(i)} = 1$, for $i \in M_{\overline{K}}$, $j = 1, \ldots, |K|$, these vertices also satisfy the inequality (7) as an equality and moreover, these vertices belong to $X(\overline{K} - N_p)$. Now, for each $s \in (N_p \ - \ t)$, let $\hat{x}^s$ be a solution of (6) such that $\zeta^s = \sum_{j\in K} \hat{x}^s_j$. Note that $\hat{x}^s \in X(\overline{K} - N_p)$. Also, for $t = j(p)$, let $\hat{x}^t$ be a solution of (5) such that $\eta^t = \sum_{j\in K} \hat{x}^t_j$, and note that $\hat{x}^t \in X(\overline{K} - N_p)$. Moreover, $\hat{x}^s$ and $\hat{x}^t$ satisfy the inequality (7) as an equality when $\alpha_s = \eta^t - \zeta^s$, and $\alpha_t = \eta^t - 1$.

By Corollary 2.2, we have that $\dim(\operatorname{conv}(X(\overline{K} - N_p))) = |K \cup N_p|$. Let $\widetilde{X} \equiv \{(x^j, 1) \text{ for } j = 1, \ldots, |K|, (\hat{x}^t, 1), (\hat{x}^s, 1) \text{ for } s \in (N_p - t)\}$ be the set of vectors obtained by adding a new component having value 1 to each vector $x^j, \hat{x}^t$, and $\hat{x}^s$ in the collection as shown. Let us show that the vectors in $\widetilde{X}$ are linearly independent. On the contrary, suppose that these vectors are linearly dependent. Then there exists a set of multipliers $\{(\lambda_j \text{ for } j = 1, ..., |K|), \delta_t, (\mu_s \text{ for } s \in (N_p - t))\} \neq 0$ such that

$$(8) \qquad \sum_{j=1}^{|K|} \lambda_j x^j + \delta_t \hat{x}^t + \sum_{s\in(N_p-t)} \mu_s \hat{x}^s = 0 \text{ and } \sum_{j=1}^{|K|} \lambda_j + \delta_t + \sum_{s\in(N_p-t)} \mu_s = 0.$$

Since for each $s \in (N_p - t)$, $\hat{x}^t_s = 0$, $x^j_s = 0$ for $j = 1, ..., |K|$, and $\hat{x}^s_s = 1$, it follows that $\mu_s = 0 \ \ \forall \ s \in (N_p - t)$. Now, if $\delta_t = 0$, then by the linear independence of $x^j$, $j = 1, \ldots, |K|$, we would have $\lambda_j = 0, \ \ \forall \ \ j = 1, \ldots, |K|$, a contradiction. Hence, without loss of generality, suppose that $\delta_t = -1$, so that (8) becomes $\hat{x}^t = \sum_{j=1}^{|K|} \lambda_j x^j$ and $\sum_{j=1}^{|K|} \lambda_j = 1$. Now, since $\hat{x}^t_t = 0$, $x^j_t = 1$ for $j = 1, \ldots, |K|$, it follows that $\sum_{j=1}^{|K|} \lambda_j = 0$, which is a contradiction. Hence, the vectors $\{(x^j, \text{ for } j = 1, \ldots, |K|), \hat{x}^t, (\hat{x}^s \text{ for } s \in (N_p - t))\}$ are affinely independent. This completes the proof. □

PROPOSITION 3.2. *Let K be a minimal GUB cover such that the corresponding minimal GUB cover inequality gives a facet of* $\operatorname{conv}(X(\overline{K}))$. *Let* $M_{\overline{K}} \equiv \{i_1, \ldots, i_k\}$ *be arbitrarily ordered, where* $k \equiv |M_{\overline{K}}|$. *Let* $M(q) = \{i_1, \ldots, i_q\} \subseteq M_{\overline{K}}$, *and let* $N(q) = \cup_{i\in M(q)} N_i$ *for* $q = 1, \ldots, k$. *For* $q = 0$, *let* $M(q) = N(q) = \phi$. *Let* $q \in \{0, \ldots, k-1\}$ *and suppose that* $\sum_{j\in K\cup N(q)} \alpha_j x_j \geq \alpha_0$ *is valid for GUBKP and is a facet of* $\operatorname{conv}(X(\overline{K} - N(q)))$. *Consider* $i_{q+1}$. *Denote* $t = j(i_{q+1})$, *and compute*

$$(9) \qquad \eta^t(q) = \min\left\{ \sum_{j\in K\cup N(q)} \alpha_j x_j : x \in X, \ x_j = 0 \ \forall \ j \in N_{i_{q+1}} \right\}.$$

*Also, for each* $s \in N(i_{q+1}) - t$, *compute*

$$\zeta^s(q) = \min\left\{ \sum_{j \in K \cup N(q)} \alpha_j x_j : x \in X,\ x_s = 1 \right\}. \tag{10}$$

*Then*

$$\sum_{j \in K \cup N(q)} \alpha_j x_j + \sum_{s \in (N_{i_{q+1}} - t)} (\eta^t(q) - \zeta^s(q)) x_s + (\eta^t(q) - \alpha_0) x_t \geq \eta^t(q) \tag{11}$$

*is* (i) *valid for GUBKP, and* (ii) *is a facet of* $\operatorname{conv}(X(\overline{K} - N(q+1)))$.

The proof of Proposition 3.2 follows the same argument used in the proof of Proposition 3.1. Note that if $q = 0$, then $\eta^t(q) = \eta^t$ and $\zeta^s(q) = \zeta^s$, as given by (5) and (6), respectively. This proposition establishes an inductive sequential procedure for generating facets for GUBKP from minimal GUB cover inequalities in the spirit of Balas and Zemel [2] for the ordinary knapsack polytope, except that a simultaneous lifting of the variables within each GUB constraint needs to be conducted in this case, as mentioned earlier.

Now, examining (7) and (11), note that a sequentially lifted inequality obtained from a minimal GUB cover inequality is of the form

$$\sum_{j \in K} x_j + \sum_{j \in \overline{K}_-} \alpha_j x_j - \sum_{j \in \overline{K}_+} \alpha_j (1 - x_j) \geq 1,$$

or

$$\sum_{j \in K} x_j + \sum_{j \in \overline{K}_-} \alpha_j x_j + \sum_{j \in \overline{K}_+} \alpha_j x_j \geq 1 + \sum_{j \in \overline{K}_+} \alpha_j. \tag{12}$$

The derivation of the coefficients $\alpha_j$ of this lifted inequality requires the solution of a sequence of GUB-constrained knapsack problems. Furthermore, the values of the coefficients depend on the sequence in which the indices $i \in M_{\overline{K}}$ are considered. For each $i \in M_{\overline{K}}$, let $\alpha_j{}'$ denote the value of $\alpha_j$ for $j \in N_i$ when $i = i_1$, i.e., when $i$ is taken to be the first index in $M_{\overline{K}}$. In other words, $\alpha_t{}' = \eta^t - 1$ for $t = j(i)$, and $\alpha_s{}' = \eta^t - \zeta^s \ \forall\ s \in (N_i - t)$. The subproblem that determines these initial coefficients has a simpler structure than the subsequent GUB-constrained knapsack problems that have to be solved to find the other coefficients of the sequence, and because of this structure, the values of $\eta^t$ and $\zeta^s$ can be easily obtained, as shown in the following propositions. (Some of the proofs are obvious, and are hence omitted.)

PROPOSITION 3.3. *Let* $K_h$ *be the index set of the* $h$ *largest* $a_{j(i)}$ *for* $i \in M_K$. *For a minimal GUB cover* $K$ *and for all* $t \in \overline{K}_+$, *we have by* (5) *that* $\eta^t = h$, *where* $h$ *is defined by*

$$\sum_{k \in K_h} a_k \geq b - \sum_{i \in (M_{\overline{K}} - M_t)} a_{j(i)} > \sum_{k \in K_{h-1}} a_k.$$

PROPOSITION 3.4. *Let* $K_h$ *be the index set of the* $h$ *largest* $a_{j(i)}$, *for* $i \in M_K$. *For a minimal GUB cover* $K$ *and for any* $s \in \overline{K}_-$, *we have* $\zeta^s = h$ *in* (6), *where* $h$ *is defined by*

$$\sum_{k \in K_h} a_k \geq b - a_s - \sum_{i \in (M_{\overline{K}} - M_s)} a_{j(i)} > \sum_{k \in K_{h-1}} a_k.$$

Note that for a given $p \in M_{\overline{K}}$, $\zeta^{j1} \geq \zeta^{j2}$ whenever $a_{j1} \leq a_{j2}$, for $j1,\ j2 \in N_p \cap \overline{K}_-$.

COROLLARY 3.5. *For a given $p \in M_{\overline{K}}$, if $\eta^t = 1$, then $\zeta^s = 1 \ \ \forall\ s \in (N_p - t)$, where $t \equiv j(p)$.*

*Proof.* Follows from the fact that $1 \leq \zeta^s \leq \eta^t \ \ \forall\ s \in N_p - t$. □

COROLLARY 3.6. *Let $t_1 \in \arg\max_{i \in M_{\overline{K}}}(a_{j(i)})$. If $\eta^{t_1} = 1$, then $\zeta^s = 1 \ \ \forall\ s \in (N_i - j(i)),\ i \in M_{\overline{K}}$, and $\eta^{j(i)} = 1 \ \ \forall\ i \in M_{\overline{K}}$.*

*Proof.* From (5), it follows that $1 \leq \eta^{j(i)} \leq \eta^{t_1} \ \ \forall\ i \in M_{\overline{K}}$. Using this fact along with Corollary 3.5 establishes the required result. □

The above results suggest a sufficient condition under which a minimal GUB cover inequality would be a facet of GUBKP.

PROPOSITION 3.7. *If the minimal GUB cover inequality is a facet of* conv$(X(\overline{K}))$ *(see Proposition 2.7), and if $\max_{j \in K}(a_j) + \sum_{i \in M_{\overline{K}} - M_{t_1}} a_{j(i)} \geq b$, where $t_1 \in \arg\max_{i \in M_{\overline{K}}}(a_{j(i)})$, then the minimal GUB cover inequality is a facet of GUBKP.*

*Proof.* Since $\max_{j \in K}(a_j) + \sum_{i \in (M_{\overline{K}} - M_{t_1})} a_{j(i)} \geq b$, we have that $\eta^{t_1} = 1$. By Corollary 3.6, all the coefficients in the lifted inequality of the form (7) are zeros. Examining (9), (10), and (11), we continue to obtain zeros for the lifted coefficients in Proposition 3.1, and so the minimal GUB cover inequality is a facet of GUBKP. □

We now show that the readily obtained coefficients $\alpha'_j$, $j \in \overline{K}$, provide bounds on the coefficients $\beta_j$, $j \in \overline{K}$, of arbitrary valid inequalities (not necessarily sequentially lifted) that have unit coefficients for all $j \in K$.

PROPOSITION 3.8. *If $\sum_{j \in K} x_j + \sum_{j \in \overline{K}_-} \beta_j x_j - \sum_{j \in \overline{K}_+} \beta_j(1 - x_j) \geq 1$ is valid for GUBKP, then $\beta_j \leq \alpha_j{}' \ \ \forall\ j \in \overline{K}_+$, and $\beta_j \geq \alpha'_j - (\alpha'_t - \beta_t) \ \forall\ j \in \overline{K}_-$, where $t = j(p)$, $p \in M_{\overline{K}}$, such that $j \in N_p$.*

*Proof.* Assume that for some $t \in \overline{K}_+$, $\beta_t > \alpha_t{}'$. From Proposition 3.1, we have that $\alpha_t{}' = \eta^t - 1$. Let $\bar{x}$ be an optimal solution for (5) such that $\eta^t = \sum_{j \in K} \bar{x}_j$. Then $\bar{x} \in X$, but

$$\sum_{j \in K} \bar{x}_j + \sum_{j \in \overline{K}_-} \beta_j \bar{x}_j - \sum_{j \in \overline{K}_+} \beta_j(1 - \bar{x}_j) = \eta^t - \beta_t = \alpha_t{}' + 1 - \beta_t < 1,$$

and so, the inequality in the proposition is violated. Hence, $\beta_t \leq \alpha_t{}' \ \forall\ t \in \overline{K}_+$. For the case of $\beta_j$, $j \in \overline{K}_-$, suppose that for some $p \in M_{\overline{K}}$, and $s \in N_p$, we have $\beta_s < \alpha'_s - (\alpha'_t - \beta_t)$ where $t \equiv j(p) \neq s$. Consider problem (6), and let $\bar{x} \in X$ solve this problem. Then, since $\alpha'_s - \alpha'_t = 1 - \zeta^s$, we have that

$$\sum_{j \in K} \bar{x}_j + \sum_{j \in \overline{K}_-} \beta_j \bar{x}_j - \sum_{j \in \overline{K}_+} \beta_j(1 - \bar{x}_j) = \zeta^s + \beta_s - \beta_t < \zeta^s + \alpha'_s - \alpha'_t = 1,$$

and so the inequality is again violated. This completes the proof. □

We now consider the task of efficiently computing the coefficients $\alpha_j$ for $j \in \overline{K}$, of a sequentially lifted facet (12). As mentioned earlier, the task of computing $\alpha_j$ for $j \in \overline{K}$ involves the solution of a set of 0-1 GUB-constrained knapsack problems. However, because of the special structure of these problems, we can easily obtain the coefficients $\alpha_j$ for all $j \in \overline{K}$ within a time complexity of $O(n|M_K|)$ by adapting the procedure due to Zemel [16] that was proposed for the ordinary knapsack problem. Toward this end, consider the following propositions. (The proofs are straightforward and are hence omitted.)

PROPOSITION 3.9. *Suppose that $\overline{x}$ is an optimal solution to the following problem with $z = \overline{z}$, and with all data integer valued:*

$$w(z) = \max\left\{\sum_{j\in N} a_j x_j : \sum_{j\in N} c_j x_j \le z, \sum_{j\in N_i} x_j \le 1 \ \forall\ i \in M,\ x_j \in (0,1)\ \forall\ j \in N\right\}.$$

*Then, $\overline{x}$ is an optimal solution to the following GUB-constrained knapsack problem*

$$P(b) : v(b) = \min\left\{\sum_{j\in N} c_j x_j : \sum_{j\in N} a_j x_j \ge b, \sum_{j\in N_i} x_j \le 1 \ \forall\ i \in M,\ x_j \in (0,1)\ \forall\ j \in N\right\}$$

*for all $b$ satisfying $w(\overline{z}-1) < b \le w(\overline{z})$.* □

PROPOSITION 3.10. *Let the function $w(z)$ and the problem $P(b)$ be as defined in Proposition 3.9, for any integers $z$ and $b$. Then, $v(b) = \min\{z : w(z) \ge b\}$.*

Now, examining (12), suppose that we have a (partially) lifted inequality of the form

$$\sum_{j\in K} x_j + \sum_{j\in T_-} \alpha_j x_j - \sum_{j\in T_+} \alpha_j (1 - x_j) \ge 1. \tag{13}$$

We want to find a lifting of (13) with respect to the variables $x_j$, $\forall\ j \in N_p$, for some $p \in (M_{\overline{K}} - M_T)$. By Proposition 3.2 and inequality (12), we have that for $t = j(p)$ and $\forall\ s \in (N_p - t)$,

$$1 + \alpha_t = \min\left\{\sum_{j\in K} x_j + \sum_{j\in T_-} \alpha_j x_j - \sum_{j\in T_+} \alpha_j(1 - x_j) : \sum_{j\in K\cup T} a_j x_j \ge b - \sum_{i\in(M_{\overline{K}} - M_T - p)} a_{j(i)}, \sum_{j\in N_i} x_j \le 1 \ \forall\ i \in M,\ x_j \in (0,1)\ \forall\ j \in N\right\},$$

and

$$1 + \alpha_t - \alpha_s = \min\left\{\sum_{j\in K} x_j + \sum_{j\in T_-} \alpha_j x_j - \sum_{j\in T_+} \alpha_j(1 - x_j) : \sum_{j\in K\cup T} a_j x_j \ge b - a_s - \sum_{i\in(M_{\overline{K}} - M_T - p)} a_{j(i)}, \sum_{j\in N_i} x_j \le 1 \ \forall\ i \in M,\ x_j \in (0,1)\ \forall\ j \in N\right\}.$$

Let $L \equiv K \cup T$, $\overline{L} \equiv N - L$, and define

$$w_L(z) = \max\left\{\sum_{j\in L} a_j x_j : \sum_{j\in K} x_j + \sum_{j\in T_-} \alpha_j x_j - \sum_{j\in T_+} \alpha_j(1 - x_j) \le z, \sum_{j\in N_i} x_j \le 1 \ \forall\ i \in M,\ x_j \in (0,1)\ \forall\ j \in N\right\}. \tag{14}$$

By Proposition 3.10, we have that

$$1+\alpha_t = \min\left\{ z : w_L(z) \geq b - \sum_{i\in(M_{\overline{L}}^- - p)} a_{j(i)} \right\},$$

and

$$1+\alpha_t-\alpha_s = \min\left\{ z : w_L(z) \geq b - a_s - \sum_{i\in(M_{\overline{L}}^- - p)} a_{j(i)} \right\}.$$

Hence, we can efficiently obtain the coefficients $\alpha_j$ for $j \in N_p$, by computing $w_L(z)$ efficiently for different pertinent values of $z$. Toward this end, consider the following recursive equation for computing the function $w_L(z)$. Note that

$$w_{L\cup N_p}(z) = \max\{ \max_{s\in(N_p-t)} [a_s + w_L(z+\alpha_t-\alpha_s)],\ [a_t+w_L(z)],\ [w_L(z+\alpha_t)]\}. \tag{15}$$

Hence, we can compute the coefficients $\alpha_j$ for $j \in \overline{K}$ by using the recursive equation (15). Consider the problem (14). Let $\hat{z}$ be the smallest among the alternative optimal solutions of $\max_z(w_L(z))$. Then, it follows that $w_L(z) = w_L(\hat{z})$ $\forall$ $z \geq \hat{z}$. For any minimal GUB cover $K$, to begin with, since $w_K(z) = w_K(|M_K|)$ $\forall$ $z \geq |M_K|$, we only need to compute $w_K(z)$ for $z = 1, \ldots, |M_K|$. Moreover, since $\alpha_t \geq 0$ and $\alpha_t - \alpha_s \geq 0$ $\forall$ $s \in (N_p - t)$, we have recursively that the value in (15), for each $L$ and $N_p$ thereafter, also remains a constant for $z \geq |M_K|$. Therefore, each function $w_L(z)$ needs to be evaluated (recursively) via (15) only for $z = 1, \ldots, |M_K|$. Hence, the time complexity of computing the lifted coefficients $\alpha_j$ for $j \in \overline{K}$ is $O(n|M_K|)$.

**4. Simultaneously lifted facets from minimal GUB covers.** We now consider an implementation of the reformulation–linearization technique (RLT) (see Sherali and Adams [11]) to characterize a class of valid inequalities (facets) of GUBKP, obtainable via a simultaneous lifting of minimal GUB cover inequalities. Toward this end, define a set $F$ corresponding to feasible solutions for GKP as

$$F \equiv \left\{ J \subseteq N : \sum_{j\in J} a_j \geq b,\ |J \cap N_i| \leq 1\ \forall\ i \in M \right\} \text{ and let } \overline{F} \equiv \{J \subseteq N : J \notin F\}.$$

Then, we can directly write conv$(X)$ as a convex combination of all feasible solutions to $X$ by associating, for each $J \subseteq N$, a convex combination weight $y_J$ with a vector that has ones in positions $j \in J$ and zeros otherwise. Noting that feasibly requires that $y_J \equiv 0$ if $J \in \overline{F}$, we get

$$\begin{aligned} GUBKP \equiv \text{conv}(X) = \Bigg\{ x : x_j = & \sum_{J:\ j\in J} y_J\ \forall\ j \in N, \\ & \sum_{J\subseteq N} y_J = 1, \\ & y_J \geq 0,\ \ \forall\ J \in F,\ y_J \equiv 0\ \ \forall\ J \in \overline{F} \Bigg\}. \end{aligned} \tag{16}$$

Using the standard projection operation, the set of all $x$'s for which there exist corresponding vectors $y$ that yield a feasible solution to (16) is given by duality or Farkas's Lemma (see Nemhauser and Wolsey, [9]) as

$$GUBKP \equiv \left\{ x : \sum_{j\in N} \pi_j^k x_j \geq \pi_0^k \right. \qquad \text{(17)} \qquad \left. \text{where } (\pi_j^k, \pi_0^k),\ k = 1,\dots,K, \text{are the extreme directions of } \Pi \right\}$$

where $\Pi \equiv \{(\pi, \pi_0) : \sum_{j\in J} \pi_j - \pi_0 \geq 0 \ \forall\ J \in F\}$.

We now consider the characterization of a family of valid inequalities of GUBKP obtainable via a simultaneous lifting of the minimal GUB cover inequality. Recall that the minimal GUB cover inequality $\sum_{j\in K} x_j \geq 1$ is a valid inequality for $\text{conv}(X(\overline{K}))$, and a facet for $\text{conv}(X(\overline{K}))$ if $\min_{j\in K}(a_j) + \sum_{i\in M_{\overline{K}}} a_{j(i)} \geq b$. We are interested in finding a lifted inequality, which is a facet of GUBKP, and is of the form $\sum_{j\in K} x_j + \sum_{j\in \overline{K}_-} \pi_j x_j - \sum_{j\in\overline{K}_+} \pi_j \bar{x}_j \geq 1$ where $\bar{x}_j \equiv (1 - x_j)\ \forall\ j \in N$, and $\pi_j\ \forall\ j \in \overline{K}$ are unrestricted in sign. This is of the form

$$\sum_{j\in K} x_j + \sum_{j\in\overline{K}_-} \pi_j x_j + \sum_{j\in\overline{K}_+} \pi_j x_j \geq 1 + \sum_{j\in\overline{K}_+} \pi_j. \tag{18}$$

Motivated by (17) and the form of (18), consider a polyhedral set $\Pi_{\overline{K}}$, where $\Pi_{\overline{K}} \equiv \{\pi_j,\ j \in \overline{K} : (\pi, \pi_0) \in \Pi,\ \pi_j = 1\ \forall\ j \in K,\ \pi_0 = 1 + \sum_{j\in\overline{K}_+} \pi_j\}$. Let us now proceed through a series of simplifications in characterizing $\Pi_{\overline{K}}$ more precisely, and then state our main result regarding this set and the validity of (18), in the same spirit as that of (17). To begin, observe that $\Pi_{\overline{K}}$ can be represented as follows, where $\pi_{\overline{K}} \equiv \{\pi_j : j \in \overline{K}\}$

$$\Pi_{\overline{K}} \equiv \left\{ \pi_{\overline{K}} : \sum_{j\in J\cap\overline{K}} \pi_j \geq 1 + \sum_{j\in\overline{K}_+} \pi_j - |J \cap K|,\ \forall\ J \in F \right\}. \tag{19}$$

Equivalently, we have,

$$\Pi_{\overline{K}} \equiv \left\{ \pi_{\overline{K}} : \sum_{j\in(\overline{K}_+\ -\ J)} \pi_j - \sum_{j\in J\cap\overline{K}_-} \pi_j \leq |J\cap K| - 1,\ \forall\ J \in F \right\}.$$

Note that we need to consider only those $J \in F$ above, for which $\overline{KJ} \equiv (\overline{K}_+\ -\ J) \cup (J \cap \overline{K}_-) \neq \phi$. Hence, we have that

$$\Pi_{\overline{K}} \equiv \left\{ \pi_{\overline{K}} : \sum_{j\in(\overline{K}_+\ -\ J)} \pi_j - \sum_{j\in J\cap\overline{K}_-} \pi_j \leq |J\cap K| - 1, \right. \qquad \left. \forall\ J \in F \text{ having } \overline{KJ} \neq \phi \right\}. \tag{20}$$

Furthermore, note that we need to examine only the most restrictive constraints in (20). Toward this end, define $GUB(\overline{K}) = \{T \subseteq \overline{K} : |T \cap N_i| \leq 1 \ \forall\, i \in M_{\overline{K}}\}$. Now for any $T \in GUB(\overline{K})$, define the feasible extension of $T$ as $J(T) = \{J \in F : J = J_1 \cup T$ for some $J_1 \subseteq K\}$, and let $\overline{KT} = (\overline{K}_+ - T) \cup (T \cap \overline{K}_-)$. Accordingly, define

$$F_K = \{T \in GUB(\overline{K}) : J(T) \neq \phi, \text{ and } \overline{KT} \neq \phi\}.$$

Then, we can restate (20) as follows.

$$(21)\quad \Pi_{\overline{K}} \equiv \left\{ \pi_{\overline{K}} : \sum_{j \in (\overline{K}_+ - T)} \pi_j - \sum_{j \in T \cap \overline{K}_-} \pi_j \leq \min\{|J \cap K| : J \in J(T)\} - 1 \quad \forall\, T \in F_K \right\}.$$

We now consider an explicit representation of the minimization problem in (21). Note that $T \in F_K$ if and only if (i) $|T \cap N_i| \leq 1$ for $i \in M_{\overline{K}}$, i.e., $T \in GUB(\overline{K})$, (ii) $\sum_{j \in K_+} a_j + \sum_{j \in T} a_j \geq b$, i.e., $J(T) \neq \phi$, and (iii) $\overline{KT} \neq \phi$. Hence, for each $T \in F_K$, we can represent the minimization problem in (21), denoted by $AGUBKP(T)$, as follows.

$$AGUBKP(T) : \text{minimize} \left\{ \sum_{j \in K_+} y_j : \sum_{j \in K_+} a_j y_j \geq b - \sum_{j \in T} a_j,\ y_j \in (0,1)\ \forall\, j \in K_+ \right\}.$$

Note that $1 \leq \nu(AGUBKP(T)) \leq |M_K|$, where $\nu(P)$ denotes the optimal objective value of the corresponding problem $P$. Of course, $AGUBKP$ is an easy problem in the sense that we can readily compute $\nu(AGUBKP(T))$ for each $T \in F_K$, using a greedy procedure. Let $\bar{b} = b - \sum_{j \in T} a_j$. If $\bar{b}$ is less than or equal to $\max_{j \in K_+}(a_j)$, then $\nu(AGUBKP(T)) = 1$. Otherwise, if $\bar{b}$ is less than or equal to the sum of the first two largest $a_j$'s for $j \in K_+$, then $\nu(AGUBKP(T)) = 2$, and so on. Hence the time complexity of solving AGUBKP is $O(|M| \log |M|)$. Let $\mathbf{N}(T) = \nu(AGUBKP(T)) - 1$. Then, we have

$$(22)\qquad \Pi_{\overline{K}} \equiv \left\{ \pi_{\overline{K}} : \sum_{j \in (\overline{K}_+ - T)} \pi_j - \sum_{j \in T \cap \overline{K}_-} \pi_j \leq \mathbf{N}(T) \ \ \forall\, T \in F_K \right\}.$$

PROPOSITION 4.1. *For a minimal GUB cover $K$, the inequality* (18) *is a valid inequality for GUBKP if and only if $\pi_{\overline{K}} \in \Pi_{\overline{K}}$, where $\pi_{\overline{K}} = \{\pi_j : j \in \overline{K}\}$, and $\Pi_{\overline{K}}$ is given by* (22).

*Proof.* $\pi x \geq \pi_0$ is valid for GUBKP if and only if $\sum_{j \in J} \pi_j \geq \pi_0 \ \ \forall\, J \in F$, that is, from (17), if and only if $(\pi, \pi_0) \in \Pi$. Hence, noting the form of (18) and the derivation of (22), we have that (18) is valid for GUBKP if and only if $\pi_{\overline{K}} \in \Pi_{\overline{K}}$, where $\Pi_{\overline{K}}$ is given by (22). This completes the proof. □

PROPOSITION 4.2. *Let $K$ be a minimal GUB cover such that $\min_{j \in K}(a_j) + \sum_{i \in M_{\overline{K}}} a_{j(i)} \geq b$, and let $\Pi_{\overline{K}}$ be given by* (22). *Then, the inequality* (18) *having $1 + \sum_{j \in \overline{K}_+} \pi_j > 0$ is a facet of GUBKP if and only if $(\pi_j,\ j \in \overline{K})$ is a vertex of $\Pi_{\overline{K}}$ with $1 + \sum_{j \in \overline{K}_+} \pi_j > 0$.*

*Proof.* As shown similarly in Sherali and Adams [11], it is readily verified that $\pi x \geq 1$ is a facet of GUBKP if and only if $\pi$ is an extreme point of $\Pi_1$, where

$$\Pi_1 = \left\{ \pi : \sum_{j \in J} \pi_j \geq 1 \; \forall \; J \in F \right\}. \tag{23}$$

Hence, (18) with $\pi_j = \overline{\pi}_j$ for $j \in \overline{K}$ is a facet of GUBKP if and only if the scaled partitioned vector $\hat{\pi}$, where

$$\hat{\pi} = \left\{ \left( \hat{\pi}_j = \frac{1}{(1 + \sum_{j \in \overline{K}_+} \overline{\pi}_j)}, \; j \in K \right), \; \left( \hat{\pi}_j = \frac{\overline{\pi}_j}{(1 + \sum_{j \in \overline{K}_+} \overline{\pi}_j)}, \; j \in \overline{K} \right) \right\} \tag{24}$$

is an extreme point of (23).

Now, for each $j \in K$, the set $J(j) \equiv \{j\} \cup \overline{K}_+ \in F$ by the hypothesis of the theorem, and the corresponding constraints of (23) are linearly independent and are binding at the solution (24). The latter $|K|$ linearly independent equality constraints appear as

$$\pi_j = 1 - \sum_{t \in \overline{K}_+} \pi_t \text{ for } j \in K, \tag{25}$$

and so determine $\pi_j$, $j \in K$, uniquely in terms of $\pi_j$, $j \in \overline{K}_+$. Now, (24) is a vertex of (23) if and only if it is feasible to (23) and there exist some $|\overline{K}|$ hyperplanes binding from (23) that are linearly independent in combination with (25). This happens if and only if $\{\hat{\pi}_j, \; j \in \overline{K}\}$ is an extreme point of the set obtained by imposing (25) on (23), i.e., the set

$$\left\{ \pi_{\overline{K}} : \sum_{j \in J \cap \overline{K}} \pi_j \geq 1 + |J \cap K| \left( \sum_{t \in \overline{K}_+} \pi_t - 1 \right) \; \forall \; J \in F \right\}. \tag{26}$$

This holds if and only if $(\hat{\pi}_j, \; j \in \overline{K})$ is feasible to (26), and there exist some $|\overline{K}|$ linearly independent hyperplanes that are binding at $(\hat{\pi}_j, \; j \in \overline{K})$. Feasibility of $\hat{\pi}$ to (26) requires from (24) that

$$\sum_{j \in J \cap \overline{K}} \frac{\overline{\pi}_j}{(1 + \sum_{t \in \overline{K}_+} \overline{\pi}_t)} \geq 1 + |J \cap K| \left\{ \frac{\sum_{t \in \overline{K}_+} \overline{\pi}_t}{(1 + \sum_{t \in \overline{K}_+} \overline{\pi}_t)} - 1 \right\} \; \forall \; J \in F.$$

That is, we must have

$$\sum_{j \in J \cap \overline{K}} \overline{\pi}_j \geq 1 + \sum_{t \in \overline{K}_+} \overline{\pi}_t - |J \cap K| \; \forall \; J \in F. \tag{27}$$

Note by (19) that (27) is equivalent to requiring that $\overline{\pi}_{\overline{K}}$ belongs to $\Pi_{\overline{K}}$. Moreover, an inequality in (26) is binding at $\hat{\pi}$ if and only if the corresponding inequality in (27) is binding. Also, a collection of $|\overline{K}|$ linearly independent equations from (26) give $\hat{\pi}$ as the unique solution if and only if the corresponding $|\overline{K}|$ equations from (27) give $\overline{\pi}$ as the unique solution, because from (24), there is a one-to-one correspondence between $\hat{\pi}$ and $\overline{\pi}$ according to

$$\left\{ \hat{\pi}_j = \frac{\overline{\pi}_j}{(1 + \sum_{j \in \overline{K}_+} \overline{\pi}_j)} \; \forall \; j \in \overline{K} \right\} \text{ and } \left\{ \overline{\pi}_j = \frac{\hat{\pi}_j}{(1 - \sum_{j \in \overline{K}_+} \hat{\pi}_j)} \; \forall \; j \in \overline{K} \right\}.$$

TABLE 1

| $T$ | $\overline{KT}$ | $\sum_{j\in T} a_j$ | $\nu(AGUBKP(T))$ | Inequalities of $\Pi_{\overline{K}}$ in (22) |
|---|---|---|---|---|
| $\phi$ | 9 | 0 | 2 | $\pi_9 \leq 1$ |
| 7 | 7, 9 | 1 | 2 | $\pi_9 - \pi_7 \leq 1$ |
| 8 | 8, 9 | 1 | 2 | $\pi_9 - \pi_8 \leq 1$ |

Hence, $\hat{\pi}_{\overline{K}}$ is an extreme point of (26) if and only if $\overline{\pi}_{\overline{K}}$ is an extreme point of $\Pi_{\overline{K}}$ with $1 + \sum_{j\in\overline{K}_+} \overline{\pi}_j > 0$, and this completes the proof. □

*Example* 4.1. Consider the following example to illustrate the above simultaneous lifting procedure. Let $X \equiv \{x \in (0,1)^9 : x_1 + x_2 + 2x_3 + x_4 + x_5 + 2x_6 + x_7 + x_8 + 3x_9 \geq 4,\ x_1 + x_2 + x_3 \leq 1,\ x_4 + x_5 + x_6 \leq 1,\ x_7 + x_8 + x_9 \leq 1\}$. A minimal GUB cover inequality is $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 1$, which is a facet of conv$(X(\overline{K}))$, where $\overline{K} = \{7, 8, 9\}$. For this minimal cover, we have that $F_K = \{\phi, \{7\}, \{8\}\}$, thereby leading to the computations shown in Table 1.

The point $(0, 0, 1)$ is the only vertex of $\Pi_{\overline{K}}$ with $1 + \sum_{j\in\overline{K}_+} \pi_j = 2 > 0$. Hence, $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_9 \geq 2$ is the only facet obtainable from the minimal GUB cover inequality by the lifting procedure.

*Remark* 4.1. In a spirit similar to Remark 2.1, the foregoing proposition can be used in the context of a separation problem for generating a simultaneously lifted facet of GUBKP, based on a given minimal GUB cover, that deletes a given fractional solution to the continuous relaxation of a 0-1 problem. In this context, an additional linear program would need to be solved over the polytope (22) to generate the cut coefficients for $j \in \overline{K}$, where $K$ is the minimal GUB cover being used. Note that if such a strategy is being used as in Crowder, Johnson, and Padberg [4] and Hoffman and Padberg [6] within an overall algorithm for solving a 0-1 integer program, then if this problem is sparse, we might expect $|\overline{K}|$ and $|M|$ to be manageably small when employing a GUB-constrained polytope based on a single problem constraint. Thus, the generation of the foregoing facetial cut would not be too computationally burdensome. In this type of an analysis, for further restricting a subset of the cut coefficients a priori before employing a reduced sized set (22) along with Propositions 4.1 and 4.2 to generate the remaining coefficients defining strong valid inequalities, it would be computationally useful to have knowledge of lower and upper bounds on the simultaneously lifted facet coefficients $\pi_j$ for $j \in \overline{K}$ in (18). This topic is addressed next.

**4.1 Lower and upper bounds on the lifted coefficient $\pi_j$ for $j \in \overline{K}$ in lifted inequality (18) under Proposition 4.2.** To begin with, let us consider the lifted coefficients $\pi_t$ for $t \in \overline{K}_+$. Since GUBKP is a full-dimensional polytope, it follows that for any $t \in \overline{K}_+$, we have $T = (\overline{K}_+ - t) \in F_K$, and from (22), we directly have that $\pi_t \leq \mathbf{N}(\overline{K}_+ - t)$. Hence, from (5), an upper bound $UB_t$ on $\pi_t$ is given by

$$UB_t = \mathbf{N}(\overline{K}_+ - t) = \eta^t - 1 \text{ for each } t \in \overline{K}_+.$$

Now, for given lower bounds $LB_s$ on $\pi_s\ \forall\ s \in (N_p - t)$, where $p \equiv M_t$, let us derive a lower bound $LB_t$ on $\pi_t$. Toward this end, examine any $T \in F_K$ such that $t \notin T$. Then, from (22), we have that

$$(28)\qquad \sum_{j\in(\overline{K}_+ - T)} \pi_j - \sum_{j\in T\cap\overline{K}_-} \pi_j \leq \mathbf{N}(T)\ \forall\ T \in F_k \text{ such that } t \notin T.$$

From (28), since $t \in (\overline{K}_+ - T)$, we have that

$$(29)\quad \pi_t \leq \mathbf{N}(T) - \left[ \sum_{j \in (\overline{K}_+ - T - t)} \pi_j - \sum_{j \in T \cap \overline{K}_-} \pi_j \right] \quad \forall\, T \in F_K \text{ such that } t \notin T.$$

Note that (29) is comprised of all the constraints of $\Pi_{\overline{K}}$ that contain $\pi_t$. Since at any vertex of $\Pi_{\overline{K}}$, at least one of (29) must be binding, we have that

$$(30)\quad \pi_t = \min \left\{ \mathbf{N}(T) - \left[ \sum_{j \in (\overline{K}_+ - T - t)} \pi_j - \sum_{j \in T \cap \overline{K}_-} \pi_j \right] : T \in F_K \text{ with } t \notin T \right\}.$$

For all $T \in F_k$ such that $t \notin T$, define

$$\nu(T) \equiv \sum_{j \in (\overline{K}_+ - T)} \pi_j - \sum_{j \in T \cap \overline{K}_-} \pi_j, \qquad \overline{\nu}(T) \equiv \sum_{j \in (\overline{K}_+ - T - t)} \pi_j - \sum_{j \in T \cap \overline{K}_-} \pi_j.$$

Then equation (30) reads

$$(31)\qquad \pi_t = \min\{\mathbf{N}(T) - \overline{\nu}(T) : T \in F_K \text{ with } t \notin T\}.$$

For any $T \in F_K$ such that $t \notin T$, if $(T + t) \in F_K$, then we have that $\overline{\nu}(T) = \nu(T + t) \leq \mathbf{N}(T + t)$ by (28). On the other hand, if $(T + t) \notin F_K$, then there exists some $s \in T \cap N_p$ and $\overline{\nu}(T) = \nu(T + t - s) - \pi_s \leq \mathbf{N}(T + t - s) - \pi_s$. Hence, from (31), we have

$$\pi_t = \min\{ \min_{T \in T_1} [\mathbf{N}(T) - \mathbf{N}(T + t)],\ \min_{T \in T_2} [\mathbf{N}(T) - \mathbf{N}(T + t - s) + \pi_s]\},$$

where $T_1 \equiv \{T \in F_K : t \notin T \text{ and } (T + t) \in F_K\}$, $T_2 \equiv \{T \in F_K : t \notin T$ and there exists some $s \in (N_p - t) \cap T$ where $p \equiv M_t\}$. Hence, for given lower bounds $LB_s$ on $\pi_s$ for $s \in (N_p - t)$, a lower bound $LB_t$ on $\pi_t$, for $t \in \overline{K}_+$, is given by

$$(32)\quad LB_t = \min\{ \min_{T \in T_1} [\mathbf{N}(T) - \mathbf{N}(T + t)],\ \min_{T \in T_2} [\mathbf{N}(T) - \mathbf{N}(T + t - s) + LB_s]\}.$$

Next, let us derive lower and upper bounds on the lifted coefficients $\pi_s$, for any $s \in \overline{K}_-$. From (22), we have that when $T = \{s\} \cup \{\overline{K}_+ - t\}$, where $t \equiv j(M_s)$, $\pi_t - \pi_s \leq \mathbf{N}(T)$. Hence, we have $\pi_s \geq \pi_t$ - $\mathbf{N}(T)$. Consequently, for a given lower bound $LB_t$ on $\pi_t$, $t \in \overline{K}_+$, a lower bound $LB_s$ for any $s \in \overline{K}_-$ is given by

$$(33)\qquad LB_s = LB_t - \mathbf{N}(\overline{K}_+ - t + s) \text{ where } t \equiv j(M_s).$$

*Remark* 4.2. Note that the lower bounds (32) and (33) are conditional bounds, each being determined based on lower bounds of the other. These conditional bounds are useful if we restrict the class of facets to have prescribed lower bounds on the $\pi_t$ or the $\pi_s$ coefficients. Otherwise, we need to derive unconditional lower bounds $LB_j$ $\forall\, j \in \overline{K}$. Toward this end, we derive an unconditional lower bound of zero on $\pi_t$, $\forall\, t \in \overline{K}_+$, as follows. By the transformation in Johnson and Padberg ([7], Prop. 2.1), since the inequality (18) is a facet of GUBKP and $|M_K| > 1$ (otherwise, dim(GUBKP) $< n$), it can be readily shown that the inequality

$$(34)\qquad \sum_{j \in K_+} z_j + \sum_{j \in \overline{K}_+} \pi_j z_j + \sum_{i \in M_{\overline{K}}} \sum_{j \in \{N_i - j(i)\}} (\pi_{j(i)} - \pi_j) z_j \leq |M_K| - 1$$

is a facet of the convex hull of the polytope

$$Z \equiv \operatorname{conv}\left\{ z \in (0,1)^n : \sum_{i \in M} \sum_{j \in N_i} \bar{a}_j z_j \leq \bar{b},\ \sum_{j \in N_i} z_j \leq 1\ \forall\ i \in M \right\},$$

where for all $i \in M$, $\bar{a}_{j(i)} = a_{j(i)}$, $\bar{a}_j = a_{j(i)} - a_j\ \forall\ j \in \{N_i - j(i)\}$, and $\bar{b} = b + \sum_{j \in N_+} a_j$. Since the constraints of the polytope $Z$ have all nonnegative coefficients, we have that all the coefficients of the facet (34) are nonnegative [5]. Accordingly, for each $t \in \overline{K}_+$, we have

(35) $$\pi_t \geq 0 \text{ and } \pi_t - \pi_s \geq 0\ \forall\ s \in (N_p - t), \text{ where } p = M_t.$$

From (35), we have a valid lower bound of zero on $\pi_t$, $\forall\ t \in \overline{K}_+$. Consequently, from (33), we have that $LB_s = -\mathbf{N}(\overline{K}_+ - t + s)\ \forall\ s \in (N_p - t)$, where $p \equiv M_t$.

Finally, let us derive an upper bound $UB_s$ for any given $s \in \overline{K}_-$. Note from (22) that the collection of constraints defining $\Pi_{\overline{K}}$ that contain the coefficient $\pi_s$ is given by

(36) $$\pi_s \geq \sum_{j \in (\overline{K}_+ - T)} \pi_j - \sum_{j \in (T-s) \cap \overline{K}_-} \pi_j - \mathbf{N}(T)\ \forall\ T \in F_K \text{ such that } s \in T.$$

Again, at any vertex of $\Pi_{\overline{K}}$, since at least one of (36) must be binding, we have that

(37) $$\pi_s = \max\left\{ \sum_{j \in (\overline{K}_+ - T)} \pi_j - \sum_{j \in (T-s) \cap \overline{K}_-} \pi_j - \mathbf{N}(T) : T \in F_K \text{ with } s \in T \right\}.$$

Now, in (37), if $(T - s) \in F_K$, we have from (22) that

$$\sum_{j \in (\overline{K}_+ - T)} \pi_j - \sum_{j \in (T-s) \cap \overline{K}_-} \pi_j \leq \mathbf{N}(T - s).$$

On the other hand, if $(T - s) \notin F_K$, then $\mathbf{N}(T - s) \equiv \infty$, and so the foregoing inequality holds for all $T \in F_K$ such that $s \in T$. Furthermore, for any $T \in F_K$ such that $s \in T$, we also have $(T - s + t) \in F_K$, where $t \equiv j(M_s)$. Consequently, from (22), the corresponding constraint for $T' \equiv (T - s + t) \in F_K$ yields

$$\sum_{j \in (\overline{K}_+ - T)} \pi_j - \sum_{j \in (T-s) \cap \overline{K}_-} \pi_j = \sum_{j \in (\overline{K}_+ - T')} \pi_j - \sum_{j \in T' \cap \overline{K}_-} \pi_j + \pi_t$$

(38) $$\leq \mathbf{N}(T') + \pi_t \leq \mathbf{N}(T') + UB_t.$$

Note that if $\overline{KT'} \equiv (\overline{K}_+ - T') \cup (T' \cap \overline{K}_-) = \phi$, we simply have $\mathbf{N}(T') \equiv 0$ in that case. Combining the last two inequalities, we may write for any $T \in F_K$ such that $s \in T$, and $t \equiv j(M_s)$,

$$\sum_{j \in (\overline{K}_+ - T)} \pi_j - \sum_{j \in (T-s) \cap \overline{K}_-} \pi_j \leq \min\{\mathbf{N}(T - s),\ \mathbf{N}(T - s + t) + UB_t\}.$$

Substituting this in (37) above, we obtain the following upper bound $UB_s$ on $\pi_s$ for any $s \in \overline{K}_-$.

(39) $$UB_s = \max_{T \in F_K:\ s \in T} \{\min\{\mathbf{N}(T - s),\ \mathbf{N}(T - s + t) + UB_t\} - \mathbf{N}(T)\},$$

where $t \equiv j(M_s)$.

*Example* 4.2. Consider the following constraints of a GUB-constrained knapsack problem, where $X \equiv \{x \in (0,1)^{12} : 2x_1 + 5x_2 + 2x_3 + 3x_4 + x_5 + 3x_6 + x_7 + 3x_8 + 2x_9 + 2x_{10} + 2x_{11} + 2x_{12} \geq 16,\ x_1 + x_2 \leq 1,\ x_3 + x_4 \leq 1,\ x_5 + x_6 \leq 1,\ x_7 + x_8 \leq 1,\ x_9 \leq 1,\ x_{10} \leq 1,\ x_{11} \leq 1,\ x_{12} \leq 1\}$. For a minimal GUB cover $K$={9, 10, 11, 12}, the minimal GUB cover inequality is $x_9 + x_{10} + x_{11} + x_{12} \geq 1$, which is a facet of conv$(X(\overline{K}))$ by Proposition 2.7. Note that $\overline{K}_+ = \{2,4,6,8\}$ and $\overline{K}_- = \{1,3,5,7\}$. We now consider a facet of the form (18) with $\pi_j \geq 0\ \forall\ j \in \overline{K}_-$. Let us derive lower and upper bounds on the coefficient $\pi_2$, where $\{2\} \in \overline{K}_+$. Since $\eta^2 = 4$, we have that $UB_2 = \eta^2 - 1 = 3$. Furthermore, the set $T_1 \equiv \{T : T \in F_K$ with $\{2\} \notin T$ and $(T + \{2\}) \in F_K\}$, is given by $T_1 = \{(3,6,8),\ (4,6,8)\}$. Also the set $T_2 \equiv \{T : T \in F_K$ with $\{2\} \notin T$ and $\{1\} \in T\}$ is given by $T_2 = \{(1,4,6),\ (1,4,8),\ (1,6,8),\ (1,3,5,8),\ (1,3,6,7),\ (1,3,6,8),\ (1,4,5,8),\ (2,4,6,7),\ (1,4,6,8)\}$. Hence, by (32), conditioned on $LB_1 = 0$, a lower bound $LB_2$ can be computed as follows.

$$LB_2 = \min\{\min_{T \in T_1}[\mathbf{N}(T) - \mathbf{N}(T + \{2\})],\ \min_{T \in T_2}[\mathbf{N}(T) - \mathbf{N}(T + \{2\} - \{1\}) + 0]\} = 1.$$

Next, let us select $\{1\} \in \overline{K}_-$, and illustrate the computation of an upper bound on the coefficient $\pi_1$ in any lifted facet (18). The set $T' \equiv \{T : T \in F_K$ with $\{1\} \in T\}$ is given by $\{(1,4,6),\ (1,4,8),\ (1,6,8),\ (1,3,5,8),\ (1,3,6,7),\ (1,3,6,8),\ (1,4,5,8),\ (1,4,6,7),\ (1,4,6,8)\}$. Since $UB_2 = 3$, we have by (39) that

$$UB_1 = \max_{T \in T'}\{\min\{\mathbf{N}(T - \{1\}),\ \mathbf{N}(T - \{1\} + \{2\}) + 3\} - \mathbf{N}(T)\} = 2.$$

Hence, we have that $0 \leq \pi_1 \leq 2$ and $1 \leq \pi_2 \leq 3$ in any lifted facet (18) having $\pi_1 \geq 0$. Note that we can also compute an unconditional lower bound $LB_1$, taking $LB_2 = 0$. By (33), $LB_1 = 0 - \mathbf{N}(\overline{K}_+ - \{2\} + \{1\}) = -2$. Hence, a set of unconditional bounds on $\pi_1$ and $\pi_2$ are given by $-2 \leq \pi_1 \leq 2$ and $0 \leq \pi_2 \leq 3$.

**5. A special case: The zero-one knapsack polytope.** Consider a special case of GUBKP with $|N_i| = 1\ \forall\ i \in M$, which represents the ordinary knapsack polytope, denoted by $KP$. That is, $KP \equiv$ conv$\{x \in (0,1)^n : \sum_{j \in N} a_j x_j \geq b\}$ where the data is all integer, $N = \{1, \ldots, n\}$, $b > 0$, $0 < a_j \leq b\ \forall\ j \in N$, and $\sum_{j \neq k} a_j \geq b$ for all $k \in N$. Recall that the minimal (GUB) cover inequality $\sum_{j \in K} x_j \geq 1$ is a facet of conv$(KP(\overline{K}))$, where $KP(\overline{K}) = KP \cap \{x \in (0,1)^n : x_j = 1,\ \forall\ j \in \overline{K}\}$. Our interest is in characterizing a (simultaneously) lifted facet, as in Balas and Zemel [2], of the form

$$\sum_{j \in K} x_j + \sum_{j \in \overline{K}} \pi_j x_j \geq 1 + \sum_{j \in \overline{K}} \pi_j, \tag{40}$$

where $K$ is a minimal (GUB) cover of $KP$. Toward this end, in the spirit of (21) and problem $AGUBKP(T)$, we define

$$f(\theta) = \min\left\{\sum_{j \in K} y_j : \sum_{j \in K} a_j y_j \geq \bar{b} + \theta\right\} - 1,$$

where $\bar{b} = b - \sum_{j \in \overline{K}} a_j$.

By Proposition 4.2, we have that (40) is a facet of KP if and only if $(\pi_j,\ j \in \overline{K})$ is an extreme point of $\Pi_{\overline{K}}$ with $1 + \sum_{j\in\overline{K}} \pi_j > 0$, where

$$\Pi_{\overline{K}} \equiv \left\{ \pi_{\overline{K}} : \sum_{j\in(\overline{K}-T)} \pi_j \le f\left( \sum_{j\in(\overline{K}-T)} a_j \right) \ \forall\, T \in F_K \right\} \tag{41}$$

and where $F_K = \{T \subset \overline{K} : \sum_{j\in K\cup T} a_j \ge b\}$.

This is precisely Balas and Zemel's characterization of simultaneously lifted facets obtainable from minimal cover inequalities. We now derive upper and lower bounds on $\pi_j,\ j \in \overline{K}$, for such facets of KP.

**5.1. Upper bound $UB_t$ on $\pi_t$, $t \in \overline{K}$.** From (41), by examining $T = \overline{K} - \{t\} \in F_K$, we directly have that $\pi_t \le f(a_t)$. Hence, an upper bound $UB_t$ on $\pi_t$ is given by

$$UB_t = f(a_t) \text{ for each } t \in \overline{K}.$$

Note that $UB_t$ is the same as Balas and Zemel's upper bound on $\pi_t,\ t \in \overline{K}$.

**5.2. Lower bound $LB_t$ on $\pi_t$, $t \in \overline{K}$.** Balas and Zemel [2] derive the following lower bound, denoted by $LBBZ_t$ :

$$LBBZ_t = h \qquad \forall\, t \in S_h,$$

where letting $E(K)$ denote the extension of $K$ as before, we have $S_0 = \overline{E(K)}$, and $S_h = \{t \in (E(K) - K) : \sum_{j\in K_h} a_j \le a_t < \sum_{j\in K_{h+1}} a_j\}$, where $K_h$ is the index set of the $h$ largest $a_j$ for $j \in K$. Note that $UB_t = LBBZ_t$ or $LBBZ_t + 1$.

We now construct a tighter lower bound on $\pi_t$. Consider any $t \in \overline{K}$, and let us examine any $T \in F_K$ such that $t \notin T$. From (41), we have that

$$\pi_t \le f\left( \sum_{j\in(\overline{K}-T)} a_j \right) - \sum_{j\in(\overline{K}-T-\{t\})} \pi_j \ \forall\, T \in F_K \text{ such that } t \notin T. \tag{42}$$

But since at an extreme point of $\Pi_{\overline{K}}$, at least one of (42) must be binding, we have that

$$\pi_t = \min \left\{ f\left( \sum_{j\in(\overline{K}-T)} a_j \right) - \sum_{j\in(\overline{K}-T-\{t\})} \pi_j : T \in F_K \text{ with } t \notin T \right\},$$

given other $\pi_j$ values. Consequently, we get

$$LB_t = \min \left\{ f\left( \sum_{j\in(\overline{K}-T)} a_j \right) - f\left( \sum_{j\in(\overline{K}-T-\{t\})} a_j \right) : T \in F_K \text{ with } t \notin T \right\}. \tag{43}$$

PROPOSITION 5.1. *$LB_t \ge LBBZ_t\ \forall\, t \in \overline{K}$.*

*Proof.* By the monotone increasing nature of the function $f$, it follows that $LB_t \ge 0\ \forall\, t \in \overline{K}$. Since $LBBZ_t = 0$ for $t \in \overline{E(K)}$, the result holds trivially for this case. Hence, suppose that $t \in (E(K) - K)$. Consider any $T \in F_K$ with $t \notin T$, and examine two cases.

*Case* i. $a_t = \sum_{j\in K_h} a_j$. It follows that $LBBZ_t = h \equiv f(a_t - \bar{b}) + 1$. But we have, defining $\hat{b} = b - \sum_{j\in T} a_j - a_t > 0$, that

$$\begin{aligned} f\left(\sum_{j\in(\overline{K}-T)} a_j\right) - f\left(\sum_{j\in(\overline{K}-T-\{t\})} a_j\right) &= \min\left\{\sum_{j\in K} y_j : \sum_{j\in K} a_j y_j \ge \hat{b} + a_t\right\} \\ &\quad - \min\left\{\sum_{j\in K} y_j : \sum_{j\in K} a_j y_j \ge \hat{b}\right\} \\ &\ge \min\left\{\sum_{j\in K} y_j : \sum_{j\in K} a_j y_j \ge a_t\right\} \\ &= f(a_t - \bar{b}) + 1. \end{aligned} \tag{44}$$

This implies from (43) that $LB_t \ge LBBZ_t$.

*Case* ii. $\sum_{j\in K_h} a_j < a_t < \sum_{j\in K_{h+1}} a_j$. In this case, we have, $LBBZ_t = h \equiv f(a_t - \bar{b})$. Let $\Delta$ be the amount that needs to be subtracted from $a_t$ so that $a_t - \Delta = \sum_{j\in K_h} a_j$. Then, using the monotone increasing nature of the function $f$ and following (44), we have that

(45)
$$\begin{aligned} f\left(\sum_{j\in(\overline{K}-T)} a_j\right) - f\left(\sum_{j\in(\overline{K}-T-\{t\})} a_j\right) &\ge f\left(\sum_{j\in(\overline{K}-T)} a_j - \Delta\right) f\left(\sum_{j\in(\overline{K}-T-\{t\})} a_j\right) \\ &\ge f(a_t - \Delta - \bar{b}) + 1 = f(a_t - \bar{b}). \end{aligned}$$

Hence from (43) and (45), we have that $LB_t \ge LBBZ_t$. Therefore, the result holds for any $t \in (E(K) - K)$ as well, and this completes the proof. □

*Example* 5.1. Consider $KP = \text{conv}\{x \in (0,1)^4 : 3x_1 + 3x_2 + 3x_3 + 2x_4 \ge 7\}$. Let $K = \{1,2\}$, so that $E(K) = \{1,2,3\}$ and $\bar{b} = 2$. Consider $t = \{4\} \in \overline{E(K)}$. Note that $\{T \in F_K : t \notin T\} = \{3\}$. Hence, from (43), we have that $LB_4 = f(a_4) - f(0) = f(a_4) = UB_4 = 1 > LBBZ_4 = 0$. Note, however, that $K$ is not a strong cover, as evidenced by the (strong) minimal cover $K' = \{3,4\}$. Otherwise, by the definition of a strong cover, we would have had for any $t \in \overline{E(K)}$, if it exists, that $UB_t = f(a_t) = 0$, and so $LB_t = 0 \equiv LBBZ_t$ as well.

*Example* 5.2. Consider $KP = \text{conv}\{x \in (0,1)^4 : 3x_1 + 3x_2 + 3x_3 + 4x_4 \ge 7\}$. The minimal cover $K = \{1,2,3\}$ is a strong cover for KP. Since $\bar{b} = 3$, $\overline{K} = \{4\}$, and for $t = \{4\} \in (E(K) - K)$, we have that $\{T \in F_K : t \notin T\} = \phi$; this gives $LB_4 = f(a_4) - f(0) = f(a_4) \equiv UB_4 = 2$. However, $LBBZ_t = 1 < LB_4$.

## REFERENCES

[1] E. Balas, *Facets of the knapsack polytope*, Math. Programming, 8 (1975), pp. 146-164.
[2] E. Balas and E. Zemel, *Facets of the knapsack polytope from minimal covers*, SIAM J. Appl. Math., 34 (1978), pp. 119-148.
[3] J.L. Balintify, G.T. Ross, P. Sinha, and A.A. Zoltners, *A mathematical programming system for preference and compatibility maximized menu planning and scheduling*, Math. Programming, 15 (1978), pp. 63-76.
[4] H. Crowder, E.L. Johnson, and M. Padberg, *Solving large-scale zero-one linear programming problems*, Oper. Res., 31 (1983), pp. 803-834.

[5] P.L. Hammer, E.L Johnson, and U.N. Peled, *Facets of regular* 0-1 *polytopes*, Math. Programming, 8 (175), pp. 179-206.

[6] K. Hoffman and M. Padberg, *Improving LP-representations of zero-one linear programs for branch-and-cut*, ORSA J. Comput., 3 (1991), pp. 121-134.

[7] E.L. Johnson and M.W. Padberg, *A note on the knapsack problem with special ordered sets*, Oper. Res. Lett., 1 (1981), pp. 18-22.

[8] R.K. Martin and L. Schrage, *Subset coefficient reduction cuts for* 0-1 *mixed integer programming*, Oper. Res., 33 (1985), pp. 505-526.

[9] G.L. Nemhauser and L.A. Wolsey, *Integer and combinatorial optimization*, John Wiley & Sons, New York, 1988.

[10] M.W. Padberg, *A note on zero-one programming*, Oper. Res., 23 (1975), pp. 833-837.

[11] H.D. Sherali and W.P. Adams, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM J. Discrete Math., 3 (1990), pp. 411-430.

[12] Y. Lee and H.D. Sherali, *Unrelated machine scheduling with time-window and machine-downtime constraints: An application to a naval battle-group problem*, Ann. Oper. Res., issue on Applications on Combinatorial Optimization, to appear.

[13] P. Sinha and A.A. Zoltners, *The multiple-choice knapsack problem*, Oper. Res., 27 (1979), pp. 503-515.

[14] L.A. Wolsey, *Valid inequalities for* 0-1 *knapsacks and MIP with generalized upper bound constraints*, Discrete Appl. Math., 29 (1990), pp. 251-261.

[15] E. Zemel, *Lifting the facets of zero-one polytopes*, Math. Programming, 15 (1978), pp. 268-277.

[16] ———, *Easily computable facets of the knapsack polytope*, Math. Oper. Res., 14 (1989), pp. 760-764.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 155–178, May 1995

# PRESERVING AND INCREASING LOCAL EDGE-CONNECTIVITY IN MIXED GRAPHS *

JØRGEN BANG-JENSEN[†], ANDRÁS FRANK[‡], AND BILL JACKSON[§]

**Abstract.** Generalizing and unifying earlier results of W. Mader, and A. Frank and B. Jackson, we prove two splitting theorems concerning mixed graphs. By invoking these theorems we obtain min-max formulae for the minimum number of new edges to be added to a mixed graph so that the resulting graph satisfies local edge-connectivity prescriptions. An extension of Edmonds's theorem on disjoint arborescences is also deduced along with a new sufficient condition for the solvability of the edge-disjoint paths problem in digraphs. The approach gives rise to strongly polynomial algorithms for the corresponding optimization problems.

**Key words.** mixed graphs, splitting theorems, edge-connectivity, augmentation, branchings, network synthesis

**AMS subject classifications.** 05C40, 68R10, 90B

## 1. Introduction and preliminaries.

Our main concern, the edge-connectivity augmentation problem, is as follows: given a mixed graph $M$, what is the minimum number (or, more generally, the minimum cost) $\gamma$ of new edges to be added to $M$ so that in the resulting graph $M'$, the local edge-connectivity $\lambda(x, y; M')$ between every pair of nodes $x, y$ is at least a prescribed value $r(x, y)$?

Several special cases were solved earlier for directed and undirected graphs. First, let $M$ be undirected. When $r \equiv 1$, the minimum cost augmentation problem reduces to a minimum cost tree problem. For $r \equiv 2$, the problem was solved independently by Eswaran and Tarjan [4] and Plesnik [22]. For this case, the minimum cost augmentation problem is already NP-complete.

The uniform case $r \equiv k$ for an arbitrary integer $k \geq 2$ was first solved by Watanabe and Nakamura [24], who developed a polynomial time algorithm as well as a min-max relationship. Slightly later, Cai and Sun [1] also solved this special case. The algorithm of Watanabe and Nakamura has been improved by Naor, Gusfield, and Martel [21]. Neither of these algorithms gives rise to a strongly polynomial time algorithm in the capacitated case. The first such approach was given by Frank [6]. The same paper includes a complete solution of the generalization to arbitrary (symmetric) demand functions $r(u, v)$.

For directed augmentation, the case $r \equiv 1$ was solved by Eswaran and Tarjan [4] while the general uniform case $r \equiv k (\geq 1)$ was solved by Frank [6]. Another interesting approach is by Gabow [9]. A related problem on augmentation was solved by Gusfield

* Received by the editors February 19, 1992; accepted for publication (in revised form) February 18, 1994.

[†] Department of Mathematics and Computer Science, Odense University, Denmark. The research of this author was partially supported by Danish Research Council grant 11-9635-1.

[‡] Department of Computer Science, Eötvös University, Budapest, Hungary, and Research Institute for Discrete Mathematics, University of Bonn, Germany. The research of this author was partially supported by Hungarian National Foundation for Scientific Research (OTKA) grant 2118 and Danish Research Council grant 11-9635-1.

[§] Department of Mathematical Studies, Goldsmiths' College, London, England. The research of this author was partially supported by the Department of Mathematics and Computer Science, Odense University.

[12], who described a way of adding a minimum number of directed or undirected edges to a mixed graph so that each edge belongs to a (possibly mixed) circuit with no backward directed edge. (Note, however, that our general mixed augmentation problem is not a generalization of Gusfield's.) Finally, several degree-constrained and node-cost variants were also solved by Frank [6].

On the negative side, for directed graphs the nonuniform demand problem was shown to be NP-complete by Frank [6] even if $r(u,v) \equiv 1$ for every pair of nodes $u, v$ of a specified subset $T \subset V$ and $r(u,v) \equiv 0$ otherwise. In this light, relatively little space is left for possible generalizations admitting good characterizations and/or polynomial time algorithms. (This sentence may serve as an excuse in case the reader feels that the hypothesis of the generalizations we discuss below is more technical than necessary.)

In the present paper we show how the augmentation problem for mixed graphs can be solved for certain demand functions that are more general than the uniform one. (By a *mixed graph* $M$ we mean a graph that may have both directed and undirected edges.) When the starting graph is mixed, one may wish to add both directed and undirected edges. Unfortunately, we do not have anything to say about this general case. Our results concern only the two extremes, when either only directed edges or undirected edges are allowed to be added to the given mixed graph $M$.

*Splitting off* a pair of edges $e = us, f = st$ means that we replace $e$ and $f$ by a new edge $ut$. The resulting mixed graph will be denoted by $M^{ef}$. This operation is defined only if both $e$ and $f$ are undirected (respectively, directed) and then the newly added edge $ut$ is considered undirected (directed). Accordingly, we speak of undirected or directed splittings.

Two theorems of W. Mader concerning directed and undirected splittings are important tools in the proofs by Frank in [6]. Here we follow an analogous line, and the basis for the present generalization is an extension of the existing splitting theorems. When a splitting-off operation is performed, the local edge-connectivity never increases. The content of the splitting-off theorems is that under certain conditions there is an appropriate pair $\{e = us, f = st\}$ of edges whose splitting preserves all local or global edge-connectivity between nodes distinct from $s$.

An interesting by-product of our investigations is an extension of Edmonds's theorem on the existence of $k$ disjoint arborescences [2]. A new sufficient condition will also be deduced for the existence of $k$ edge-disjoint paths in a directed graph connecting specified pairs of nodes.

Given two elements $s, t$ and a subset $X$ of a ground-set $U$, we say that $X$ is an *$s\bar{t}$-set* if $s \in X, t \notin X$. $X$ *separates* $s$ *from* $t$ (or $s$ and $t$) if $|X \cap \{s,t\}| = 1$. A family $\{X_1, \dots, X_t\}$ of pairwise disjoint, nonempty subsets of $U$ is called a *subpartition.*

Let $G = (U, E)$ be an undirected graph. $d_G(X,Y)$ denotes the number of undirected edges between $X - Y$ and $Y - X$. $\bar{d}_G(X,Y) := d_G(X, U-Y)(= d_G(U-X, Y))$. $d_G(X)$ stands for $d_G(X, U - X)$. Observe that $\bar{d}_G(X,Y) = \bar{d}_G(U - X, U - Y)$. When it will not cause ambiguity we shall leave out the subscript.

PROPOSITION 1.1. *For* $X, Y \subseteq U$,

$$d_G(X) + d_G(Y) = d_G(X \cap Y) + d_G(X \cup Y) + 2d_G(X,Y), \tag{1.1a}$$

$$d_G(X) + d_G(Y) = d_G(X - Y) + d_G(Y - X) + 2\bar{d}_G(X,Y). \tag{1.1b}$$

For a directed graph $D = (U, A), \rho_D(X)$ denotes the number of edges entering $X, \delta_D(X) := \rho_D(U - X)$, and $\beta_D(X) := \min(\rho_D(X), \delta_D(X))$. Note that $\beta_D(X) = \beta_D(U - X)$. $d_D(X,Y)$ denotes the number of edges with one end in $X - Y$ and one end in $Y - X$. $\bar{d}_D(X,Y) := d_D(X, U - Y)(= d_D(U - X, Y)$. An *out-arborescence* $F$ is a directed tree in which every node but one has in-degree 1 and the exceptional node, called the *root,* is of in-degree 0. (Equivalently, there is a directed path from the root to every other node of $F$.)

PROPOSITION 1.2. *For* $X, Y \subseteq U$,

$$\text{(1.2a)} \qquad \rho_D(X) + \rho_D(Y) = \rho_D(X \cap Y) + \rho_D(X \cup Y) + d_D(X,Y).$$

*If* $\delta_D(X \cap Y) = \rho_D(X \cap Y)$, *then*

$$\text{(1.2b)} \qquad \rho_D(X) + \rho_D(Y) = \rho_D(X - Y) + \rho_D(Y - X) + \bar{d}_D(X,Y).$$

*If* $\delta_D(X \cup Y) = \rho_D(X \cup Y)$, *then*

$$\text{(1.2c)} \qquad \rho_D(X) + \rho_D(Y) = \delta_D(X - Y) + \delta_D(Y - X) + \bar{d}_D(X,Y).$$

*If* $\delta_D(X \cap Y) = \rho_D(X \cap Y)$ *or* $\delta_D(X \cup Y) = \rho_D(X \cup Y)$, *then*

$$\text{(1.2d)} \qquad \beta_D(X) + \beta_D(Y) \geq \beta_D(X - Y) + \beta_D(Y - X) + \bar{d}_D(X,Y).$$

*Proof.* Equation (1.2a) follows by showing that each edge has the same contribution to the two sides. A similar argument shows that $\rho_D(X) + \rho_D(Y) = \rho_D(X - Y) + \rho_D(Y - X) + \bar{d}_D(X,Y) + (\rho_D(X \cap Y) - \delta_D(X \cap Y))$ holds for any digraph $D$ from which (1.2b) follows. The derivation of (1.2c) is analogous.

Let us prove (1.2d). The two cases are clearly equivalent: Substitute $U - X$ for $X$ and $U - Y$ for $Y$. So assume that $\delta_D(X \cap Y) = \rho_D(X \cap Y)$. If $\beta_D(X) = \rho_D(X)$ and $\beta_D(Y) = \rho_D(Y)$ then by (1.2b), $\beta_D(X) + \beta_D(Y) = \rho_D(X) + \rho_D(Y) = \rho_D(X - Y) + \rho_D(Y - X) + \bar{d}_D(X,Y) \geq \beta_D(X - Y) + \beta_D(Y - X) + \bar{d}_D(X,Y)$. The case when $\beta_D(X) = \delta_D(X)$ and $\beta_D(Y) = \delta_D(X)$ is analogous. Finally, suppose that $\beta_D(X) = \rho_D(X)$ and $\beta_D(Y) = \delta_D(Y)$. Let $Y' := U - Y$. Then, applying (1.2a) to $X$ and $Y'$ we get $\beta_D(X) + \beta_D(Y) = \rho_D(X) + \rho_D(Y') = \rho_D(X \cap Y') + \rho_D(X \cup Y') + d_D(X,Y') = \rho_D(X - Y) + \delta_D(Y - X) + \bar{d}_D(X,Y) \geq \beta_D(X - Y) + \beta_D(Y - X) + \bar{d}_D(X,Y)$. □

Let $M = (U, A \cup E)$ be a mixed graph composed as the union of a directed graph $D = (U, A)$ and an undirected graph $G = (U, E)$. Let $\rho_M(X) := \rho_D(X) + d_G(X), \delta_M(X) := \delta_D(X) + d_G(X)$, and $\beta_M(X) := \min(\rho_M(X), \delta_M(X))$. We say that a node $v$ of a $M$ is *di-Eulerian* if $\rho_D(v) = \delta_D(v)$. $M$ is called di-Eulerian if every node of $M$ is di-Eulerian.

By combining Propositions 1.1 and 1.2, we obtain the following proposition.

PROPOSITION 1.3. *For* $X, Y \subseteq U$,

$$\text{(1.3a)} \qquad \rho_M(X) + \rho_M(Y) = \rho_M(X \cap Y) + \rho_M(X \cup Y) + d_D(X,Y) + 2d_G(X,Y).$$

*If* $\delta_D(X \cap Y) = \rho_D(X \cap Y)$, *then*

$$\text{(1.3b)} \qquad \rho_M(X) + \rho_M(Y) = \rho_M(X - Y) + \rho_M(Y - X) + \bar{d}_D(X,Y) + 2\bar{d}_G(X,Y).$$

*If* $\delta_D(X \cup Y) = \rho_D(X \cup Y)$, *then*

$$\text{(1.3c)} \qquad \rho_M(X) + \rho_M(Y) = \delta_M(X - Y) + \delta_M(Y - X) + \bar{d}_D(X,Y) + 2\bar{d}_G(X,Y).$$

*If* $\delta_D(X \cap Y) = \rho_D(X \cap Y)$ *or* $\delta_D(X \cup Y) = \rho_D(X \cup Y)$, *then*

$$\text{(1.3d)} \qquad \beta_M(X) + \beta_M(Y) \geq \beta_M(X - Y) + \beta_M(Y - X) + \bar{d}_D(X,Y) + 2\bar{d}_G(X,Y).$$

By a *feasible path* (or simply *path*) of a mixed graph $M$ we mean a sequence $\{v_0, v_0v_1, v_1, v_1v_2, v_2, \dots, v_{n-1}, v_{n-1}v_n, v_n\}$, where each $v_iv_{i+1}$ is a directed or undirected edge of $M$. The *local edge-connectivity* $\lambda(s,t;M) = \lambda(s,t)$ from $s$ to $t$ is in the maximum number of edge-disjoint paths from $s$ to $t$. By a version of Menger's theorem, this is equal to the minimum of $\delta_D(S) + d_G(S)$ over all $s\bar{t}$-sets $S$. Note that $\lambda(s,t)$ can be computed by a max-flow min-cut (MFMC) computation.

Let $U$ be a set and $r(x,y)(x,y \in U)$, an arbitrary symmetric, nonnegative function. Define a set function $R$ as follows. Let $R(\emptyset) = R(U) = 0$, and for $X \subset U$ let

$$R(X) := \max(r(x,y) : x,y \in U, X \text{ separates } x \text{ and } y). \tag{1.4}$$

Clearly, $R(X) = R(U - X)$.

LEMMA 1.1. *For arbitrary $X, Y \subseteq U$, at least one of the following two inequalities holds:*

$$R(X) + R(Y) \leq R(X \cap Y) + R(X \cup Y), \tag{1.5a}$$

$$R(X) + R(Y) \leq R(X - Y) + R(Y - X). \tag{1.5b}$$

*Proof.* First observe that if $Y$ is replaced by $U - Y$, then (1.5a) and (1.5b) transform into each other. Let $(z,z')$ be a pair that maximizes $r(z,z')$ over all pairs which are separated by at least one of the sets $X$ and $Y$. By symmetry we may assume that $z \in X$ and $z' \in U - X$. By replacing $Y$ by $U - Y$, if necessary, we may also assume that $z \notin Y$.

If $z' \in Y$, then $r(z,z') = R(X) = R(Y) = R(X-Y) = R(Y-X)$, and hence (1.5b) holds (actually with equality). If $z' \notin Y$, then $r(z,z') = R(X) = R(X \cup Y) = R(X-Y)$. Clearly, $R(Y) \leq R(X \cap Y)$ or $R(Y) \leq R(Y - X)$. Accordingly, (1.5a) or (1.5b) holds. □

Let $M = (U, A \cup E)$ be a mixed graph with a specified node $s$ satisfying $\rho_M(s) = \delta_M(s)$. Throughout this paper we will use the notation $V := U - s$. Let

$$T(M) := \{x \in V : \rho_M(x) \neq \delta_M(x)\} \tag{1.6}$$

be the set of non-di-Eulerian nodes. Observe that $\rho_M(T(M)) = \delta_M(T(M))$ and hence $T(M)$ never consists of one element. Let $k$ be a positive integer and assume that

$$\lambda(x,y;M) \geq k \quad \text{for every } x,y \in T(M). \tag{1.7}$$

Suppose that $r(x,y)$ satisfies

$$r(x,y) \leq k \quad \text{for every } x,y \in U, \quad \text{and} \tag{1.8a}$$

$$r(x,y) = k \quad \text{for every } x,y \in T(M). \tag{1.8b}$$

For $X \subseteq U$ define $q(X) := R(X) - \beta_M(X)$.

LEMMA 1.2. *For $X, Y \subseteq U$, at least one of the following two inequalities holds:*

$$q(X) + q(Y) \leq q(X \cap Y) + q(X \cup Y) - (2d_G(X,Y) + d_D(X,Y)), \tag{1.9a}$$

$$q(X) + q(Y) \leq q(X - Y) + q(Y - X) - (2\bar{d}_G(X,Y) + \bar{d}_D(X,Y)). \tag{1.9b}$$

*Proof.* Since $q(X) = q(U - X)$, the inequalities in (1.9) transform into each other when $X$ is replaced by its complement $U - X$. Therefore, we can assume that $\beta_M(X) = \rho_M(X) \leq \delta_M(X)$ and $\beta_M(Y) = \rho_M(Y) \leq \delta_M(Y)$.

If $R$ satisfies (1.5a), then by subtracting (1.3a) from (1.5a) we obtain (1.9a). Now assume that $R$ does not satisfy (1.5a). Then at least one of $T(M) \cap X \cap Y$ and $T(M) - (X \cup Y)$ is empty, otherwise (1.4) and (1.8) would imply that $R(X) = R(Y) = R(X \cap Y) = R(X \cup Y) = k$, and hence (1.5a) would hold. Therefore (1.3d) holds. Furthermore, by Lemma 1.1, $R$ satisfies (1.5b). Subtracting (1.3d) from (1.5b) we obtain (1.9b). $\square$

For $x, y \in U$, let us define

(1.10a) $$r_M(x,y) := \min(k, \lambda(x,y;M)) \qquad \text{if } x, y \in V,$$

(1.10b) $$r_M(x,y) := 0 \quad \text{if } s \in \{x,y\}.$$

Note that $r_M$ depends on $M, s$, and $k$ and satisfies (1.8).

LEMMA 1.3. $r_M(x,y) = r_M(y,x)$.

*Proof.* This clearly holds if $\lambda(x,y;M) = \lambda(y,x;M)$ and, by (1.10b), $s \in \{x,y\}$. Assume that $x, y \in V$ and $\lambda(x,y;M) < \lambda(y,x;M)$. There is an $x\bar{y}$-set $X$ for which $\delta_M(X) = \lambda(x,y;M)$. We cannot have $\lambda(x,y;M) < k$ since at least one of the sets $X$ and $U - X$, say $X$, is then disjoint from $T(M)$. But then $\rho_M(X) = \delta_M(X)$ and hence $\lambda(x,y;M) = \delta_M(X) = \rho_M(X) \geq \lambda(y,x;M)$, a contradiction. Therefore, $k \leq \lambda(x,y;M) < \lambda(y,x;M)$, that is, $r_M(x,y) = k = r_M(y,x)$ as required. $\square$

Define

(1.11) $$R_M(X) := \max(r_M(x,y) : X \text{ separates } x \text{ and } y).$$

Note that by (1.10) $R_M(X) = R_M(V - X)$ for every $X \subseteq V$.

LEMMA 1.4. *For any subset $X \subseteq V$ separating nodes $x, y \in V$,*

(1.12a) $$\beta_M(X) \geq R_M(X) \geq r_M(x,y).$$

*Moreover, if $\lambda(x,y;M) \leq k$, then there is a subset $X_0$ of $V$ separating $x$ and $y$ for which*

(1.12b) $$\beta_M(X_0) = r_M(x,y).$$

*Proof.* By symmetry we may assume that $x \in X$ and $y \in V - X$. Then $r_M(x,y) \leq \lambda(x,y;M) \leq \delta_M(X)$ and $r_M(y,x) \leq \lambda(y,x;M) \leq \rho_M(X)$. From this and Lemma 1.3 we get $\beta_M(X) \geq r_M(x,y)$. This, in turn, along with the definition of $R_M(X)$, implies (1.12a).

If $\lambda(x,y;M) \leq k$, then $\lambda(x,y;M) = r_M(x,y)$ and, by Menger's theorem, there is a subset $X_0 \subset V$ separating $x$ and $y$ for which $\beta_M(X_0) = \lambda(x,y;M)$ and (1.12b) follows. $\square$

Let $s_M(X) := \beta_M(X) - R_M(X)$. By Lemma 1.4, $s_M(X) \geq 0$ for every $X \subseteq V$. We call a nonempty set $X \subseteq V$ *tight* (*dangerous*) if $s_M(X) = 0\,(s_M(X) \leq 1)$. We may distinguish between the two possible types of tight (dangerous) sets by use of the prefix "in" or "out." Note that $V$ is not tight if $\rho_M(s), \delta_M(s) > 0$. Lemma 1.2 immediately provides the following lemma.

LEMMA 1.5. *For $X, Y \subseteq V$, one of the following inequalities holds:*

(1.13a) $$s_M(X) + s_M(Y) \geq s_M(X \cap Y) + s_M(X \cup Y) + 2d_G(X,Y) + d_D(X,Y),$$

(1.13b) $$s_M(X) + s_M(Y) \geq s_M(X - Y) + s_M(Y - X) + 2\bar{d}_G(X,Y) + \bar{d}_D(X,Y).$$

We are going to prove two splitting theorems for $M$. In §2 each edge incident to $s$ is directed, while in §3 the edges incident to $s$ are all undirected.

## 2. Directed splitting.

When a splitting operation is carried out, the local edge-connectivity may drop. There are theorems for directed graphs stating that global or local edge-connectivities may be preserved by an appropriate choice of edges to be split off. One is from Mader [20].

THEOREM 2.1. *Let $D = (V + s, A)$ be a directed graph for which $\lambda(x, y; D) \geq k$ for every $x, y \in V$ and $\rho(s) = \delta(s)$. Then, for every edge $f = st$ there is an edge $e = us$ such that $\lambda(x, y; D^{ef}) \geq k$ for every $x, y \in V$.*

The next theorem was proven by Frank [5] and Jackson [13].

THEOREM 2.2. *Let $D = (V + s, A)$ be a directed Eulerian graph, that is, $\rho(x) = \delta(x)$ for every node $x$ of $D$. Then, for every edge $f = st$ there is an edge $e = us$ such that $\lambda(x, y; D^{ef}) = \lambda(x, y; D)$ for every $x, y \in V$.*

Our first result is a common generalization of these two theorems. (Recall the definition of function $r_M(x, y)$ in (1.10).)

THEOREM 2.3. *Let $M = (V + s, A \cup E)$ be a mixed graph, satisfying* (1.7). *Assume that $s$ is incident only with directed edges and $\rho_M(s) = \delta_M(s) > 0$. Then, for every edge $f = st$, there is an edge $e = us$ such that*

$$\lambda(x, y; M^{ef}) \geq r_M(x, y) \quad \textit{for every } x, y \in V. \tag{2.1}$$

If $M = D$ is a directed graph and $\lambda(x, y; D) \geq k$ for every $x, y \in V$, then $r_M(x, y) = k$ and we are back at Theorem 2.1. If $M = D$ is directed Eulerian graph and $k := \max(\lambda(x, y; D) : x, y \in V)$, then we are back at Theorem 2.2.

We call a pair $\{e, f\}$ satisfying (2.1) *splittable*. This is equivalent to requiring that

$$r_{M^{ef}}(x, y) \geq r_M(x, y) \quad \text{for every } x, y \in V. \tag{2.2}$$

Repeatedly applying Theorem 2.3 $\rho_M(s)$ times, one obtains the following theorem.

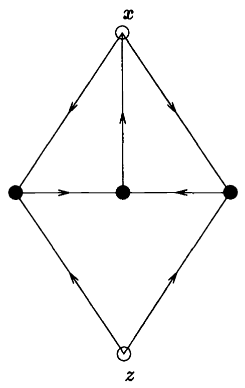
THEOREM 2.4. *Let $M = (V + s, A \cup E)$ be a mixed graph satisfying* (1.7). *Assume that $s$ is incident only with directed edges and $\rho_M(s) = \delta_M(s)$. Then the edges entering and leaving $s$ can be matched into $\rho_M(s)$ disjoint pairs so that $\lambda(x, y; M^+) \geq r_M(x, y)$ for every $x, y \in V$, where $M^+$ denotes the mixed graph arising from $M$ by splitting off all these pairs.*

*Proof of Theorem* 2.3. We may assume that every edge of $M$ is directed since replacing each undirected edge with a pair of oppositely directed edges does not affect the local edge-connectivities. (Incidentally, this means that having a mixed graph in Theorem 2.3 rather than a directed one is not a big thing; the point is that Theorems 2.1 and 2.2 can be combined into one.) Note that for edges $e = us, f = st$, one has $\beta_{M^{ef}}(X) = \beta_M(X) - 1$ if $u, t \in X$ and $\beta_{M^{ef}}(X) = \beta_M(X)$ otherwise.

CLAIM 2.1. *A pair $\{e = us, f = st\}$ is splittable in $M$ if and only if there is no tight set $X$ containing $u$ and $t$.*

*Proof.* First suppose that $X$ is a tight set containing $u$ and $t$. Then $\beta_{M^{ef}}(X) + 1 = \beta_M(X) = R_M(X)$. There are nodes $x \in X, y \in V - X$ such that $R_M(X) = r_M(x, y)$. By applying (1.12) to $M^{ef}$ we obtain $r_{M^{ef}}(x, y) \leq \beta_{M^{ef}}(X) < \beta_M(X) = R_M(X) = r_M(x, y)$, that is, $\{e, f\}$ is not splittable.

Conversely, suppose that $\{e, f\}$ is not splittable. Then there are nodes $x, y \in V$ such that $\lambda(x, y; M^{ef}) < r_M(x, y) \leq k$. Then $r_{M^{ef}}(x, y) = \lambda(x, y; M^{ef})$, and by applying Lemma 1.4 to $M^{ef}$ we see that there is a set $X \subset V$ separating $x$ and $y$ for which



$T = \bullet$

FIG. 1. *Here $k = 2, T$ consists of three nodes, and there are no two edge-disjoint out-arborescences rooted at $z$ with both containing every element of $T$. Observe that $\rho(x) = 1 < \delta(x) = 2$ holds for the only node $x$ not in $T$.*

$\beta_{M^{ef}}(X) = r_{M^{ef}}(x,y)$. Using (1.12a) we have $\beta_M(X) - 1 \leq \beta_{M^{ef}}(X) = r_{M^{ef}}(x,y) \leq r_M(x,y) - 1 \leq R_M(X) - 1 \leq \beta_M(X) - 1$. Hence equality follows everywhere. In particular, $\beta_M(X) = R_M(X)$, that is, $X$ is tight. Also, $\beta_M(X) - 1 = \beta_{M^{ef}}(X)$, that is, $u, t \in X$. $\square$

CLAIM 2.2. *There are no two maximal tight $t\bar{s}$-sets.*

*Proof.* Assume, indirectly, that $X$ and $Y$ are such sets. Apply Lemma 1.5. If (1.13a) holds, then we have $0 + 0 = s_M(X) + s_M(Y) \geq s_M(X \cap Y) + s_M(X \cup Y) + d_D(X,Y) \geq 0$, from which $s_M(X \cup Y) = 0$, contradicting the maximality of $X$ and $Y$.

If (1.13b) holds, then we have $0 + 0 = s_M(X) + s_M(Y) \geq s_M(X - Y) + s_M(Y - X) + \bar{d}_D(X,Y) \geq 0 + 0 + \bar{d}_D(X,Y)$. Hence $\bar{d}_D(X,Y) = 0$, which contradicts the existence of the edge $st$. $\square$

If there is no tight $t\bar{s}$-set, then choose an arbitrary edge $e = us$ of $M$. If there are tight $t\bar{s}$-sets, then by Claim 2.2 there is a unique maximal one denoted by $X$. We claim that there is an edge $e = us$ with $u \notin X$. Assume this is not the case. Then the existence of the edge $st$ and the fact that $\rho_M(s) = \delta_M(s)$ imply that $\delta_M(V-X) < \rho(X)$ and $\rho_M(V - X) < \delta(X)$, that is, $\beta_M(V - X) < \beta_M(X)$. This is impossible, however, since $\beta_M(V - X) \geq R_M(V - X) = R_M(X) = \beta_M(X)$. By Claim 2.1 the pair $\{us, st\}$ is splittable. $\square$

Mader [20] showed how his Theorem 2.1 implies the following basic result of Edmonds on edge-disjoint arborescences.

THEOREM 2.5 [2], [3]. *In a digraph $D = (U, A)$ with a special node $z$ there are $k$ edge-disjoint spanning out-arborescences of root $z$ if and only if $\rho(X) \geq k$ holds for each subset $X \subseteq U - z$ of nodes (or, equivalently, there are $k$ edge-disjoint paths from $z$ to every other node of $D$.)*

The following possible generalization naturally emerges. In addition to $z$, we are given a subset $T \subseteq U - z$ so that $\rho(X) \geq k$ for every subset $X \subseteq U - z, X \cap T \neq \varnothing$. Is it true that there are $k$ edge-disjoint out-arborescences rooted at $z$ so that each contains every element of $T$? The answer is yes if $T = U - z$ (by Edmonds's theorem) or if $|T| = 1$ (by Menger's theorem). But Lovász [15] found the example in Fig. 1 which shows that such a statement is not true in general. In this light, the following result might have some value.

THEOREM 2.6. *Let $D = (U, A)$ be a digraph with a special node $z$ called a root,* *and let $T' := \{x \in U - z : \rho(x) < \delta(x)\}$. Assume that $\lambda(z, x; D) \geq k (\geq 1)$ for every $x \in T'$. Then there is a family $\mathcal{F}$ of $k$ edge-disjoint out-arborescences rooted at $z$ so that every node $x \in U$ belongs to at least $r(x) := \min(k, \lambda(z, x; D))$ members of $\mathcal{F}$.*

*Proof.* The theorem is trivial if $|U| = 2$, so suppose that $|U| \geq 3$. We may assume that there is no edge in $D$ entering $z$. Now $U - T' - z$ is nonempty; otherwise $\rho(x) < \delta(x)$ would hold for every node of $D$, which is not possible since $\sum \rho(x) = |A| = \sum \delta(x)$.

Let $s \in U - T' - z$ be a node for which $r(s)$ is minimum. By the hypothesis made on $T', r(s) \leq r(x)$ for every $x \in U$. Extend $D$ by adding $\rho(x) - \delta(x)$ parallel edges from $x$ to $z$ for each $x \in U - T' - z$, and $k$ parallel edges from $x$ to $z$ for each $x \in T'$. Let $D'$ denote the resulting digraph.

Clearly, $T(D') \subseteq T' + z$. We claim that (1.7) holds for $D'$. This is equivalent to saying that $\rho_{D'}(X) \geq k$, and $\delta_{D'}(X) \geq k$ holds for every subset $X \subseteq V - z$ for which $X \cap T'$ is nonempty. The first inequality follows from the hypothesis. The second one follows from the fact that $\delta_{D'}(X) = \rho_{D'}(X) - \sum(\rho_{D'}(x) - \delta_{D'}(x) : x \in X) \geq \rho_{D'}(X) \geq k$.

We can apply Theorem 2.3, which implies that there are edges $e = us, f = st$ such that $\lambda(z, x; D_1) \geq r(x)$ holds for every $x \in U - s$, where $D_1$ denotes the digraph arising from $D'$ by splitting off $e$ and $f$. It is also clear that $\lambda(z, s; D_1) \geq r(s) - 1$.

By induction there is a family $\mathcal{F} = \{F_1, \ldots, F_k\}$ of $k$ edge-disjoint out-arborescences in $D_1$ rooted at $z$ such that each node $x$ belongs to at least $r(x)$ members of $\mathcal{F}$ for $x \in U - s$, and $s$ belongs to at least $r(s) - 1$ members of $\mathcal{F}$. Let $a = ut$ denote the edge of $D_1$ which results from the splitting of $f = st$ and $e = us$.

Suppose first that one member of $\mathcal{F}$, say $F_1$, contains $a$. If (i) $s$ is not contained in $F_1$, define $\bar{F}_1 := F_1 - a + e + f$. If (ii) $s$ is contained in $F_1$, let $P$ denote the unique subpath of $F_1$ from $z$ to $s$ with its last edge $h = ws$. If $P$ does not use $a$, define $\bar{F}_1 := F_1 - a + f$. If $P$ uses $a$, define $\bar{F}_1 := F_1 - a - h + e + f$. Finally, if no member of $\mathcal{F}$ contains $a$, define $\bar{F}_1 := F_1$.

By these constructions $\bar{F}_1$ is an out-arborescence of $D$ containing each node belonging to $F_1$ plus, possibly, node $s$. Hence we have a family $\bar{\mathcal{F}} = \{\bar{F}_1, F_2, \ldots, F_k\}$ of $k$ out-arborescences of $D$ so that each node $x$ other than $s$ belongs to at least $r(x)$ of them, and $s$ belongs to at least $r(s) - 1$ of them. If $s$ belongs to at least $r(s)$ members of $\bar{\mathcal{F}}$, then this family satisfies the requirements of the theorem. If (i) occurred, then we are surely in this case.

Suppose $s$ is contained in precisely $r(s) - 1$ members of $\bar{\mathcal{F}}$. Then (ii) occurs and by the choice of $s, r(x) \geq r(s)$ for every $x \in U$. Hence every node $x$ is in strictly more members of $\bar{\mathcal{F}}$ than $s$ is. Therefore, there is a member $F$ of $\mathcal{F}$ containing $x$ but not $s$. By the construction of $\bar{\mathcal{F}}$, at least one of the edges $e = us$ and $h = ws$ is not used by the out-arborescences of $\bar{\mathcal{F}}$. Accordingly, choose $x$ to be $u$ or $w$. We conclude that by replacing the out-arborescence $F$ by $F + xs$ in $\bar{\mathcal{F}}$, we obtain a family of $k$ out-arborescences satisfying the requirements. $\square$

Clearly, if in Theorem 2.6, $\lambda(z, x; D) \geq k$ holds for every $x \in U$, then we are back at Edmonds's theorem. Another special case may also be worth mentioning. Call a digraph $D = (U, A)$ with root $z$ a *preflow digraph* if $\rho(x) \geq \delta(x)$ holds for every $x \in U - z$. (The name arises from an MFMC algorithm of Karzanov [14] and Goldberg and Tarjan [11], where a preflow was defined as a function on the edge-set of a digraph so that the in-sum is at least the out-sum at every node except the root.) An easy, well-known fact from network flow theory is that any flow from the source to the terminal may be decomposed into path-flows. The following corollary may be considered as a generalization.

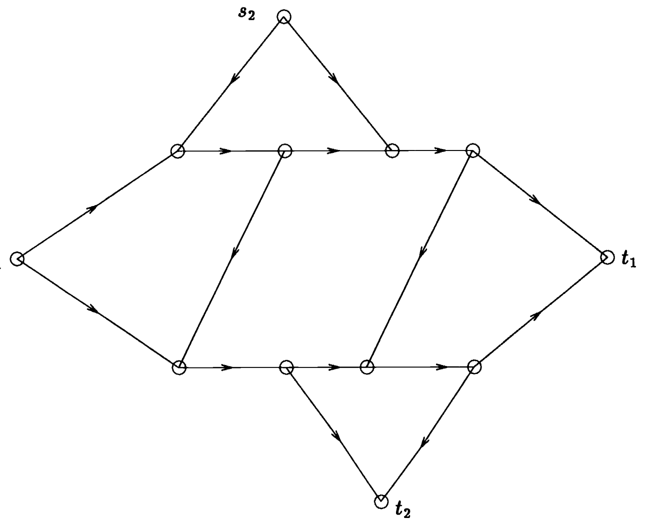


FIG. 2. *Here $k = 2$ but the example can easily be generalized to arbitrary $k$. In fact, we can have arbitrarily many paths between $s_i$ and $t_j$ $(i, j = 1, 2, \dots, k)$ and still not have the edge-disjoint paths.*

COROLLARY 2.1. *In a preflow digraph $D = (U, A)$, for any integer $k(\geq 1)$ there is a family $\mathcal{F}$ of $k$ edge-disjoint out-arborescences of root $z$ such that every node $x$ belongs to $\min(k, \lambda(z, x; D))$ members of $\mathcal{F}$. In particular, if $k := \max(\lambda(z, x; D) : x \in U - z)$, then every $x$ belongs to $\lambda(z, x; D)$ members of $\mathcal{F}$.*

Y. Shiloach [23] pointed out that Edmonds's theorem immediately implies the following pretty result. Given $k$ pairs $(s_1, t_1), \dots, (s_k, t_k)$ of nodes in a $k$ edge-connected digraph $D$, there are edge-disjoint paths from $s_i$ to $t_i$ $(i = 1, \dots, k)$.

Using Theorem 2.6 we have the following generalization.

COROLLARY 2.2. *Let $(s_1, t_1), \dots, (s_k, t_k)$ be $k$ pairs of nodes in a digraph $D = (U, A)$ such that for every node $x$ with $\rho(x) < \delta(x)$ or $x = t_j$ there are edge-disjoint paths from $s_i$ to $x$ $(i = 1, \dots, k)$. Then there are edge-disjoint paths from $s_i$ to $t_i$ $(i = 1, \dots, k)$.*

*Proof.* Extend the digraph by a new node $z$ and an edge $zs_i$ for each $i = 1, \dots, k$. By Theorem 2.6 there are $k$ edge-disjoint out-arborescences rooted at $z$ such that each contains every $t_i$. Since there are $k$ edges leaving $z$, each edge $zs_i$ belongs to one of these out-arborescences denoted by $F_i$. Now $F_i$ includes a path $P_i$ from $s_i$ to $t_i$ $(i = 1, \dots, k)$, and these paths satisfy the requirements. □

Note that if we only impose the condition in Corollary 2.2 on the vertices $t_i, i = 1, 2, \dots, k$, then $D$ may not have edge-disjoint paths from $s_i$ to $t_i$ $(i = 1, 2, \dots, k)$. This can be seen from the example in Fig. 2.

## 3. Undirected splitting.

Generalizing earlier results of Lovász [16], [17] (see also [18]), Mader proved the following powerful theorem on undirected splitting. For a short proof, see Frank [7]. In what follows $U = V + s$ will denote the node set of the graph in question. We will use the terms $R_M, r_M, \delta_M, \rho_M$, and $\beta_M$ introduced in §1.

THEOREM 3.1 [19]. *Let $G = (V + s, E)$ be a (connected) undirected graph in which $0 < d_G(s) \neq 3$ and there is no cut-edge incident to $s$. Then there exists a pair of edges $e = su, f = st$ such that $\lambda(x, y; G) = \lambda(x, y; G^{ef})$ holds for every $x, y \in V$.*

The main result of this section is an extension of Mader's theorem to mixed graphs. Let $M = (V+s, A \cup E)$ be a mixed graph composed from a digraph $D = (V+s, A)$ and an undirected graph $G = (V+s, E)$ so that $s$ is incident only with undirected edges. By a *cut-edge* of a mixed graph $M$ we mean an edge $e \in E$ such that $M - e$ is disconnected (in the undirected sense).

THEOREM 3.2. *Suppose that in $M = (V+s, A \cup E)$, node $s$ is incident only with undirected edges, $0 < d_M(s) \neq 3$, and*

(3.1) *there is no cut-edge incident to $s$.*

*Let $k \geq 2$ be an integer satisfying* (1.7). *Then there is a pair of edges $e = su, f = st$ such that*

(3.2a) $$\lambda(x, y; M^{ef}) \geq r_M(x, y) \quad \textit{for every } x, y \in V.$$

We call a pair $\{e, f\}$ satisfying (3.2a) *splittable.* Inequality (3.2a) is equivalent to the following:

(3.2b) $$r_{M^{ef}}(x, y) \geq r_M(x, y) \quad \text{for every } x, y \in V.$$

In order to make repeated splittings, the following lemma is useful.

LEMMA 3.1. *If $\{e, f\}$ is splittable in a mixed graph $M$ satisfying the hypothesis of Theorem* 3.2, *then $M^{ef}$ satisfies* (3.1).

*Proof.* Let $x$ and $y$ be two neighbours of $s$. We claim that $\lambda(x, y; M) \geq 2$. Indeed, $\lambda(x, y; M) \geq 1$ since $xs, ys \in E$ by the assumption. If $\lambda(x, y; M) = 1$, then there is an $x\bar{y}$-set $X$ with $\delta_D(X) = 0$ and $d_G(X) = 1$. Let $h$ denote the unique edge of $G$ between $X$ and $V + s - X$. Then $h$ is either $xs$ or $ys$. Since $k \geq 2$, one of the sets $X$ and $V + s - X$ is disjoint from $T(M)$. By (1.7), $\rho_D(X) = \delta_D(X)(= 0)$, showing that $h$ is a cut-edge incident to $s$. Thus $\lambda(x, y; M) \geq 2$. By (3.2a), $\lambda(x, y; M^{ef}) \geq 2$ for every two neighbours $x, y$ of $s$. This implies the claim. □

A closely related form of Theorem 3.2 is as follows.

THEOREM 3.3. *Let $M = (V+s, A \cup E)$ be a mixed graph with a node $s$ such that $s$ is incident only with undirected edges, $d(s)$ is even, and* (3.1) *holds. Let $k \geq 2$ be an integer satisfying* (1.7). *Then the set of edges incident to $s$ can be matched into $d(s)/2$ disjoint pairs so that $\lambda(x, y; M^+) \geq r_M(x, y)$ for every $x, y \in V$, where $M^+$ denotes the mixed graph arising from $M$ by splitting off all these pairs.*

This theorem is analogous to Theorem 2.4 except that Theorem 3.3 does not hold for $k = 1$ (see Fig. 3).

CLAIM 3.1. *Theorems* 3.2 *and* 3.3 *are equivalent.*

*Proof.* First assume the truth of Theorem 3.2 and let $\{e, f\}$ be a splittable pair. By Lemma 3.1, Theorem 3.2 can be applied $d_G(s)/2$ times. Theorem 3.3 follows by observing that a pair splittable in $M^{ef}$ is splittable in $M$ as well.

Conversely, assume that Theorem 3.3 is true. If $d_G(s)$ is even, then there is nothing to prove, so assume that $d_G(s)$ is odd. Then $d_G(s) \geq 5$. Let $M'$ denote a mixed graph arising from $M$ by adding a new node $x$ and three parallel undirected edges between $x$ and $s$. Property (3.1) holds for $M'$ and hence Theorem 3.3 applies to $M'$. Since $d_G(s) \geq 5$, among the $(d_G(s) + 3)/2$ splittable pairs in $M'$ provided by Theorem 3.3, at least one pair $\{e, f\}$ must consist of original edges. Clearly, $\{e, f\}$ is splittable in $M$, as well. □

*Proof of Theorem* 3.3. By Lemma 3.1 it suffices to prove that there is one splittable pair. We may suppose that every undirected edge $h$ of $M$ is incident to $s$,

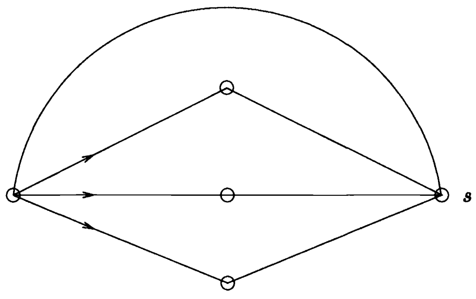


FIG. 3. *There is no way to match the edges at s so that the mixed graph remains strongly connected.*

otherwise we could replace $h$ by two oppositely directed edges. Let $M$ denote a counterexample in which every edge not incident to $s$ is directed and the total number of nodes and edges is minimum. It is clear that $M$ is connected (in the undirected sense). Note that for $X \subseteq V$ and edges $e = su, f = st$ one has $\beta_{M^{ef}(X)} = \beta_M(X) - 2$ if $u, t \in X$ and $\beta_{M^{ef}}(X) = \beta_M(X)$ otherwise. □

CLAIM 3.2. *A pair $\{e, f\}$ of edges $e = su, f = st$ is splittable if and only if there is no dangerous set $X$ containing $t$ and $u$.*

*Proof.* First suppose that $X$ is a dangerous set containing $u$ and $t$. Then $\beta_{M^{ef}}(X) + 2 = \beta_M(X) \leq R_M(X) + 1$. There are nodes $x \in X, y \in V - X$ such that $R_M(X) = r_M(x, y)$. By applying Lemma 1.4 to $M^{ef}$, we obtain $r_{M^{ef}}(x, y) \leq \beta_{M^{ef}}(X) = \beta_M(X) - 2 \leq R_M(X) - 1 = r_M(x, y) - 1$, that is, $\{e, f\}$ is not splittable.

Conversely, suppose that $\{e, f\}$ is not splittable. Then there are nodes $x, y \in V$ such that $\lambda(x, y; M^{ef}) < r_M(x, y) \leq k$. Then $r_{M^{ef}}(x, y) = \lambda(x, y; M^{ef})$, and by applying Lemma 1.4 to $M^{ef}$ we see that there is a set $X \subset V$ separating $x$ and $y$, for which $\beta_{M^{ef}}(X) = r_{M^{ef}}(x, y)$. Using (1.12a) we have $\beta_M(X) - 2 \leq \beta_{M^{ef}}(X) = r_{M^{ef}}(x, y) \leq r_M(x, y) - 1 \leq R_M(X) - 1 \leq \beta_M(X) - 1$. Hence $\beta_M(X) \leq R_M(X) + 1$, that is, $X$ is dangerous. Furthermore, $\beta_M(X) > \beta_{M^{ef}}(X)$, that is, $u, t \in X$. □

CLAIM 3.3. *Let $X \subseteq V$ be a tight set. A pair $\{e = su, f = st\}$ of edges is splittable in $M$ if the corresponding pair $\{e', f'\}$ is splittable in the contracted mixed graph $M' := M/X$.*

*Proof.* For a subset $Z$ of nodes of $M$ for which either $Z \subseteq V - X$ or $X \subseteq Z$, let $Z'$ denote the subset of nodes of $M'$ corresponding to $Z$. For such a $Z$, clearly $R_{M'}(Z') \geq R_M(Z)$ and $\rho_{M'}(Z') = \rho_M(Z)$. Therefore, if $Z$ is dangerous in $M$, then $Z'$ is dangerous in $M'$. This fact and Claim 3.2 imply that

(*) there is no dangerous set $Z$ in $M$ containing $u$ and $t$ such that $X \subseteq Z$ or $Z \subseteq V - X$.

By claim 3.2, if $\{e, f\}$ is not splittable in $M$, there is a dangerous set $Y$ containing $u$ and $t$. If (1.13a) holds for $X$ and $Y$, then $0 + 1 \geq s_M(X) + s_M(Y) \geq s_M(X \cap Y) + s_M(X \cup Y) \geq s_M(X \cup Y)$, that is, $Z := X \cup Y$ is dangerous, contradicting (*).

If (1.13b) holds, then $0 + 1 \geq s_M(X) + s_M(Y) \geq s_M(X - Y) + s_M(Y - X) + 2\bar{d}_G(X, Y) \geq s_M(Y - X) + 2\bar{d}_G(X, Y)$. Hence $s_M(Y - X) \leq 1$ and $\bar{d}_G(X, Y) = 0$, in particular, $u, t \notin X \cap Y$. That is, $Z := Y - X$ is dangerous and contains $u$ and $t$, contradicting (*) □

We call a tight set $X$ *trivial* if $|X| = 1$. Since $M$ is a minimal counterexample, Claim 3.3 shows that

(3.3) in $M$ every tight set is trivial.

CLAIM 3.4. *For every $u, v \in V$,*

$$r_M(u,v) = \min(\beta_M(u), \beta_M(v), k). \tag{3.4}$$

*Proof.* By (1.12a), $\min(\beta_M(u), \beta_M(v)) \geq r_M(u,v)$, so if $\lambda(u,v;M) \geq k$, then (3.4) follows. If $\lambda(u,v;M) < k$, then by Lemma 1.4 there is a set $X \subset V$ separating $u$ and $v$, such that $\beta_M(X) = R_M(X) = r_M(u,v)$. That is, $X$ is tight and by (3.3), $|X| = 1$, from which the claim follows. □

CLAIM 3.5. *$V - T(M)$ as nonempty.*

*Proof.* Let $Y := \{v \in V : \rho_D(v) > \delta_D(v)\}, X := \{v \in V : \rho_D(v) < \delta_D(v)\}$. If the claim is false, then $X$ and $Y$ form a partition of $V$. By the definition of $X$ and $Y$ there is a directed edge $h := xy$ from $X$ to $Y$.

Let $M' := M - h$. Then $T(M') \subseteq T(M)$ and, since $k \geq 2$, (3.1) is valid for $M'$. Since $\rho_M(y) > \delta_M(y) \geq k$, the set $\{y\}$ is not in-tight. Similarly, $\{x\}$ is not out-tight. Using this fact and (3.3) we get that $h$ does not enter any in-tight set. Therefore, $\lambda(u,v;M) = \lambda(u,v;M')$ for every $u,v \in V$. Since $T(M') \subseteq T(M)$, (1.7) holds for $M'$. Because of the minimality of $M, M'$ is not a counterexample and there is a pair $\{e,f\}$ of edges splittable in $M'$. Hence $\{e,f\}$ is splittable in $M$ as well, a contradiction. □

Let $t_0 \in V - T(M)$ be a node for which

$$\beta_M(t_0) \text{ is minimum.} \tag{3.5}$$

We distinguish between two cases.

*Case* 1. $d_G(s,t_0) \geq 1$.

In order to be consistent with the notation in earlier claims, for Case 1 let us rename $t_0$ by $t$. That is, $d_G(s,t) \geq 1$.

CLAIM 3.6. *For every $t\bar{s}$-set $X (X \neq \{t\})$,*

$$R_M(X - t) \geq R_M(X). \tag{3.6}$$

*Proof.* There is a pair of nodes $x, y$ such that $x \in X, y \in V - X$, and $R_M(X) = r_M(x,y)$. If $x \neq t$, then $R_M(X-t) \geq r_M(x,y) = R_M(X)$ and (3.6) follows. Assume that $x = t$ and let $u \in X - t$ be an arbitrary node. By (3.4) we have $R_M(X) = r_M(t,y) = \min(\beta_M(t), \beta_M(y), k)$ and $R_M(X-t) \geq r_M(u,t) = \min(\beta_M(u), \beta_M(t), k)$. Hence (3.6) follows if $\beta_M(u) \geq \beta_M(t)$ or if $\beta_M(u) \geq k$. So assume that $\beta_M(u) < \beta_M(t)$ and $\beta_M(u) < k$. By (3.5), $u$ must be in $T(M)$. Since $T(M)$ never consists of a single node, (1.7) implies that $\beta_M(u) \geq k$, a contradiction. □

CLAIM 3.7. *If $X \subseteq V$ is dangerous, then $d_G(s,X) \leq d_G(s, V-X)$.*

*Proof.* Let $\alpha := d_G(s,X)$ and $\beta := d_G(s, V-X)$, and assume that $X$ is, say, in-dangerous. We have $R_M(V-X) = R_M(X+s) = R_M(X) \geq \rho_M(X) - 1 = \delta_M(V-X) - \beta + \alpha - 1 \geq R_M(V-X) - \beta + \alpha - 1$, from which $\alpha \leq \beta + 1$ follows. However, we cannot have equality, otherwise $d_G(s) = 2\beta + 1$ would follow, contradicting the hypothesis of the theorem that $d_G(s)$ is even. □

Let $\mathcal{S}$ denote the set of neighbours of $s$. Since no pair $\{su, st\}$ is splittable in $M$, Claim 3.2 implies that every element of $S$ belongs to a dangerous $t\bar{s}$-set. Let $\mathcal{L}$ be a minimal family of such dangerous sets so that $\cup(X : X \in \mathcal{L}) \supseteq S$. By Claim 3.7, $|\mathcal{L}| \geq 2$. We may assume that the members of $\mathcal{L}$ are maximal dangerous $t\bar{s}$-sets.

CLAIM 3.8. $|\mathcal{L}| \geq 3$.

*Proof.* By Claim 3.7, $|\mathcal{L}| \geq 2$. Assume that $\mathcal{L}$ has just two members, $X$ and $Y$. Since $S \subseteq X \cup Y$, by Claim 3.7 we have $d_G(s,X) \leq d_G(s, V-X) < d_G(s,Y) \leq$

$d_G(s, V - Y) < d_G(s, X)$, a contradiction. Here the two strict inequalities hold since $S \subseteq X \cup Y$ and $t \in S \cap X \cap Y$. □

CLAIM 3.9. *For every two members $X, Y$ of $\mathcal{L}$, $|X-Y| = |Y-X| = 1, \bar{d}_D(X,Y) = 0$, and $\bar{d}_G(X,Y) = 1$.*

*Proof.* If (1.13b) holds for $X$ and $Y$, then $1 + 1 \geq s_M(X) + s_M(Y) \geq s_M(X - Y) + s_M(Y - X) + 2\bar{d}_G(X,Y) \geq 0 + 0 + 2$, and hence $\bar{d}_G(X,Y) = 1$ and both $X - Y$ and $Y - X$ are tight. By (3.3) the claim follows.

If (1.13b) does not hold, then (1.13a) does. Since $X$ and $Y$ are maximal dangerous sets, $s_M(X \cup Y) \geq 2$, and hence $1+1 \geq s_M(X)+s_M(Y) \geq s_M(X \cap Y)+s_M(X \cup Y) \geq 0 + 2$. Therefore, $s_M(X \cap Y) = 0$ and by (3.3) $|X \cap Y| = 1$, that is, $X \cap Y = \{t\}$. By Claim 3.6, $R_M(X - t) \geq R_M(X)$ and $R_M(Y - t) \geq R_M(Y)$. Therefore, $R_M(X) + R_M(Y) \leq R_M(X - t) + R_M(Y - t) = R_M(X - Y) + R_M(Y - X)$. That is, (1.5b) holds for $R_M$.

Since $X \cap Y = \{t\}$, (1.3d) holds and, therefore, (1.13b) holds, a contradiction. □

Let $X_1, X_2, X_3$ be three members of $\mathcal{L}$ and $Z := X_1 \cap X_2 \cap X_3$. By the minimality of $\mathcal{L}$, each $X_i$ has an element $x_i$ not in any other member of $\mathcal{L}$. By Claim 3.9 it follows that $X_i = Z + x_i$ $(i = 1,2,3)$ and $\bar{d}_D(X_i, X_j) = 0, \bar{d}_G(X_i, X_j) = 1$ for $(1 \leq i < j \leq 3)$. Hence only one edge leaves or enters $Z$, namely, the edge $st$. That is, $st$ is a cut-edge, contradicting (3.1). This contradiction shows that Case 1 cannot occur.

*Case* 2. $d_G(s, t_0) = 0$.

Since $M$ is connected, $\rho_M(t_0 = \delta_M(t_0) > 0$. Let $at_0$ and $t_0b$ be arbitrary edges in $M$ so that, if possible, $a \neq b$. Note that $a = b$ only if $t_0$ has just one neighbour in $M$. Let $M'$ denote the mixed graph arising from $M$ by splitting off $at_0$ and $t_0b$.

Clearly, $T(M) = T(M')$ and $\lambda(t_0, v; M') \geq \lambda(t_0, v; M) - 1$ holds for every $v \in V - t_0$. Moreover, we claim that $\lambda(u, v; M') = \lambda(u, v; M)$ for every $u, v \in V - t_0$. This is straightforward if $a = b$, and follows from (3.3) and Claim 2.1 if $a \neq b$. Therefore, we have

(3.7a) $$r_{M'}(t_0, v) \geq r_M(t_0, v) - 1 \quad \text{for every } v \in V - t_0,$$

(3.7b) $$r_{M'}(u, v) = r_M(u, v) \quad \text{for every } u, v \in V - t_0.$$

CLAIM 3.10. *$M'$ satisfies* (3.1).

*Proof.* If (3.1) is not true for $M'$, there is a set $C \subseteq V$ separating $t_0$ from $a$ and $b$ with $\rho_M(C) = \delta_M(C) = 2$ such that there is just one undirected edge $h = sz$ entering $C$ and one of the edges $at_0$ and $t_0b$ enters $C$ while the other one leaves $C$.

Let $h' = su$ be another undirected edge of $M$. Then $u \neq z$ and we claim that $\lambda(u, z; M) \geq 2$, otherwise there is a $z\bar{u}$-set $X$ with $\rho_M(X) = 1$. Since $d_G(X) \geq 1, \rho_D(X) = 0$. Since $k \geq 2, X$ cannot separate any two members of $T(M)$ and hence $\delta_D(X) = \rho_D(X) = 0$. But then $h$ or $h'$ is a cut-edge violating (3.1).

$\lambda(u, z; M) \geq 2$ and $\rho_M(C) = 2$ imply that $C$ is tight. By (3.3), $|C| = 1$, that is, $C = \{z\} = \{t_0\}$. The existence of edge $h$ contradicts the assumption that $d_G(s, t_0) = 0$. □

By the minimality of $M, M'$ is not a counterexample of Theorem 3.3. Since (3.1) holds for $M'$, there is a pair $\{e := su, f := st\}$ of undirected edges splittable in $M'$. Since we are at Case 2, $t \neq t_0$.

CLAIM 3.11. *$\{e, f\}$ is splittable in $M$.*

*Proof.* By claim 3.2, if the pair $\{e, f\}$ is not splittable in $M$, then there is a dangerous set $X \subseteq V$ containing $u$ and $t$. By (3.7), $R_{M'}(X) \geq R_M(X) - 1$. Using the fact that $X$ is not dangerous in $M'$, that is, $\beta_{M'}(X) > R_{M'}(X) + 2$, we obtain

$$(3.8) \qquad \beta_M(X) \geq \beta_{M'}(X) \geq R_{M'}(X) + 2 \geq R_M(X) + 1 \geq \beta_M(X).$$

Hence equality follows everywhere, in particular, $R_{M'}(X) = R_M(X) - 1$. This and (3.7) imply that $R_M(X) = r_M(t_0, y)$ for some $y \in V$, which is separated from $t_0$ by $X$, and

$$(3.9) \qquad r_M(t_0, y) > r_M(x, y)$$

for any $x \in V$, which is separated from $y$ by $X$.

If $t_0 \in X$, then choose $x := t$, an element different from $t_0$. If $t_0 \in V - X$, then there must be an element $x \in V - X - t_0$; otherwise, $V - X = \{t_0\}$, from which $\beta_M(X) \geq \beta_M(t_0) + 2$ follows. Using (3.4) we have $R_M(X) = r_M(t_0, y) = \min(\beta_M(y), \beta_M(t_0), k) \leq \beta_M(t_0) \leq \beta_M(X) - 2$, contradicting the hypothesis that $X$ is dangerous.

In both cases, from (3.9) and (3.4) we have $\min(\beta_M(t_0), \beta_M(y), k) = r_M(t_0, y) > r_M(x, y) = \min(\beta_M(x), \beta_M(y), k)$, which implies $\beta_M(x) < \beta_M(t_0)$ and $\beta_M(x) < k$. The first inequality shows, by (3.5), that $x \in T(M)$, while the second one implies, by (1.7), that $x \notin T(M)$, a contradiction. □

Claim 3.11 contradicts the fact that $M$ is a counterexample. Thus Case 2 is also impossible and the proof of Theorem 3.3 is complete. □

We mention two special cases. In the first, $r_M(x, y) \equiv k$ is assumed, while the second concerns mixed graphs with all di-Eulerian nodes.

COROLLARY 3.1. *Suppose that in a mixed graph $M = (V + s, A \cup E)$, node $s$ is incident only with undirected edges, $0 < d(s) \neq 3$, and there is no cut-edge incident to $s$. Let $k \geq 2$ be an integer such that $\lambda(x, y; M) \geq k$ for every $x, y \in V$. Then there is a pair of edges $e = su, f = st$ such that $\lambda(x, y; M^{ef}) \geq k$ for every $x, y \in V$.*

COROLLARY 3.2. *Suppose that in a mixed graph $M = (V + s, A \cup E)$, node $s$ is incident only with undirected edges, $0 < d(s) \neq 3$, there is no cut-edge incident to $s$, and $\rho_M(v) = \delta_M(v)$ for every node $v \in V$. Then there is a pair of edges $e = su, f = st$ such that $\lambda(x, y; M^{ef}) = \lambda(x, y; M)$ for every $x, y \in V$.*

Note that this corollary is already a generalization of Mader's Theorem 3.1.

We close this section by pointing out that for a mixed graph $M = (V + s, A \cup E)$, one cannot always split away a pair of edges incident to $s$ in such a way that for every pair of vertices $x, y \in V \min(\lambda(x, y; M), \lambda(y, x; M))$ is preserved. Such an example is given in Fig. 4.

## 4. Increasing edge-connectivity.

This section is offered to exhibit two new edge-connectivity augmentation results according to whether only directed or undirected edges are allowed to be added. We will formulate the results for mixed starting graphs, but these forms are clearly equivalent to the cases when the starting graph is a directed graph. Therefore, our first theorem is basically a directed augmentation theorem in which both the starting graph and the new edges to be added are directed, while in the second theorem the starting graph is directed and the new edges are undirected.

In both theorems we have the same requirement for the demand function $r$, namely, $r(x, y)$ is symmetric, not larger than a specified positive integer $k$, and precisely $k$ for pairs of non-di-Eulerian nodes $x, y$. An interesting phenomenon in the case

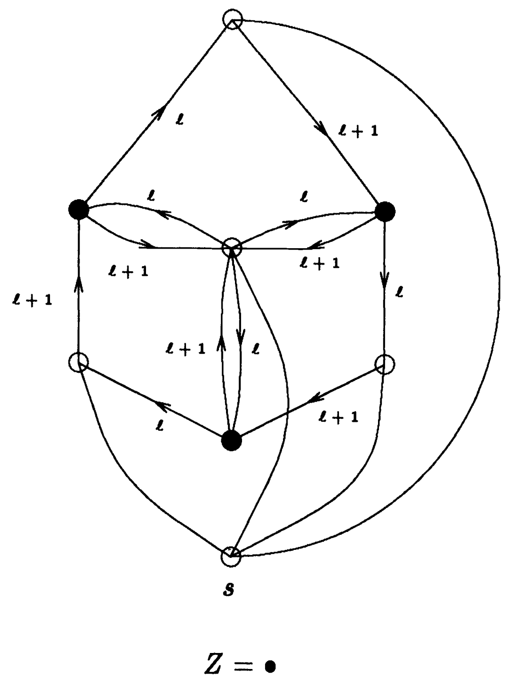


FIG. 4. *Here $\ell$ and $\ell + 1$ denote the multiplicity of the arcs. It is not difficult to see that* $\min(\lambda(x, y; M), \lambda(y, x; M)) = 2\ell+1$ *for any choice of $x, y \in Z$, and none of the two possible splittings at $s$ preserves this.*

of undirected augmentation is that for $k = 1$, the necessary and sufficient condition is different from the one given for $k > 1$.

Our proof method strongly follows that of Frank [6], which had two ingredients. The first was the splitting theorems of W. Mader, while the second was an observation that the set of degree vectors of possible augmentation forms a so-called contrapolymatroid, a matroid-like structure. The idea of using a splitting theorem for augmentation problems dates back to as early as 1976 (Plesnik [22]). Cai and Sun [1] also use splitting theorems. Actually, this approach was our main motivation in developing stronger splitting theorems in §§2 and 3.

To be more specific, let $N$ be a mixed graph composed from a directed graph $D = (V, A)$ and an undirected graph $G = (V, E)$. Let $T(D) := \{v \in V : \rho_D(v) \neq \delta_D(v)\}$ be the set of non-di-Eulerian nodes of $D$. Let $k$ be a positive integer and let $r(x, y)(x, y \in V)$ be a nonnegative integer-valued demand function satisfying

$$r(x, y) = r(y, x) \leq k \quad \text{for every } x, y \in V, \quad \text{and} \tag{4.1a}$$

$$r(x, y) \equiv k \quad \text{for every } x, y \in T(D). \tag{4.1b}$$

Let $R(\emptyset) = R(V) = 0$, and for $X \subseteq V$ let

$$R(X) := \max(r(x, y) : X \text{ separates } x \text{ and } y). \tag{4.2}$$

Let us define

$$q_{\text{in}}(X) := R(X) - \rho_N(X), \tag{4.3a}$$

$$q_{\text{out}}(X) := R(X) - \delta_N(X). \tag{4.3b}$$

THEOREM 4.1. *Given a mixed graph $N = (V, A \cup E)$, positive integers $k, \gamma$, and a demand function $r(x,y)$ satisfying* (4.1), *$N$ can be extended to a mixed graph $N^+$ by adding $\gamma$ new directed edges so that*

$$\lambda(x,y;N^+) \geq r(x,y) \quad \textit{for every } x,y \in V \tag{4.4}$$

*if and only if*

$$\sum q_{\text{in}}(X_i) \leq \gamma \tag{4.5a}$$

*and*

$$\sum q_{\text{out}}(X_i) \leq \gamma \tag{4.5b}$$

*hold for every subpartition $\{X_1, \dots, X_t\}$ of $V$.*

*Proof.* It can be assumed that $N$ is a directed graph because undirected edge of $N$ can be replaced by a pair of two oppositely directed edges, and this operation does not affect the local edge-connectivity. That is, $N = D$. Let $N^+$ denote an augmentation of $D$ with $\gamma$ new edges.

CLAIM 4.1. *$N^+$ satisfies* (4.4) *if and only if*

$$\rho_{N^+}(X) \geq R(X) \quad \textit{and} \quad \delta_{N^+}(X) \geq R(X) \tag{4.6}$$

*hold for every $X \subseteq V$.*

*Proof.* If $N^+$ satisfies (4.4), then for any subset $X$ separating $x$ and $y$, $\rho_{N^+}(X) \geq \lambda(x,y;N^+) \geq r(x,y)$. Hence $\rho_{N^+}(X) \geq R(X)$ for every $X \subseteq V$. The second inequality in (4.6) follows analogously. Conversely, assume that (4.6) is satisfied. By Menger's theorem there is a $y\bar{x}$-set $X$ for which $\lambda(x,y;N^+) = \rho_{N^+}(X)$. Hence $\lambda(x,y;N^+) = \rho_{N^+}(X) \geq R(X) \geq r(x,y)$ as required. □

We first examine the proof of necessity. By (4.6) we have $\gamma \geq \sum \rho_{N^+}(X_i) - \sum \rho_D(X_i) \geq \sum R(X_i) - \sum \rho_D(X_i) = \sum q_{\text{in}}(X_i)$, that is, (4.5a) holds. Inequality (4.5b) follows analogously.

The proof of sufficiency is structured as follows. First, we extend $D$ by adding a new node $s$ together with new directed and undirected edges incident to $s$. Secondly, we get rid of some new edges. Finally, we replace each remaining undirected edge by a pair of oppositely directed edges and apply Theorem 2.4.

To be more specific, extend $D$ by adding a new node $s$, $k$ parallel undirected edges connecting $s$ and $x$ for every $x \in V - T(D)$, and $k$ parallel directed edges from $s$ to $x$ and $x$ to $s$ for every $x \in T(D)$. The resulting mixed graph $M$ satisfies

$$\rho_M(X) \geq R(X), \tag{4.7a}$$

$$\delta_M(X) \geq R(X) \tag{4.7b}$$

for every $X \subseteq V$.

Let $s(X) = \beta_M(X) - R(X)$, $s_{\text{in}}(X) = \rho_M(X) - R(X)$, and $s_{\text{out}}(X) = \delta_M(X) - R(X)$ for $X \subset V$. By (4.7) these "surplus" functions are nonnegative. We say that $X$ is *R-tight, in-R-tight,* and *out-R-tight* if $s(X) = 0$, $s_{\text{in}}(X) = 0$, and $s_{\text{out}}(X) = 0$, respectively.

Secondly, starting with the undirected edges and then continuing with the directed ones, discard new edges from $M$ one by one as long as possible without violating

(4.7). Henceforth we use $M$ to denote the final graph. Recall the notation $\beta_M(X) = \min(\rho_M(X), \delta_M(X))$. During this process new $R$-tight sets may arise, and if a set becomes $R$-tight at any moment, it stays so throughout.

LEMMA 4.1. *$\delta_M(S) \leq \gamma$ and $\rho_M(s) \leq \gamma$.*

*Proof.* We prove only the first inequality; the second is analogous. In $M$,

(4.8) every directed edge $e = sx$ enters an in-$R$-tight subset of $V$,

since otherwise $e$ could have been discarded without violating (4.7).

We also claim that in $M$,

(4.9) every undirected edge $sx$ enters an in-$R$-tight set $X \subseteq V - T(D)$.

Indeed, let $M'$ denote the current graph at the moment of the discarding phase when the last undirected edge has been discarded. The fact that $e$ cannot be discarded means that there exists a set $X \subseteq V$ containing $x$ so that $\beta_{M'}(X) = R(X)$. Since at this moment no new directed edge has yet been discarded, $X$ cannot contain any element of $T(D)$. That is, $X \subseteq V - T(D)$ and hence $\rho_{M'}(X) = \delta_{M'}(X)$, that is, (4.9) follows.

Let $S := \{x \in V - T(D)$: there is an undirected edge $sx$ of $M\}$. Let $S_{\text{in}} := \{x \in T(D)$: there is a directed edge $sx$ of $M\}$. Let us call an in-$R$-tight set $X$ *extreme* if there is no in-$R$-tight set $Y$ with $X \cap (S \cup S_{\text{in}}) \subset Y \cap (S \cup S_{\text{in}})$, and if $X \cap (S \cup S_{\text{in}}) = Y \cap (S \cup S_{\text{in}})$ for an in-$R$-tight set $Y$, then $X \subseteq Y$. Thus $X$ is as large inside $S \cup S_{\text{in}}$ as possible, and subject to this, $X$ is as small outside $S \cup S_{\text{in}}$ as possible. □

CLAIM 4.2. *For any two in-R-tight sets $X, Y$, at least one of the following holds:*

(a) *$X \cup Y$ is in-R-tight;*

(b) *both $X - Y$ and $Y - X$ are in-R-tight and $X \cap Y \cap (S \cup S_{\text{in}}) = \varnothing$;*

(c) *$T(D) \subseteq X \cup Y$ and $X \cap Y \cap T(D) \neq \varnothing$.*

*Proof.* If (1.5a) holds, then by (1.3a), $0 + 0 = s_{\text{in}}(X) + s_{\text{in}}(Y) \geq s_{\text{in}}(X \cap Y) + s_{\text{in}}(X \cup Y) \geq 0$. It follows that $s_{\text{in}}(X \cup Y) = 0$, that is, (a) holds.

Now suppose that (1.5a) does not hold. If $X \cap Y \cap T(D) \neq \varnothing$, then $T(D) \subseteq X \cup Y$, otherwise $R(X) = R(Y) = R(X \cap Y) = R(X \cup Y) = k$ and (1.5a) would hold. That is, we are at alternative (c).

So assume that $X \cap Y \cap T(D) = \varnothing$. Now (1.3b) applies to $M$ and by Lemma 1.1 inequality (1.5b) holds. We obtain $\rho_M(X) + \rho_M(Y) = \rho_M(X - Y) + \rho_M(Y - X) + \bar{d}_{D'}(X, Y) + 2\bar{d}_{G'}(X, Y)$, where $D'$ and $G'$ denote the directed and undirected part of $M$, respectively. Combining this inequality with (1.5b) we get $0 + 0 = s_{\text{in}}(X) + s_{\text{in}}(Y) \geq s_{\text{in}}(X - Y) + s_{\text{in}}(Y - X) + \bar{d}_{D'}(X, Y) + 2\bar{d}_{G'}(X, Y) \geq 0$. It follows that $s_{\text{in}}(X - Y) = 0 = s_{\text{in}}(Y - X)$ and $\bar{d}_{D'}(X, Y) = 0 = 2\bar{d}_{G'}(X, Y)$, that is, (b) holds. □

By (4.8), there is a family $\mathcal{F}_1$ of in-$R$-tight sets whose union includes $S_{\text{in}}$. We may choose $\mathcal{F}_1$ so that its members are extreme sets and $|\mathcal{F}_1|$ is minimum.

CLAIM 4.3. *$\mathcal{F}_1$ is either a subpartition or consists of two members whose union includes $T(D)$.*

*Proof.* If $\mathcal{F}_1$ is not a subpartition, then it has two members $X, Y$ with $X \cap Y \neq \varnothing$. Since the members of $\mathcal{F}_1$ are extreme and $|\mathcal{F}_1|$ is minimal, alternatives (a) and (b) cannot occur in Claim 4.2. Therefore, (c) must hold. □

Let $Z := T(D) \cup \bigcup(X : X \in \mathcal{F}_1)$. By (4.9), for every $y \in S - Z$ there is an in-tight set $Y \subseteq V - T(D)$ containing $y$. We claim that $Y \cap Z = \varnothing$. If not, then $X \cap Y \neq \varnothing$ for a member $X$ of $\mathcal{F}_1$. But this contradicts Claim 4.2 because alternatives (a) and (b) cannot hold since $X$ is extreme; since (c) cannot hold either since $Y \cap T(D) = \varnothing$.

Therefore, there is a family $\mathcal{F}_2$ of in-$R$-tight subsets of $V-Z$ whose union includes $S-Z$. Assume that $|\mathcal{F}_2|$ is miniumum and, subject to this, $\sum(|X| : X \in \mathcal{F}_2)$ is minimum.

CLAIM 4.4. *$\mathcal{F}_2$ is a subpartition of $V-Z$.*

*Proof.* Indirectly, let $X, Y$ be two members of $\mathcal{F}_2$ with $X \cap Y \neq \varnothing$. This contradicts Claim 4.2 because the minimal choice of $\mathcal{F}_2$ implies that neither alternative (a) nor (b) may hold, and (c) is also impossible since $X, Y \subseteq V-Z \subseteq V-T(D)$. □

Let $\mathcal{F} := \mathcal{F}_1 \cup \mathcal{F}_2$. By Claim 4.4, if $\mathcal{F}_1$ is a subpartition so is $\mathcal{F}$. By (4.5a), $\gamma \geq \sum(R(X) - \rho_D(X) : X \in \mathcal{F}) = \sum(R(X) - \rho_M(X) : X \in \mathcal{F}) + \delta_M(s) = \delta_M(s)$ as required for the lemma.

If $\mathcal{F}_1$ is not a subpartition, then by Claim 4.3 it consists of two members $A, B$ with $T(D) \subseteq A \cup B$. Now $R(A-B) = R(A) = R(B-A) = R(B) = k$ and $\rho_D(X) = \delta_D(X)$ for every $X \in \mathcal{F}_2$. Furthermore, (1.2c) applies to $A$ and $B$, from which $\delta_D(A-B) + \delta_D(B-A) \leq \rho_D(A) + \rho_D(B)$.

By applying (4.5b) to the subpartition consisting of $A-B, B-A$, and the members of $\mathcal{F}_2$ we get $\gamma \geq [R(A-B) - \delta_D(A-B)] + [R(B-A) - \delta_D(B-A)] + \sum(R(X) - \delta_D(X) : X \in \mathcal{F}_2) \geq [R(A) - \rho_D(A)] + [R(B) - \rho_D(B)] + \sum(R(X) - \rho_D(X) : X \in \mathcal{F}_2) \geq [R(A) - \rho_M(A)] + [R(B) - \rho_M(B)] + \sum(R(X) - \rho_M(X) : X \in \mathcal{F}_2) + \delta_M(s) = \delta_M(s)$, and the proof of Lemma 4.1 is complete. □

By adding back some discarded new edges, if necessary, we may assume that $\delta_M(s) = \rho_M(s) = \gamma$. Now replace each undirected edge of $M$ by a pair of oppositely directed edges. By our construction, the resulting digraph $D'$ satisfies $T(D') \subseteq T(D)$. Therefore, we can apply Theorem 2.4 to $M := D'$. The resulting digraph $N^+ := M^+$ satisfies (4.4). □

*Remark.* It is interesting to note that if a node $v$ is di-Eulerian in $D$ (that is, $\rho_D(v) = \delta_D(v)$), then $v$ is di-Eulerian in the augmented $D^+$ as well.

Let us mention two corollaries. For simplicity we formulate them for directed starting graphs. In the first one we assume that $T$ is empty, that is, $D$ is di-Eulerian. Then $k$ may be chosen arbitrarily large and hence $r$ is not bounded above.

COROLLARY 4.1. *Given a di-Eulerian digraph $D = (V, A)$ and a symmetric demand function $r, D$ can be extended to a digraph $D^+$ by adding $\gamma$ new edges so that $\lambda(x, y; D^+) \geq r(x, y)$ for every $x, y \in V$ if and only if $\sum q_{\text{in}}(X_i) \leq \gamma$ and $\sum q_{\text{out}}(X_i) \leq \gamma$ hold for every subpartition $\{X_1, \dots, X_t\}$ of $V$. Furthermore, $D^+$ may be chosen to be di-Eulerian.*

In the second application we do not have any positive demand for di-Eulerian nodes.

COROLLARY 4.2. *We are given a digraph $D = (V, A)$, positive integers $k, \gamma$, and a subset $T \subseteq V$ so that $\rho_D(v) = \delta_D(v)$ holds for every $v \in V - T$. $D$ can be extended to a digraph $D^+$ by adding $\gamma$ new directed edges so that $\lambda(x, y; D^+) \geq k$ for every $x, y \in T$ if and only if*

$$(4.10) \qquad \sum(k - \rho_D(X_i)) \leq \gamma \quad \textit{and} \quad \sum(k - \delta_D(X_i)) \leq \gamma$$

*hold for every subpartition $\{X_1, \dots, X_t\}$ of $V$ for which $X_i \cap T, T - X_i \neq \varnothing$ $(i = 1, \dots, t)$.*

The special case $T = V$ of this corollary was proven in Frank [6].

Our next goal is to prove an augmentation theorem when only undirected edges are allowed to be added. Our result is a generalization of Theorem 5.5 in [6], where the starting graph is an undirected graph. Let $N$ be a mixed graph composed from a directed graph $D = (V, A)$ and an undirected graph $G = (V, E)$, and let $r(x, y)$ be a demand function satisfying (4.1). We say that a component $C$ of $N$ is *marginal* (with respect to $r$) if $r(u,v) \leq \lambda(u,v;N)$ for every $u,v \in C$, and $r(u,v) \leq \lambda(u,v;N)+1$ for every $u,v$ separated by $C$. In other words, $C$ is marginal if we do not want to increase the local edge-connectivity between the element $C$, and the demand for increased local edge-connectivity between a node in $C$ and a node outside $C$ is 0 or 1.

THEOREM 4.2. *We are given a mixed graph $N$, integers $k \geq 2, \gamma \geq 0$, and a demand function $r(x,y)$ satisfying* (4.1) *so that there are no marginal components. $N$ can be extended to a mixed graph $N^+$ by adding $\gamma$ new undirected edges so that*

$$\lambda(x,y;N^+) \geq r(x,y) \quad \text{for every } x,y \in V \tag{4.11}$$

*if and only if*

$$\sum (R(X_i) - \beta_N(X_i)) \leq 2\gamma \tag{4.12}$$

*holds for every subpartition $\{X_1,\dots,X_t\}$ of $V$.*

*Remark.* If $N$ has a marginal component, then the above min-max theorem is not true as is shown by the empty graph on four nodes (taking $r(u,v) \equiv 1$). For the special case when $N$ is undirected, in [6, Thm. 5.3] a very simple reduction method was used to get rid of marginal components. The same method easily generalizes to mixed graphs. Since no new idea is required, we leave out the details.

*Proof.* Again we may assume that $N$ is a directed graph, that is, $N = D$. Let $N^+$ denote an augmentation of $D$ with $\gamma$ new undirected edges.

CLAIM 4.5. *$N^+$ satisfies* (4.11) *if and only if*

$$\beta_{N^+}(X) \geq R(X) \tag{4.13}$$

*holds for every $X \subseteq V$.*

*Proof.* First suppose that $N^+$ satisfies (4.11). By applying Lemma 1.4 to $N^+$, we obtain that $\beta_{N^+}(X) \geq r_{N^+}(x,y) \geq r(x,y)$ for any subset $X$ separating $x$ and $y$. Hence (4.11) follows.

Conversely, assume that (4.11) is satisfied. By Menger's theorem there is a $y\bar{x}$-set $X$ for which $\lambda(x,y;N^+) = \rho_{N^+}(X)$. Hence $\lambda(x,y;N^+) = \rho_{N^+}(X) \geq \beta_{N^+}(X) \geq R(X) \geq r(x,y)$ as required. □

We first examine the proof of necessity. If $N^+$ satisfies (4.11), then by Claim 4.5 there are at least $R(X) - \beta_N(X)$ new edges between $X$ and $V - X$. Therefore, the number $\gamma$ of new edges is at least half of $\sum_{i=1}^{t}(R(X_i) - \beta_N(X_i))$.

We now examine the proof of sufficiency. First, extend $D$ by adding a new node $s$ and $k$ parallel edges connecting $s$ and $x$ for every $x \in V$. The resulting mixed graph $M$ satisfies

$$\beta_M(X) \geq R(X) \tag{4.14}$$

for every $X \subseteq V$.

Second, discard new edges one by one as long as possible without violating (4.14). Henceforth we use $M$ to denote the final mixed graph and let $S := \{x \in V : \text{there is an edge in } M \text{ between } s \text{ and } x\}$. We call an $R$-tight set $X$ *extreme* if there is no $R$-tight set $Y$ with $X \cap S \subset Y \cap S$, and if, in addition, $X \cap S = Y \cap S$ for an $R$-tight set $Y$, then $X \subseteq Y$.

LEMMA 4.2. $d_M(s) \leq 2\gamma$.

*Proof.* Since no further new edge can be left out of $M$ without violating (4.14), there is a family $\mathcal{F}$ of $R$-tight sets whose union includes $S$. We may choose $\mathcal{F}$ so that its members are extreme and $|\mathcal{F}|$ is minimum. □

CLAIM 4.6. *$\mathcal{F}$ is a subpartition of $V$.*

*Proof.* Assume indirectly that $X \cap Y \neq \varnothing$ for some $X, Y \in \mathcal{F}$. By Lemma 1.2, at least one of the following inequalities holds:

$$\begin{aligned} & 0+0 = s(X) + s(Y) \geq s(X \cap Y) + s(X \cup Y) + 2d_{G'}(X,Y) \\ \text{(4.15a)} \qquad & \qquad + d_D(X,Y) \geq 0, \\ & 0+0 = s(X) + s(Y) \geq s(X - Y) + s(Y - X) + 2\bar{d}_{G'}(X,Y) \\ \text{(4.15b)} \qquad & \qquad + \bar{d}_D(X,Y) \geq 0, \end{aligned}$$

where $G'$ denotes the undirected part of $M$.

If (4.15a) holds, then $s(X \cup Y) = 0$, that is, $X \cup Y$ is $R$-tight, contradicting the fact that $X, Y$ are extreme. If (4.15b) holds, then $s(X - Y) = s(Y - X) = 0$ and $\bar{d}_{G'}(X,Y) = 0$. Therefore, both $X - Y$ and $Y - X$ are $R$-tight and $X \cap Y \cap S = \varnothing$, which again contradicts the extremality of $X$ and $Y$. □

By (4.12), $2\gamma \geq \sum(R(X) - \beta_D(X) : X \in \mathcal{F}) = \sum(R(X) - \beta_M(X) : X \in \mathcal{F}) + d_M(s) = d_M(s)$, and Lemma 4.2 follows. □

By adding back new edges which are parallel to existing new edges, we may assume that $d_M(s) = 2\gamma$. We claim that there is no cut-edge of $M$ incident to $s$. Indeed, if $e = st$ were such an edge, then let $C$ denote the component of $M - e$ containing $t$. Since $e$ is the only edge of $M$ leaving $C, C$ is a marginal component contradicting the hypothesis.

The theorem now immediately follows from Theorem 3.3. □

If $N$ is an undirected graph in Theorem 4.2, then every node is di-Eulerian, and hence $r$ may be an arbitrary symmetric function. Therefore, Theorem 4.2 is a generalization of the following result from [6].

COROLLARY 4.3. *Given an undirected graph $G = (V, E)$ and a symmetric demand function $r(x,y) \geq 2$, it is possible to add $\gamma$ new undirected edges to $G$ so that in the resulting graph $G^+, \lambda(x,y;G^+) \geq r(x,y)$ holds for every pair of nodes $x, y$ if and only if $\sum(R(X_i) - d_G(X_i)) \leq 2\gamma$ holds for every subpartition $\{X_i\}$ of $V$.*

In another special case, $r \equiv k \geq 2$.

COROLLARY 4.4. *Let $N = (V, A \cup E)$ be a mixed graph and let $k \geq 2, \gamma \geq 1$ be integers. $N$ can be made $k$-edge connected by adding $\gamma$ new undirected edges if and only if*

$$\sum(k - \beta_N(X_i)) \leq 2\gamma$$

*holds for every subpartition $\{X_1, \ldots, X_t\}$ of $V$.*

The example in Fig. 5 shows that for $k = 1$, Corollary 4.4 (and hence Theorem 4.2) is not true in general. However, we can prove the following theorem.

THEOREM 4.3. *A mixed graph $N$ with connected underlying graph can be made 1-edge-connected (= strongly connected) by adding $\gamma$ new undirected edges if and only if (*) every family $\mathcal{F}$ of $\gamma + 1$ disjoint subsets of nodes contains (not necessarily distinct) members $X, Y$ for which $\rho_N(X) > 0$ and $\delta_N(Y) > 0$.*

*Proof.* First, suppose that $E'$ is a set of new undirected edges whose addition makes $N$ strongly connected, and there is a family $\mathcal{F}$ of $\gamma + 1$ disjoint subsets of $V$ so that for each member of $\mathcal{F}$, say, $\rho_N(X) = 0$. Since the underlying graph is connected, $X_0 := V - \bigcup(X : X \in \mathcal{F})$ is nonempty. Moreover, since there is no edge (undirected or directed) connecting distinct members of $\mathcal{F}, N - X_0$ has at least $\gamma + 1$ components.

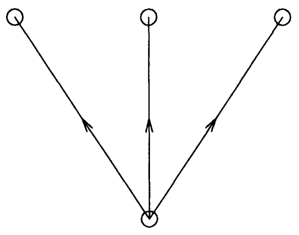


FIG. 5. *It is easy to see that we need three edges here, but $k = 1$ implies $\sum(k - \beta_N(X_i)) \leq 4$, which suggests that two edges would suffice.*

Since the union of any $j$ members of $\mathcal{F}(j = 1, \ldots, \gamma + 1)$ must be connected to the rest by an element of $E'$, we get $|E'| \geq \gamma + 1$, showing that $(*)$ is necessary.

To see the sufficiency we may assume again that $N$ is directed. Now $(^*)$ implies (4.10) for $k = 1$ and $T = V$, and hence, by Corollary 4.2, there are $\gamma$ directed edges whose addition makes $N$ strongly connected. If we leave out the orientations of the newly added edges we get the required undirected augmentations. □

**5. Variations, polyhedra, and algorithms.** In this section we briefly outline some variations of the augmentation problem, the polyhedral background, and some algorithmic aspects. We are concerned here with the case when only undirected edges are allowed to be added. A similar approach was discussed in detail in Frank [6]. Since no new idea is required, we refer the reader to that paper for definitions and details. For the directed augmentation problem, we have not yet found analogous methods to handle minimum node-cost and degree-constrained versions. This is a possible subject of future research.

Let $N, k, \gamma, r$ be the same as in Theorem 4.2, but this time, rather than finding a minimum cardinality augmentation, we are interested in an augmentation satisfying (4.11) in which the degree of every node is a prescribed value.

THEOREM 5.1. *Given a mixed graph $N = (V, A+E)$ and an integer-valued vector $m : V \rightarrow \mathbf{Z}^+$ for which*

$$m(V) \text{ is even,} \tag{5.1}$$

*$N$ can be extended to $N^+$ satisfying (4.11) by adding a set $F$ of new undirected edges for which $d_F(v) = m(v)$ for every $v \in V$ if and only if*

$$m(X) + \beta_N(X) \geq R(X) \text{ for every } X \subseteq V. \tag{5.2}$$

*Proof.* Add a new node $s$ and $m(v)$ parallel undirected edges between $s$ and $v$ for every $v \in V$. Since there is no marginal component of $N$, in the extended graph there is no cut-edge incident to $s$. We may apply Theorem 3.3 and the result immediately follows. □

From this easy derivation one should realize that Theorem 5.1 is nothing but a reinterpretation of Theorem 3.3. Let us call an integral-valued vector $m$ satisfying (5.1) and (5.2) an *augmentation vector.*

Let $q : 2^V \rightarrow \mathbf{Z}$ be an integer-valued set function. We call $q$ *skew supermodular* if $q(\varnothing) = 0$ and

$$q(X) + q(Y) \leq q(X \cap Y) + q(X \cup Y) \quad \text{or} \tag{5.3a}$$

$$q(X) + q(Y) \leq q(X - Y) + q(Y - X) \tag{5.3b}$$

hold for every pair of subsets $X, Y \subseteq V$ (we point out that other names were used earlier instead of skew supermodular, for example, weakly supermodular [10] and $X$-supermodular [8]). The following theorem was proved in [6, Thm. 7.1] for set functions of form $q(X) = R(X) - d_G(X)$. However, its proof relied only on one feature of $q$, namely, that $q$ is skew supermodular. Hence we state the theorem in this more general form.

THEOREM 5.2. *Where $q$ is a skew supermodular function, the polyhedron*

$$C(q) := \{z : \mathbf{R}^V : z \geq 0, z(X) \geq q(X) \textit{ for every } X \subseteq V\} \tag{5.4}$$

*is a contrapolymatroid $C(p)$, where the unique fully supermodular function $p$ defining $C(q)$ is given by*

$$p(A) := \max\left(\sum q(A_i) : \{A_1, \dots, A_t\} \textit{ a subpartition of } V\right). \tag{5.5}$$

By Lemma 1.2, $q := R - \beta_N$ is skew supermodular and hence Theorem 5.2 applies. We find that the augmentation vectors $m$ are precisely the integer-valued elements of $C(q)$ satisfying (5.1).

We briefly indicate how this fact can be used for degree-constrained and minimum node-cost augmentations. Suppose first that we have a nonnegative cost function $c : V \to \mathbf{R}_+$, and we are interested in an augmentation of a mixed graph $N$ that satisfies (4.11). Also, the total cost of new edges is minimum. Here the cost of an edge $uv$ is defined to be $c(u) + c(v)$.

It was shown in Frank [6] that with a slight modification of the greedy algorithm we can find a minimum cost integer element $m$ of a contrapolymatroid for which $m(V)$ is even. That is, with the help of the greedy algorithm, first find an integer vector $m'$ that minimizes $cx$ over $C(q)$. If $m'(V)$ is even, define $m = m'$. If $m'(V)$ is odd, define $m(v_n)$ by adding 1 to $m'(v_n)$, where $v_n$ is an element of $V$ of least cost, while $m(x) := m'(x)$ for $x \in V - v_n$. This way we obtain a minimum cost augmentation vector $m$ and, by Theorem 5.1, $m$ determines a minimum node-cost augmentation satisfying (4.11).

To consider the degree-constrained augmentation, let $f : V \to \mathbf{Z}$ and $g : V \to \mathbf{Z} \cup \{\infty\}$ be two functions with $f \leq g$. When does there exist an augmentation of $N$ satisfying (4.11) for which $f(v) \leq d_F(v) \leq g(v)$ holds for every $v \in V$? Let $B := \{x \in \mathbf{R} : f \leq x \leq g\}$ denote a box. By Theorem 5.1 the desired $F$ exists if and only if there is an integer element $m$ of $B \cap C(q)$ with the additional property that $m(V)$ is even. The intersection of a box and a contrapolymatroid is a generalized polymatroid. It was shown in [6, Prop. 6.10] that a $g$-polymatroid defined by a strong pair $(p, b)$ has no integer element $m$ for which $m(V)$ is even if and only if $p(V) = b(V)$ is odd (a submodular function $b$ and a supermodular function $p$ form a *strong pair* if $b(X) - p(Y) \geq b(X - Y) - p(Y - X)$ holds for all $X, Y \subset V$, where $V$ is the groundset for $b$ and $p$). From this one can derive the following theorem.

THEOREM 5.3. *Given $N, k, r$ as in Theorem* 4.2 *and integer-valued vector $f, g, N$ can be extended to $N^+$ satisfying* (4.11) *by adding a set $F$ of new undirected edges for which $f(v) \leq d_F(v) \leq g(v)$ for every $v \in V$ if and only if $q(X) \leq g(X)$ for every $\emptyset \subset X \subset V$, and there is no partition $\mathcal{F} := \{X_0, X_1, \dots, X_t\}$, where only $X_0$ may be empty, with the following properties: $f(X_0) = g(X_0), g(X_i) = q(X_i) (i = 1, \dots, t)$, and $g(V)$ is odd.*

Minimum node-cost degree-constrained augmentation problems can also be handled with the same technique.

To conclude, let's briefly say something about the algorithmic aspects. The proof of Theorem 4.2 consisted of two parts: the edge-deletion phase and the splitting-off phase. An argument analogous to the one used in [6] shows that the edge-deletion phase can be carried out on a graph with $n$ vertices by performing $2n^2$ MFMC computations. The splitting-off phase requires no more than $n^3$ MFMC calculations. Since one MFMC calculation can be carried out in $O(n^3)$ steps, the overall complexity of the algorithm is $O(n^6)$. Actually, these bounds are valid for the more general problem when the starting graph is endowed with integer capacities on the edges and we are allowed to add a new edge in any number of copies. Theoretically, this problem is not more general since we can replace an edge by as many parallel edges as its capacity will hold. But from a computational point of view such a reduction is not satisfactory. Fortunately, the MFMC algorithm is strongly polynomial, and hence the approach outlined above gives rise to strongly polynomial time algorithm in the capacitated case as well.

## REFERENCES

[1] G.-R. Cai and Y.-G. Sun, *The minimum augmentation of any graph to a k-edge-connected graph*, Networks, 19 (1989), pp. 151–172.

[2] J. Edmonds, *Edge-disjoint branchings*, in Combinatorial Algorithms, B. Rustin, ed., Academic Press, New York, 1973, pp. 91–96.

[3] ———, *Submodular functions, matroids, and certain polyhedra,* in Combinatorial Structures and their Applications, R. Guy, H. Hanani, N. Sauer, and J. Schonheim, eds., Gordon and Breach, New York, 1970, pp. 69–87.

[4] K. P. Eswaran and R. E. Tarjan, *Augmentation problems*, SIAM J. Comput., 5 (1976), pp. 653–665.

[5] A. Frank, *On connectivity properties of Eulerian digraphs*, Ann. Discrete Math., 41 (1989), pp. 179–194.

[6] ———, *Augmenting graphs to meet edge-connectivity requirements*, SIAM J. Discrete Math., 5 (1992), pp. 22–53.

[7] ———, *On a theorem of Mader*, Ann. Discrete Math., 101 (1992), pp. 49–57.

[8] ———, *Applications of submodular functions*, in Surveys in Combinatorics, Keith Walker, ed., London Math. Soc. Lecture Notes Ser. 187 (1993), pp. 85–136.

[9] H. N. Gabow, *Applications of a poset representation to edge-connectivity and graph rigidity*, Proc. 32nd IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1991, pp. 812–821.

[10] M. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, E. Tardos, and D. Williamson, *Improved approximation algorithms for network design problems*, in Proc. 4th ACM–SIAM Symposium on Discrete Algorithms, Arlington, VA, 1994, pp. 223–232.

[11] A. V. Goldberg and R. E. Tarjan, *A new approach to the maximum flow problem*, J. Assoc. Comput. Mach., 35 (1988), pp. 921–940.

[12] D. Gusfield, *Optimal mixed graph augmentation*, SIAM J. Comput., 16 (1987), pp. 599–612.

[13] B. Jackson, *Some remarks on arc-connectivity, vertex splitting, and orientation in digraphs*, J. Graph Theory, 12 (1988), pp. 429–436.

[14] A. V. Karzanov, *Determining the maximal flow in a network by the method of preflows*, Soviet Math. Dokl., 15 (1974), pp. 434–437.

[15] L. Lovász, *Connectivity in digraphs*, J. Combin. Theory Ser. B, 15 (1973), pp. 174–177.

[16] ———, *On two min-max theorems in graph theory*, J. Combin. Theory. Ser. B, 21 (1976), pp. 26–30.

[17] ———, *On some connectivity properties of Eulerian graphs*, Acta Math. Hungar., 28 (1976), pp. 129–138.

[18] ———, *Combinatorial Problems and Exercises*, North-Holland, Amsterdam, 1979.

[19] W. Mader, *A reduction method for edge-connectivity in graphs*, Ann. Discrete Math., 3 (1978), pp. 145–164.

[20] ———, *Konstruktion aller n-fach kantenzusammenhangenden Digrafen*, European J. Combin., 3 (1982), pp. 63–67.

[21] D. Naor, D. Gusfield, and C. Martel, *A fast algorithm for optimally increasing the edge-connectivity*, 31st IEEE Symposium on Foundations of Computer Science, St. Louis, MO, 1990, pp. 698–707.

[22] J. Plesnik, *Minimum block containing a given graph*, Archiv der Math. (Basel), XXVII (1976), pp. 668–672.

[23] Y. Shiloach, *Edge-disjoint branchings in directed multigraphs*, Inform. Process. Lett., 8 (1979), pp. 24–27.

[24] T. Watanabe and A. Nakamura, *Edge-connectivity augmentation problems*, Comput. System Sci., 35 (1987), pp. 96–144.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 179–185, May 1995

# COMPETITION GRAPHS OF STRONGLY CONNECTED AND HAMILTONIAN DIGRAPHS *

KATHRYN F. FRAUGHNAUGH†, J. RICHARD LUNDGREN†, SARAH K. MERZ†, JOHN S. MAYBEE‡, AND NORMAN J. PULLMAN§

**Abstract.** A radio communication network can be modeled by a digraph, $D$, where there is an arc from vertex $x$ to vertex $y$ if a signal sent from $x$ can be received at $y$. The competition graph, $C(D)$, of this network has the same vertex set as $D$, and $(x, y)$ is an edge in $C(D)$ if there is a vertex $z$ such that $(x, z)$ and $(y, z)$ are arcs in $D$. The competition graph can be used to assist in assigning frequencies to the transmitters in the network. Usually the digraphs for these networks are strongly connected, but the power of transmitters may vary, so they are not necessarily symmetric. Therefore it is of interest to determine which graphs are the competition graphs of strongly connected digraphs. We characterize these graphs as well as establish several large classes of graphs, including chordal, interval, and some triangle-free graphs, which are competition graphs of loopless Hamiltonian digraphs.

**Key words.** competition graph, conflict graph, strongly connected digraph, Hamiltonian digraph, communication network, edge clique cover, chordal graph, interval graph, cycle, inset

**AMS subject classification.** 05C25

## 1. Introduction.

A radio communication network can be modeled by a digraph, $D$, where there is an arc from vertex $x$ to vertex $y$ if a signal sent from $x$ can be received at $y$. The conflict graph, $C(D)$, of this network has the same vertex set as $D$, and $(x, y)$ is an edge in $C(D)$ if there is a vertex $z$ such that $(x, z)$ and $(y, z)$ are arcs in $D$. That is, signals sent from $x$ and $y$ can be received at $z$. The conflict graph can be used to assist in assigning frequencies to the transmitters in the network. Since it is desirable that a message initiated somewhere in the network be able to reach all stations, usually the digraphs for these networks are strongly connected. Nonetheless, in a very large network, with perhaps thousands of vertices, it is not unreasonable to expect that certain stations be equipped with very powerful transmitters while other stations cannot transmit signals as far. Thus the digraphs for these networks are not necessarily symmetric. One might start with a graph that is a conflict graph of a digraph and then change some of the arcs in the digraph so as to minimize or maximize the number of arcs while keeping the digraph strongly connected and the conflict graph the same. Hefner and Hintze [3] are developing algorithms to do this for a large naval communication network in the Pacific. So it is of interest to determine which graphs are the conflict graphs of strongly connected digraphs. We characterize these graphs as well as graphs that are the conflict graphs of Hamiltonian digraphs.

The conflict graph of a digraph is just another name for the competition graph

*Received by the editors March 25, 1991; accepted for publication (in revised form) April 14, 1994. The research of the authors was partially supported by Office of Naval Research research contracts N00014-88-K-0087 and N00014-91-J-1145.

† Department of Mathematics, University of Colorado at Denver, P.O. Box 173364, Denver, Colorado 80217-3364.

‡ Program in Applied Mathematics, University of Colorado at Boulder, Boulder, Colorado 80309-0526. The research of this author was partially supported by Natural Sciences and Engineering Council of Canada grant OGP0004041.

§ Department of Mathematics and Statistics, Queen's University at Kingston, Kingston, Ontario K7L 3N6, Canada.

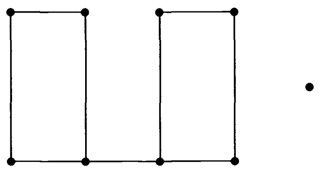


FIG.1. *G is the competition graph of a digraph but not one that is strongly connected.*

that has been studied extensively during the last 15 years (see Kim [4] and Lundgren [5] for recent surveys of the literature on competition graphs). Therefore we formulate our characterizations in terms of competition graphs. Characterizations have been found for competition graphs of acyclic digraphs [1], [5], arbitrary digraphs with or without loops [1], [11], and symmetric digraphs with or without loops [7], [9].

Let $\Theta_E(G)$ denote the edge clique cover number of $G$, that is, the smallest number of cliques that cover all the edges of $G$. The following result of Roberts and Steif [11] serves as a starting point for our work.

PROPOSITION 1.1. *If* $|V(G)| = n$, *then* $G$ *is a competition graph of a digraph that has no loops if and only if* $G \neq K_2$ *and* $\Theta_E(G) \leq n$.

## 2. Competition graphs of strongly connected digraphs.

The obvious question to ask first is whether or not the condition given in Proposition 1.1 will work for strongly connected digraphs as well. The graph given in Fig. 1 is the competition graph of a digraph but not one that is strongly connected. The problem is that while $\Theta_E(G) = 9, \Theta_E(G) + i(G) > 9$, where $i(G)$ is the number of isolated vertices in $G$. For a vertex $v$, we let $\text{In}(v)$ denote the inset of $v$, that is, the set of all vertices that have an arc into $v$. Similarly, Out $(v)$ will denote the outset of $v$. Unless stated otherwise, all digraphs are without loops.

PROPOSITION 2.1. *If* $|V(G)| = n$ *and* $G$ *is the competition graph of a strongly connected digraph, then* $\Theta_E(G) + i(G) \leq n$.

*Proof.* Suppose $G = C(D)$ for $D$ strongly connected. Then each inset in $D$ corresponds to a clique in $G$ and these $n$ cliques cover the edges of $G$. However, for $D$ to be strongly connected, each isolated vertex $x$ in $G$ has an arc in $D$ to a vertex $y$ that has inset equal to $\{x\}$. Thus, $\Theta_E(G) + i(G) \leq n$. □

Not only is this condition necessary, but it is also sufficient for $G$ to be the competition graph of a strongly connected digraph. First, we must establish the result for graphs with no isolated vertices.

THEOREM 2.2. *Let* $G$ *be a graph with no isolated vertices such that* $G \neq K_2$ *and* $\Theta_E(G) \leq n$. *Then* $G$ *is the competition graph of a strongly connected digraph.*

*Proof.* By Proposition 1.1, $G = C(\hat{D})$ for some (loopless) digraph $\hat{D}$. Among all such digraphs, let $D = (V, A)$ be one with the smallest number $k$ of strong components and suppose that $k \geq 2$.

Let $D_1, D_2, \ldots, D_k$ be the strong components of $D$ and suppose $V$ has been labeled so that if $x \in D_i, y \in D_j$, and $(x, y) \in A$, then $i \leq j$. See chapter 2 of Roberts [10] for results that prove such an ordering always exists. Observe that since $G$ has no isolated vertices, this ordering implies there are at least two vertices in $D_k$, since every vertex in $D$ has an outgoing arc.

*Case* 1. $D_1 = \{x\}$. Observe that $\text{In}(x) = \varnothing$. Since $G$ has no isolated vertices, $\text{Out}(x) \neq \varnothing$. Let $(x, y) \in A$ for some vertex $y$. Then get a new digraph $D'$ by simply adding the arc $(y, x)$ to $D$. Then, $D'$ has fewer strong components than $D$ and $C(D') = G$, a contradiction.

*Case* 2. $|D_1| \geq 2$ and there is a vertex $y \in D_k$ such that $(x,y) \notin A$ for some $x \in D_1$. Since $|D_k| \geq 2$ and $D_k$ is strongly connected, $\text{In}(y) \neq \varnothing$. As in Case 1 we construct a new digraph $D'$. Let $\text{In}_{D'}(y) = \text{In}_D(x)$ and $\text{In}_{D'}(x) = \text{In}_D(y)$ with all other arcs in $D'$ staying the same as in $D$. We claim that $D_1 \cup D_k$ is strongly connected in $D'$. To prove this, we must show that given arbitrary vertices $u, w \in D_1 \cup D_k$, there is a path from $u$ to $w$.

Let $u$ be a vertex in $D_1$ not equal to $x$ and $w$ a vertex in $D_k$ not equal to $y$. In $D, u$ reached $x$, so in $D', u$ reaches $y$. In $D, y$ reached $w$ and none of those arcs have changed; therefore, $u$ reaches all $w \in D_k$ in $D'$. In $D, x$ reached $u$ and none of those arcs have changed; therefore $x$ reaches $w$. Therefore every vertex in $D_1$ reaches every vertex in $D_k$.

In $D, w$ reached $y$, so in $D', w$ reaches $x$. In $D, x$ reached $u$ and none of those arcs have changed; thus $w$ reaches $u$. In $D, y$ reached $w$ and none of those arcs have changed; thus $y$ reaches $u$. Therefore every vertex in $D_k$ reaches every vertex in $D_1$.

Since every vertex in $D_1$ reaches every vertex in $D_k$ and every vertex in $D_k$ reaches every vertex in $D_1$, every vertex in $D_1$ reaches every vertex in $D_1$ and every vertex in $D_k$ reaches every vertex in $D_k$, completing the proof that $D_1 \cup D_k$ is strongly connected. So $D'$ has fewer strong components than $D$ and $C(D') = G$, a contradiction.

*Case* 3. $|D_1| \geq 2$ and every vertex in $D_1$ has an arc to all vertices in $D_k$. Let $x \in D_1$ and $y \in D_k$. Since every vertex in $D_1$ has an arc to $y, D_1$ is a clique in $C(D)$ and all arcs in the inset of $x$ can be removed in a new digraph $D'$ without changing the competition graph. Now let $\text{In}_{D'}(x) = \{y\}$ and let all other arcs in $D'$ remain the same as in $D$. Then $D_1 \cup D_k$ is strongly connected in $D'$ by arguments analogous to that in Case 2, so $D'$ has fewer strong components than $D$ and $C(D') = G$, a contradiction.

Thus $D$ is strongly connected, since we assumed that $k \geq 2$ was the smallest number of strongly connected components and in all cases reached a contradiction. □

COROLLARY 2.3. *Let $G$ be a graph such that $G \neq K_1$ or $K_2$ and $\Theta_E(G) + i(G) \leq n$. Then $G$ is the competition graph of a strongly connected digraph.*

*Proof.* If $G$ is a graph on $n$ isolated vertices, then $G$ is the competition graph of an $n$-cycle, so assume $G$ is not a graph on $n$ isolated vertices. If $G$ has no isolated vertices, then we are done by the previous theorem, so assume $i(G) = k \neq 0$. Let $I_k$ denote the subgraph of $G$ isomorphic to $k$ isolated vertices. Then there exists a graph $G'$ such that $G = G' \cup I_k$. Observe that $G'$ has $m = n - k$ vertices, none of which are isolated and $\Theta_E(G') \leq m$. Let $\{a_{m+1}, a_{m+2}, \ldots, a_{m+k}\}$ denote the $k$ vertices of $I_k$.

If $G' = K_2$, let $x$ and $y$ be the vertices of $G'$. Let $D$ be the digraph with cycle $x, y, a_{m+1}, a_{m+2}, \ldots, a_{m+k}$ and the arc $(x, a_{m+1})$. Then $C(D) = G$ and $D$ is strongly connected. If $G' \neq K_2$, by the previous theorem, there exists a strongly connected digraph $D' = (V', A')$ such that $C(D') = G'$. Create $D = (V, A)$ as follows. Let $V = V' + I_k$. Choose $x \in V'$. Let $\text{In}_D(a_{m+1}) = \text{In}_{D'}(x), \text{In}_D(x) = a_{m+k}$, and $(a_i, a_{i+1}) \in A'$ for $i = m+1, m+2, \ldots, m+(k-1)$. Leave all remaining arcs of $D'$ in $D$. Then none of the competitions in $D'$ have been changed so $C(D) = G$. We claim $D$ is strongly connected.

Since $D'$ is strongly connected, $x$ reaches every vertex $V'$. The only arcs that have changed are arcs from $\text{In}_{D'}(x)$. Therefore $x$ reaches every vertex in $V'$ in $D$. Let $u$ be a vertex in $\text{In}_{D'}(x)$. Clearly $u$ reaches every vertex in $I_k$ in $D$, and since $x$ reaches $u, x$ reaches every vertex in $I_k$. Therefore $x$ reaches every vertex in $D$. Let $v$ be an arbitrary vertex in $V'$. We claim $v$ reaches $x$. Let $u \in \text{In}_{D'}(x)$. There is a path from $u$

to $x$, namely $(u, a_{m+1}), (a_{m+1}, a_{m+2}), \ldots, (a_{m+k}, x)$, and $v$ reaches $u$ in $D$ because $v$ reaches $u$ in $D'$ therefore $v$ reaches $x$ in $D$. Clearly an arbitrary vertex $v$ in $I_k$ reaches $x$. Thus every vertex in $I_k$ and $V'$ reaches $x$ in $D$. Thus $D$ is strongly connected. $\square$

The previous theorems and corollary give the following extension of the result of Roberts and Steif [11].

COROLLARY 2.4. *If $G$ is a graph and $|V(G)| = n \geq 3$, then $G$ is the competition graph of a strongly connected digraph if and only if $\Theta_E(G) + i(G) \leq n$.*

**3. Competition graphs of Hamiltonian digraphs.** In looking at examples of graphs on $n$ vertices satisfying $\Theta_E(G) + i(G) \leq n$, we discovered that in many cases not only are the graphs competition graphs of strongly connected digraphs but frequently the digraphs could be chosen to be Hamiltonian. This raised the question of finding necessary and sufficient conditions on $G$ for $G = C(D)$ for $D$ Hamiltonian. From Proposition 2.1, certainly $\Theta_E(G) + i(G) \leq n$ is necessary, but is it sufficient? As we shall see, the answer in general is no, but first we present several classes of graphs for which the answer is yes. Our first result gives an edge clique covering characterization of such graphs analogous to the results on competition graphs of acyclic digraphs found by Brigham and Dutton [1] and Lundgren and Maybee [6].

THEOREM 3.1. *A graph $G$ with $|V(G)| = n$ is the competition graph of a Hamiltonian digraph if and only if $G$ has a labeling $a_1, a_2, \ldots, a_n$ of its vertices and an edge clique covering $\{C_1, \ldots, C_n\}$ satisfying $a_i \notin C_i, i = 1, \ldots, n, a_i \in C_{i+1}, i = 1, \ldots, n-1$, and $a_n \in C_1$.*

*Proof.* Suppose $G = C(D)$ for $D$ Hamiltonian and $a_1, \ldots, a_n$ is a Hamiltonian cycle in $D$. Let $C_i = \text{In}(a_i)$. Then $\{C_1, \ldots, C_n\}$ is an edge clique covering of $G$, $a_i \notin C_i$ since $D$ has no loops, $a_i \in C_{i+1}$ since $(a_i, a_{i+1})$ is an arc, and $a_n \in C_1$ since $(a_n, a_1)$ is an arc.

Now suppose we have a labeling and an edge clique covering as stated in the theorem. Construct $D$ with vertices $a_1, \ldots, a_n$ and $\text{In}(a_i) = C_i$. Then $a_i \in C_{i+1}$ implies that $(a_i, a_{i+1})$ is an arc and $a_n \in C_1$ since $(a_n, a_1)$ is an arc; therefore $D$ is Hamiltonian. From the construction, it is clear that $C(D) = G$. $\square$

The next several results give classes of graphs satisfying $\Theta_E(G) + i(G) \leq n$ that are the competition graphs of Hamiltonian digraphs.

PROPOSITION 3.2. *If $G$ is a cycle, then $G$ is the competition graph of a Hamiltonian digraph.*

*Proof.* Let $a_1, \ldots, a_n$ be the vertices of the cycle. Let $C_1 = \{a_{n-1}, a_n\}, C_2 = \{a_n, a_1\}, C_3 = \{a_1, a_2\}, \ldots, C_n = \{a_{n-2}, a_{n-1}\}$. Then $\{C_1, \ldots, C_n\}$ is an edge clique covering of $G$ satisfying Theorem 3.1. $\square$

PROPOSITION 3.3. *The complete graph $K_n$ for $n \geq 3$ is the competition graph of a Hamiltonian digraph.*

*Proof.* Let $V = \{a_1, \ldots, a_n\}$ be the vertices of $K_n$. Let $C_i = V - \{a_i\}$. Then $\{C_1, \ldots, C_n\}$ is an edge clique cover of $K_n$ satisfying Theorem 3.1. $\square$

Two other classes of graphs that are competition graphs of Hamiltonian digraphs are chordal graphs and interval graphs. See Chapters 4 and 8 of Golumbic [2] for characterizations and more information about these graphs.

PROPOSITION 3.4. *If $G$ is a chordal graph on $n \geq 3$ vertices, then $G$ is the competition graph of a Hamiltonian digraph.*

*Proof.* The proof is by induction on the number of vertices. It is easy to verify the theorem for chordal graphs with three vertices. Assume the result is true for all chordal graphs with less than $n$ vertices and suppose $G$ is a chordal graph with $n$ vertices. If $G = K_n$, then we are done by Proposition 3.3. So suppose $G \neq K_n$ and let

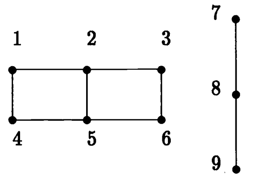


FIG. 2. *This graph is not the competition graph of a Hamiltonian digraph.*

$x$ be a simplicial vertex in $G$ and let $G' = G - \{x\}$. Since $G'$ is chordal, $G' = C(D')$ where $D'$ has the Hamiltonian cycle $a_1, a_2, \ldots, a_{n-1}$. Since $G \neq K_n$, there is a vertex $a_j \notin N[x]$, the closed neighborhood of $x$. Note that $N[x]$ is a clique since $x$ is simplicial. We define a new digraph $D$ as follows: $\text{In}_D(x) = \text{In}_{D'}(a_j); \text{In}_D(a_j) = N[x]$; and $\text{In}_D(a_i) = \text{In}_{D'}(a_i)$ for all $i \neq j$. Clearly $C(D) = G$ and $a_1, a_2, \ldots, a_{j-1}, x, a_j, \ldots, a_{n-1}$ is a Hamiltonian cycle in $D$. □

We note that since every chordal graph has a perfect elimination scheme, it automatically satisfies $\Theta_E(G) + i(G) \leq n$. Since the problem of determining the structure of digraphs with interval competition graphs remains an interesting open problem, we include the following corollary.

COROLLARY 3.5. *If $G$ is an interval graph on $n \geq 3$ vertices, then $G$ is the competition graph of a Hamiltonian digraph.*

We now prove that connected triangle-free graphs on three or more vertices are the competition graphs of Hamiltonian digraphs. To obtain this result, we need the following lemma.

LEMMA 3.6. *If $G$ is a connected triangle-free graph with $|V(G)| = n$ and $\Theta_E(G) \leq n$, then $G$ is a tree or a tree with an additional edge.*

*Proof.* Since $G$ is triangle-free, the cliques in any clique cover of minimum cardinality are just the edges of $G$. Since $G$ is connected and $\Theta_E(G) \leq n$, then either $e(G) = n - 1$ and $G$ is a tree or $e(G) = n$ and $G$ is a tree plus an edge. □

PROPOSITION 3.7. *If $G$ is a connected triangle-free graph on $n \geq 3$ vertices with $\Theta_E(G) \leq n$, then $G = C(D)$ for $D$ Hamiltonian.*

*Proof.* The proof is by induction on the number of vertices $n$. For $n = 3$, a path on three vertices is the only such graph, and by Proposition 3.4 a path on three vertices is the competition graph of a Hamiltonian digraph. Assume the result is true for all connected triangle-free graphs with $n$ vertices and suppose $G$ has $n + 1$ vertices. By Lemma 3.6, $G$ is either a tree or a tree plus an edge. If $G$ is a tree, we are done by Proposition 3.4, so assume $G$ is a tree plus an edge. If $G$ has no pendant vertices, then $G$ is a cycle and the result follows from Proposition 3.3, so assume $G$ has a pendant vertex. Let $x$ be a pendant vertex in $G$. Then $G - \{x\}$ satisfies the induction hypothesis. Let $a_1, \ldots, a_n$ be the vertices of the Hamiltonian cycle. Since $n \geq 3$ and $\deg(x) = 1$, there is a vertex $a_j$ such that $(x, a_j)$ is an edge in $G$ and a vertex $a_k$ such that $(x, a_k)$ is not an edge in $G$. We construct a new digraph $D'$ as follows: $\text{In}_{D'}(x) = \text{In}_D(a_k), \text{In}_{D'}(a_k) = \{x, a_j\}, \text{In}_{D'}(a_i) = \text{In}_D(a_i)$ for $i \neq k$. Then $C(D') = G$ and $a_1, a_2, \ldots, a_{k-1}, x, a_k, \ldots, a_n$ is a Hamiltonian cycle in $D'$, completing the proof. □

We now turn to the question of whether or not all graphs that satisfy $\Theta_E(G) + i(G) \leq n$ are the competition graphs of Hamiltonian digraphs. In fact, not all triangle-free graphs satisfying this condition work. For example, neither the triangle-free graph in Fig. 2 nor the graph in Fig. 3 is the competition graph of a Hamiltonian digraph. The reasons are a consequence of the following result.

THEOREM 3.8. *If $G$ is the competition graph of a Hamiltonian digraph, then $G$*

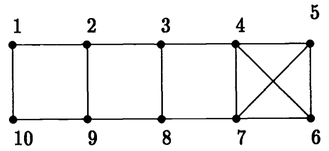


FIG. 3. *This graph is not the competition graph of a Hamiltonian digraph.*

*has an edge clique covering that has a system of distinct representatives among the vertices of the cliques.*

*Proof.* Suppose $G = C(D)$, where $D$ is Hamiltonian and let $a_1, \ldots, a_n$ be the Hamiltonian cycle in $D$. Let $C_i = \text{In}(a_i)$, then $\mathcal{C} = \{C_1, \ldots, C_n\}$ is an edge clique covering of $G$, and $\{a_n, a_1, a_2, \ldots, a_{n-1}\}$ is a system of distinct representatives (SDR) for $\mathcal{C}$. □

COROLLARY 3.9. *Suppose that $G$ is a graph with components $G_i$, each of which has $n_i$ vertices. If $G$ is the competition graph of a Hamiltonian digraph, then $\Theta_e(G_i) \leq n_i$ for each component $G_i$.*

*Proof.* Suppose that $G$ is the competition graph of a Hamiltonian digraph. If $\Theta_E(G_i) > n_i$ for some $i$, then every edge clique cover $\mathcal{C}$ of $G$ requires more than $n_i$ cliques to cover $G_i$, and each of these cliques contains only vertices of $G_i$. It follows that $\mathcal{C}$ does not have a system of distinct representatives, a contradiction. □

For the graph in Fig. 2, although $\Theta_E(G) + i(G) \leq n$, seven cliques are required to cover the component determined by $\{1, 2, 3, 4, 5, 6\}$, so no edge clique cover contains an SDR. For the graph in Fig. 3, the fact that $\Theta_E(G) = 10$ and that 5 and 6 are adjacent simplicial vertices causes the problem, as described in the following corollary to Theorem 3.8.

COROLLARY 3.10. *If $G$ is a graph on $n$ vertices satisfying $\Theta_E(G) = n$ and $G = C(D)$ for $D$ Hamiltonian, then $G$ contains no adjacent pair of simplicial vertices.*

*Proof.* Suppose that $G = C(D)$ for $D$ Hamiltonian and that $\Theta_E(G) = n$. Suppose that $x$ and $y$ are simplicial vertices in $G$ that are adjacent. Then $N[x, y]$ is a clique, so $G - \{x, y\}$ requires at least $n - 1$ cliques in any clique cover. But then no edge clique cover of $G$ contains an SDR, a contradiction. □

We can use Theorem 3.8 to determine exactly which triangle-free graphs are competition graphs of Hamiltonian digraphs.

THEOREM 3.11. *Suppose that $G$ is a triangle-free graph with components $G_i$ each of which has $n_i$ vertices. Then $G$ is the competition graph of a Hamiltonian digraph if and only if $\Theta_E(G_i) \leq n_i$ for each component $G_i$.*

*Proof.* One direction of the proof follows from Corollary 3.9. Suppose that $\Theta_E(G_i) \leq n_i$ for each component $G_i, 1 \leq i \leq k$, where $k$ is the number of components in $G$. If $n_i \geq 3$, then $G_i$ is the competition graph of a Hamiltonian digraph $D_i$ with Hamiltonian cycle $a_1^i, \ldots, a_{n_i}^i$ by Proposition 3.7. If $n_i = 2$, then $G_i = K_2$, and we let $D_i$ be the arc $(a_1^i, a_2^i)$ (observe $C(D_i) \neq G_i$ in this case). If $n_i = 1$, then $D_i$ is the isolated vertex $a_1^i$. Let $D = \bigcup_{i=1}^k D_i$. We can now form a digraph $D'$ such that $C(D') = G$ and $D'$ is Hamiltonian. For $i = 1, \ldots, k-1$, if $n_i \geq 3$, let $\text{In}_{D'}(a_1^{i+1}) = \text{In}_D(a_1^i)$. If $n_k \geq 3$, let $\text{In}_{D'}(a_1^1) = \text{In}_D(a_1^k)$. For $i = 1, \ldots, k-1$, if $n_i = 2$, let $\text{In}_{D'}(a_1^{i+1}) = \{a_1^i, a_2^i\}$. If $n_k = 2$, let $\text{In}_{D'}(a_1^1) = \{a_1^k, a_2^k\}$. For $i = 1, \ldots, k-1$, if $n_i = 1$, let $\text{In}_{D'}(a_1^{i+1}) = a_1^i$. If $n_k = 1$, let $\text{In}_{D'}(a_1^1) = a_1^k$. Let all other insets in $D'$ be the same as in $D$. Observe we have not changed any existing competitions from $C(D)$ to $C(D')$. Thus if $n_i \neq 2$, we get all the edges of $G_i$ in $C(D)$. For $i = 1, \ldots, k-1$, if $n_i = 2$, then $\text{In}_{D'}(a_1^{i+1}) = \{a_1^i, a_2^i\}$ gives the edge in $G_i = K_2$. Similarly, $\text{In}_{D'}(a_1^1)$ does this if $n_k = 2$. Thus $C(D') = G$ and $D'$ has the Hamiltonian cycle $a_1^1, \ldots, a_{n_1}^1, a_1^2, \ldots, a_{n_2}^2, \ldots, a_1^k, \ldots, a_{n_k}^k$. □

We would like to improve on the characterization given in Theorem 3.1. One possibility for $n \geq 3$ and $G$ satisfying $\Theta_E(G) \leq n$ and no isolated vertices is the converse of Theorem 3.8. While we have yet to find a counterexample, we have not been able to show that this condition is sufficient to get a Hamiltonian digraph without loops. We close this section with the following result analogous to the result of Brigham and Dutton [1] for acyclic digraphs.

THEOREM 3.12. *Suppose $G$ is a graph on $n \geq 3$ vertices satisfying $\Theta_E(G) \leq n$ and $i(G) = 0$. Then $G$ is the competition graph of a Hamiltonian digraph possibly having loops if and only if $G$ has an edge clique covering $\mathcal{C} = \{C_1, \ldots, C_n\}$ that has a system of distinct representative.*

*Proof.* One direction of the proof follows from Theorem 3.8. So suppose $G$ has such an edge clique covering $\mathcal{C}$. Let the representative for clique $C_i$ be labeled $a_{i-1}$ for $i = 2, \ldots, n$ and the representative for $C_1$ be labeled $a_n$. Define a digraph $D$ by $\text{In}(a_i) = C_i$. Then $G = C(D)$ and $a_1, \ldots a_n$ is a Hamiltonian cycle in $D$. $\square$

**4. Observations and future research.** One obvious area for future research is improvement on the theorem of competition graphs of Hamiltonian digraphs. Perhaps some modification of Theorem 3.12 holds for digraphs without loops.

If $G$ is a graph satisfying $\Theta_E(G) + i(G) \leq n$, what is needed are algorithms or at least heuristics for generating strongly connected digraphs satisfying $C(D) = G$. If $D$ represents a communication network, one might want the heuristics to minimize or maximize the number of arcs in $D$ without changing $G$. Hefner and Hintze [3] have some results in this area.

These new characterizations suggest another problem. Suppose $\Theta_E(G) + i(G) \leq n$. Let $\mathcal{D}(G) = \{D | C(D) = G\}$. What can be said about $\mathcal{D}(G)$? For example, we know that $\mathcal{D}(G)$ contains a strongly connected digraph. If $G$ is interval with an isolated vertex, then $\mathcal{D}(G)$ also contains an acyclic digraph as well as a Hamiltonian digraph. This approach may shed some light on the problem of characterizing digraphs with interval competition graphs. For a particular class of graphs, one might determine canonical representatives in $\mathcal{D}(G)$. This problem is investigated in Lundgren and Merz [8].

## REFERENCES

[1] R. C. BRIGHAM AND R. D. DUTTON, *A characterization of competition graphs*, Discrete Appl. Math., 6 (1983), pp. 315–317.

[2] M. C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[3] K. A. S. HEFNER AND D. W. HINTZE, *Maximizing arcs in a network for a given conflict graph*, to appear.

[4] S. R. KIM, *Competition Graphs and Scientific Laws for Food Webs and Other Systems*, PhD thesis, Department of Mathematics, Rutgers University, 1988.

[5] J. R. LUNDGREN, *Food webs, competition graphs, competition-common enemy graphs, and niche graphs*, Math. Appl., 17 (1989), pp. 221–244.

[6] ——— AND J. S. MAYBEE, *A characterization of graphs of competition number m*, Discrete Appl. Math., 6 (1983), pp. 319–322.

[7] ———, ———, AND C. W. RASMUSSEN, *Interval competition graphs of symmetric digraphs*, Discrete Math., 119 (1993), pp. 113–122.

[8] ——— AND S. K. MERZ, *Inversion of competition graphs*, Congr. Numer., to appear.

[9] A. RAYCHAUDHURI AND F. S. ROBERTS, *Generalized competition graphs and their applications*, Methods Oper. Res., 49 (1985), pp. 295–311.

[10] F. S. ROBERTS, *Discrete Mathematical Models*. Prentice-Hall, Englewood Cliffs, NJ, 1976.

[11] ——— AND J. E. STEIF, *A characterization of competition graphs of arbitrary digraphs*. Discrete Appl. Math., 6 (1983), pp. 323–326.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 186–195, May 1995

# A COMPETITIVE STRONG SPANNING TREE ALGORITHM FOR THE MAXIMUM BIPARTITE MATCHING PROBLEM *

JAIME GONZÁLEZ† AND OSVALDO LANDAETA†

**Abstract.** The new characterization for maximum matching in bipartite graphs given by Balinski and González is based on "strong spanning trees" and is independent on the classical notion of augmenting path. However, the algorithm that they derived runs in $0(|V||E|)$ time for bipartite graphs with $|V|$ nodes and $|E|$ edges, and so it is not competitive with the $0(\sqrt{|V|}|E|)$ algorithm of Hopcroft and Karp. In this paper we prove that the basic results given by Hopcroft and Karp can also be obtained using the new characterization, allowing us to develop a competitive algorithm.

**Key words.** bipartite graphs, maximum matching, augmenting paths, strong spanning trees

**AMS subject classification.** 90

## 1. Introduction.

A *matching* $Z$ in an undirected finite graph $G = (V, E)$ with node set $V$ and edge set $E$ is a subset of $E$ such that no two edges of $Z$ have a common node. The *maximum matching problem* is to find a matching whose number of edges is maximum.

Balinski and González [3] presented a new characterization and a new algorithm to solve the maximum matching problem in bipartite graphs; these are based on "strong spanning trees," a concept introduced by Balinski [2]. This new approach is independent of the classical notion of *augmenting paths,* on which all the relevant results (related to the problem) previously written are based.

The classical notion states that a given matching $Z$ is maximum if and only if there exists no augmenting path with respect to $Z$ (proved independently by Berge [4] and Norman and Rabin [11]). This, in turn, gives a general procedure. Starting with any matching (e.g., the empty one), a search for an augmenting path is conducted. If the search finds one, the matching is augmented and the procedure is repeated (using the resulting matching) until no augmenting path is found.

For the bipartite case, direct implementations of the above procedure give algorithms that run in $0(|V||E|)$ time (e.g., Hall [7], Ford and Fulkerson [5], Kuhn [10]). The best algorithm in this case is that of Hopcroft and Karp [8], which has a running time of $0(\sqrt{|V|}|E|)$. To obtain such a bound, they proved that the construction of a maximum matching can be performed in $\sqrt{|V|}$ "phases." For a given matching $Z$, a phase consists of finding a maximal set of node-disjoint, minimum-length, augmenting paths with respect to $Z$. They showed that each phase can be carried out in $0(|E|)$ time. Alt et al. [1] give an implementation of the Hopcroft–Karp algorithm, which runs in time $0(n^{1.5}\sqrt{m/\log n})$, an improvement by a factor of $\sqrt{\log n}$ for dense graphs.

The new algorithm of Balinski and González runs in $0(|V||E|)$ time and so is competitive with all algorithms that use augmenting paths except that of Hopcroft and Karp.

In this paper, we show that the approach originating from the concept of strong

* Received by the editors April 17, 1992; accepted for publication (in revised form) April 14, 1994. This research was supported in part by Fondo Desarrollo Cientifico y Tecnologico (FONDECYT) projects 91-1202 and 1930814.

† Departamento de Ingeniería Matemática, Universidad de Chile, Casilla 170/3, Correo 3, Santiago, Chile.

spanning trees is actually related to the classical notion of augmenting paths, allowing us to follow and reproduce the basic results given by Hopcroft and Karp, which in turn allows us to derive a competitive algorithm.

**2. Preliminaries and basic results.** Let $G = (M, N, E)$ be a finite, connected, undirected, bipartite graph with node set $M \cup N$ and edge set $E \subset M \times N$, where $M = \{1, 2, \ldots, m\}$ and $N = \{1, 2, \ldots, n\}$. The elements of $M$ and $N$ are called *row* and *column* nodes, respectively. It will be assumed that $m \geq n$ and that $(i, j) \in E$ means $i \in M$ and $j \in N$.

Let $Z$ be a given matching of $G$. A row or column node $h$ is *free* or *unmatched* if $h$ is incident with no edge of $Z$. The edges of $Z$ are said to be matched and the ones of $E - Z$ unmatched. An *augmenting path* is a path of distinct nodes whose edges are alternatively in $E - Z$ and $Z$ and whose endpoints are both free (if one of them is a row node, the other one is a column node). If $\mu$ is any augmenting path with respect to $Z$, its length (number of edges), denoted by $|\mu|$, is odd. The following are well-known results for the maximum matching problem.

LEMMA 1. *Let $Z$ be a matching of $G$. If $\mu$ is any augmenting path with respect to $Z$, then $Z\Delta\mu = (Z - \mu) \cup (\mu - Z)$ is a matching of $G$ and $|Z\Delta\mu| = |Z| + 1$.*

THEOREM 1 [4], [11]. *$Z$ is a maximum matching of $G$ if and only if there is no augmenting path with respect to $Z$.*

Since the notion of strong spanning trees cannot be applied to every bipartite graph [3], an auxiliary bipartite graph associated with $G$ is considered.

Given a bipartite graph $G = (M, N, E)$, the auxiliary graph $G^a$ associated with $G$ is the bipartite graph $G^a = (M^a, N, E^a)$, where $M^a = M \cup \{0\}$ and $E^a = E \cup \{(0, j) : j \in N\}$.

Every spanning tree $T$ of $G^a$ will be considered to be a tree rooted at row node 0. This induces a direction on all edges on paths $\mu_{0s} \subset T$ which are directed away from the root 0 toward the (row or column) node $s$. If $l'$ and $l''$ are two nodes joined by an edge of $T$, which is directed from $l'$ to $l''$, then $l'$ is called the *predecessor* of $l''$ in $T$ and $l''$ is a *successor* of $l'$ in $T$. Further, such an edge is called *odd* if $l' \in M$ and $l'' \in N$ and *even* if $l'' \in M$ and $l' \in N$. Furthermore, a node or an edge is said to be *higher* than a second node or edge in $T$ (and the second is *lower*) if the first is on the path that joins the root to the second one. For every (row or column) node $s, T(s)$ denotes the subtree of $T$ which contains $s$ and all lower nodes and edges (than $s$) of $T$ (i.e., node $s$ is the root of $T(s)$), and $M^s$ and $N^s$ denote the sets of row and column nodes, respectively, that belong to $T(s)$. If $t \in T(s)$, the *distance* from $s$ to $t$ is the number of edges of the (unique) path $\mu_{st}$ contained in $T$.

A spanning tree $T$ of $G^a$ is a *strong spanning tree* (s.s.t.) if every column node of degree 1 in $T$ is joined to the root by an (auxiliary) edge (i.e., terminal nodes in $T$ whose distance to the root is greater than 1 are all row nodes).

Given an s.s.t. $T$ of $G^a, Q(T)$ and $C(T)$ denote the sets of column nodes of degree 1 and greater than 2 in $T$, respectively. Furthermore, $D(T)$ denotes the subset of $C(T)$ which is highest, i.e., $h \in D(T)$ if $h \in C(T)$ and column nodes in $\mu_{0h}$ different from $h$ are all of degree 2 in $T$. $M^D$ and $N^D$ denote the set of row and column nodes that belong to the subtrees of $T$ rooted at the column nodes $h \in D(T)$, i.e., $M^D = \cup\{M^h : h \in D(T)\}$ and $N^D = \cup\{N^h : h \in D(T)\}$.

An s.s.t. $T$ of $G^a$ is called *closed* if either $Q(T) = \phi$ or $C(T) = \phi$ or if $i \in M^D$ and $(i, j) \in G$ implies $j \in N^D$.

Let $T$ be an s.s.t. of $G^a$. For each column node $j \notin Q(T)$ (i.e., $j$ is of at least degree 2 in $T$), let $s(j)$ be some successor of $j$ in $T$. The set $Z(T) = \cup\{(s(j), j) : j \notin Q(T)\}$

is a matching of $G$, called the *matching induced* by $T$. Note that if $Z'$ and $Z''$ are two matchings induced by an s.s.t. $T$ of $G^a$, then $|Z'| = |Z''|$.

Given a spanning tree $T$ of $G^a$, a *pivot* in $T$ consists of deleting an edge of $T$ (called the *leaving edge*) and replacing it by an edge of $G - T$ (called the *entering edge*) to form a new spanning tree of $G^a$. A pivot in $T$ is said to be *even* (respectively, odd) if the leaving edge is even (respectively, odd). Furthermore, if the leaving edge joins the nodes $l'$ and $l''$, and $l''$ is a successor of $l'$ in $T$, then the deletion of such an edge cuts $T$ in two distinct subtrees: $T(l'')$ containing the node $l''$ and $T-T(l'')$ containing the node $l'$. The entering edge is chosen from the set $B(l', l'') = \{(i,j) : i \in M^{l''}, j \in N - N^{l''}\}$. If $(f,g)$ denotes the entering edge, $P = P((l',l''),(f,g))$ denotes the corresponding pivot and $T' = T * P = (T - (l',l'')) \cup (f,g)$ is the resulting spanning tree.

LEMMA 2 [3]. *Let $T$ be an s.s.t. of $G^a$ and $h \in C(T)$. If $(l,h)$ is odd (even) edge in $T$ and $B(l,h) \neq \phi$, then $T' = (T - (l,h)) \cup (f,g)$ is an s.s.t. of $G^a$ for any $(f,g) \in B(l,h)$ and $|Z(T')| = |Z(T)|$ if $g \notin Q(T)$ and $|Z(T')| = |Z(T)|+1$ if $g \in Q(T)$, for any $Z(T)$ and $Z(T')$ matchings induced by $T$ and $T'$, respectively.*

The following is the characterization of maximum matchings in bipartite graphs using strong spanning trees.

THEOREM 2 [3]. *If $T$ is a closed s.s.t. of $G^a$, then every matching induced by $T$ is a maximum matching of $G$. Moreover, every $G^a$ has a closed s.s.t. $T$. Reciprocally, if $Z$ is any maximum matching of $G$, then there exists a closed s.s.t. $T$ of $G^a$ with $Z$ a matching induced by $T$.*

The algorithm described in [2] ensures the existence of a closed s.s.t. for every $G^a$. It generates a finite sequence of s.s.t. of $G^a, T^0, T^1, \ldots, T^r$, where $T^k$ is obtained by pivoting on some odd edge of $T^{k-1}, k = 1, \ldots, r$, and $T^r$ is closed.

**3. Strong block pivots in strong spanning trees.** The implementation of the algorithm proposed in [3] uses the idea of a "block pivot" (as suggested in [6]). This permits the description of the algorithm as one consisting of at most $|N| - 1$ stages. Each stage starts with the s.s.t. obtained at the end of the preceding stage and generates a finite sequence of (odd) pivots which terminates with a pivot whose entering edge is incident with a column node of degree 1 in that tree. At the end of the stage, a block pivot is performed to update the tree. The work required in each stage is $0(|E|)$, implying that the algorithm requires a work of $0(|V||E|)$ to find a maximum matching in the bipartite $G = (M, N, E)$. To make this idea more efficient some modification is needed. The following, as we prove later on, achieves this requirement.

Let $T$ be an s.s.t. of $G^a$. A *strong block pivot* (s.b.p.) in $T$ is a finite sequence of pivots in $T, P((l_1,h_1),(f_1,h_2)), P((l_2,h_2),(f_2,h_3)), \ldots, P((l_{k-1},h_{k-1}),(f_{k-1},h_k))$ that satisfies the following conditions:

(a) $(l_1, h_1)$ is an even edge in $T$ with column node $h_1$ of degree greater than 2 in $T$ and $f_1 \in T(l_1)$;

(b) $(l_i, h_i)$ is an odd edge in $T$ with column node $h_i$ of degree 2 in $T$ and $f_i \in T(h_i)$, for $i = 2, \ldots, k-1$;

(c) column node $h_k$ is of degree 1 in $T$ and $T' = T*P_{k-1}*P_{k-2}*\cdots*P_t$ is a spanning tree of $G^a$, for $t = 1, \ldots, k-1$, where $P_i = P((l_i,h_i),(f_i,h_{i+1})), i = 1, \ldots, k-1$.

The requirement that the first leaving edge be an even edge in $T$ is crucial to efficiency, since $T(l_1) \subseteq T(h_1)$ implies that pivoting on $(l_1, h_1)$ takes less work than pivoting on the odd edge $(l_0, h_1) \in T$, with $l_0$ being the predecessor of $h_1$ in $T$. In addition, and more importantly, the choice of an even edge (as we will prove later) allows us to construct a certain set of s.b.p.s associated with $T$ (the equivalent of the particular set of augmenting paths found in each stage of the Hopcroft–Karp algorithm), which is not always possible if the leaving edge is chosen as an odd edge (as in the Balinski–González algorithm).

To simplify notation (in some cases), $P = P_1 \cup P_2 \cup \cdots \cup P_{k-1}$ and $T' = T * P$ will denote, respectively, an s.b.p. in $T$ and the resulting spanning tree of $G^a$ obtained after $P$ is done in $T$.

Two s.s.t.'s $T$ and $T'$ of $G^a$ are called *equivalent* if $|Z(T)| = |Z(T')|$ for any $Z(T)$ and $Z(T')$ matchings induced by $T$ and $T'$, respectively.

LEMMA 3. *Let $T$ be an s.s.t. of $G^a$. If $P$ is an s.b.p. in $T$, then $T' = T*P$ is an s.s.t. of $G^a$, and $|Z(T')| = |Z(T)| + 1$ for any $Z(T), Z(T')$ matchings induced by $T$ and $T'$, respectively.*

*Proof.* The proof follows directly from the definition of strong block pivots. □

THEOREM 3. *Let $T$ be an s.s.t. of $G^a$. If $T$ does not admit strong block pivots, then every matching induced by $T$ is a maximum matching of $G$.*

*Proof.* We prove that $T$ is equivalent to a closed s.s.t. $T^*$ of $G^a$. In such a case, Theorem 2 and the definition of equivalence between trees imply the result.

Suppose that $T$ is not closed (otherwise $T^* = T$). Then, for some $h \in D(T)$, there exists an edge $(f, g)$ with $f \in T(h)$ and $g \notin N^D$. Since $h \in D(T), g$ does not belong to $Q(T)$, otherwise $P = P((l', h), (f, g))$ would be an s.b.p. in $T$, where $(l', h)$ is the even edge that belongs to the unique path in $T$ that joins $h$ with $f$, contradicting the hypothesis. Therefore, column node $g$ is of degree 2 in $T$. If $l''$ is the predecessor of $g$ in $T$, pivot on $(l'', g)$ in $T$, using $(f, g)$ as the entering edge, to obtain a new s.s.t. $T'$ which is equivalent with $T$. By repeating this argument at most $|N| - |Q(T)| - |D(T)|$ times (since no more than this number of pivots can be done), a closed s.s.t. $T^*$ equivalent with $T$ is obtained. □

Note that these results are similar to the ones given in Lemma 1 and Theorem 1.

**4. The algorithm of Hopcroft and Karp.** The following are the main results on which the Hopcroft–Karp algorithm is based.

THEOREM 4 [8]. *Let $Z'$ and $Z''$ be two matchings of $G$. If $r = |Z'|, s = |Z''|$, and $s > r$, then $Z' \Delta Z'' = (Z' - Z'') \cup (Z'' - Z')$ contains at least $s - r$ node-disjoint augmenting paths relative to $Z'$. Moreover, if $Z''$ is a maximum matching of $G$, then there exists an augmenting path relative to $Z'$ whose length is at most $2\lfloor r/(s-r)\rfloor + 1$.*

Given a matching $Z$ of $G$, an augmenting path $\mu$ relative to $Z$ is called *shortest* if $|\mu| \leq |\mu'|$ for any augmenting path $\mu'$ relative to $Z$.

THEOREM 5 [8]. *Let $Z$ be a matching of $G$ and $\mu$ a shortest augmenting path relative to $Z$. If $\mu'$ is any augmenting path relative to $Z' = Z\Delta\mu$, then $|\mu'| \geq |\mu| + |\mu \cap \mu'|$.*

By using these results the general procedure that finds a maximum matching in $G$ can be modified as follows. Starting with any matching $Z_0$ (e.g., $Z_0 = \phi$), construct a sequence of matchings $Z_1, Z_2, \ldots$, defined by $Z_{i+1} = Z_i \Delta \mu_i$, where $\mu_i$ is a shortest augmenting path relative to $Z_i$. Moreover, these results imply that the computation of the sequence $\{Z_i\}$ can be performed in at most $2\lfloor\sqrt{|N|}\rfloor + 2$ phases. Within each phase, all the segmenting paths found are node disjoint and have the same (shortest) length. All the augmenting paths are relative to the matching from the beginning of the phase.

In short, the basic steps of the algorithm are as follows:

*Step* 0. Start with $Z = \phi$.
*Step* 1. Let $l(Z)$ be the length of a shortest augmenting path relative to the current matching $Z$. Find a maximal set of paths $\{\mu_1, \mu_2, \ldots, \mu_t\}$ such that
(a) $\mu_i$ is an augmenting path relative to $Z$ and $|\mu_i| = l(Z)$, for $i = 1, \ldots, t$, and

(b) the $\mu_i$ are node disjoint. If no such paths exist, stop. The current matching $Z$ is maximum. Otherwise, go to Step 2.

*Step* 2. Construct $Z' = Z\Delta\mu_1\Delta\mu_2\Delta\dots\Delta\mu_t$. Set $Z = Z'$ and go to Step 1.

The construction of the set $\{\mu_1, \mu_2, \dots, \mu_t\}$ in Step 1 is replaced by one that obtains a maximal set of node-disjoint paths in a particular graph associated with $G$ using a depth-first search. The work required to perform such a construction is $0(|E|)$, which implies the $0(\sqrt{|N|}|E|)$ bound for the whole algorithm.

**5. The new competitive algorithm.** In this section we show that the concept of an s.s.t. is related to the notion of augmenting path. This allows us to obtain results similar to those given in Theorems 4 and 5. To this end, some additional definitions are needed.

Let $T$ be an s.s.t. of $G^a$ and $P = \{P((l_i,h_i),(f_i,h_{i+1})) : i = 1,\dots,k-1\}$ an s.b.p. in $T$. The definition of an s.b.p. implies that the (unique) paths in $T, \mu(l_1,f_1)$, $\mu(h_i,f_i), i = 2,\dots,k-1$, together with the entering edges $(f_i,h_{i+1})$, define the path $\mu(P) = \mu(l_1,f_1)\cup(f_1,h_2)\cup\mu(h_2,f_2)\cup\cdots\cup\mu(h_{k-1},f_{k-1})\cup(f_{k-1},h_k)$ (contained in $G$). The *size* of $P$, denoted by $s(P)$, is defined by the length of $\mu(P)$, i.e., $s(P) = |\mu(P)|$.

A given s.b.p. $P$ in $T$ is said to be of *smallest* size if $s(P) \le s(P')$ for any s.b.p. $P'$ in $T$.

Two s.b.p. $P$ and $P'$ in an s.s.t. $T$ of $G^a$ are said to be *disjoint* if the paths $\mu(P)$ and $\mu(P')$ are node disjoint.

THEOREM 6. *Let $T$ be an s.s.t. of $G^a$ and $Z(T)$ a matching induced by $T$. If $\mu$ is an augmenting path relative to $Z(T)$, then $T$ admits an s.b.p. $\{P((l_i,h_i),(f_i,h_{i+1})) : i = 1,\dots,k-1\}$ such that $\mu$ contains the paths $\mu(l_1,f_1),\mu(h_i,f_i)$ and the entering edges $(f_i,h_{i+1}), i = 1,\dots,k-1$, i.e., $\mu \supset \mu(P)$.*

*Proof.* Let $\mu = \{(f,j_1),(i_1,j_1),(i_1,j_2),\dots,(i_{p-1},j_{p-1}),(i_{p-1},g)\}$ be an augmenting path in $G$, relative to $Z(T)$, whose edges satisfy $(i_l,j_l) \in Z(T), (i_{l-1},j_l) \notin Z(T)$, for $l = 1,\dots,p$, where $i_0 = f \in M$ and $j_p = g \in N$ are its endpoints. The definition of $Z(T)$ implies that $g \in Q(T)$, that the predecessor of $f$ in $T$, say column node $h$, is of degree greater than 2 in $T$ (because row node $f$ is free), and that $j_s$ is the predecessor of $i_s$ in $T$ for $s = 1,\dots,p-1$ (because $(i_s,j_s) \in Z(T)$).

Let $\mu'$ and $\mu''$ be the subsets of $\mu$ defined as follows. For each edge $(i_s,j_{s+1})$ in $\mu$, let $q_s$ be the predecessor of $i_s$ in $T$ and $\mu' = \{(i_s,j_{s+1}) : q_s$ is of degree greater than 2 in $T\}$. Furthermore, let $\mu'' = \{(i_s,j_{s+1}) : (i_s,j_{s+1}) \in \mu - T\}$. These two sets are nonempty because $(i_0,j_1) = (f,j_1) \in \mu'$ and $(i_{p-1},j_p) = (i_{p-1},g) \in \mu''$ (since $g \in Q(T)$ and $(i_{p-1},g) \notin Z(T)$). Therefore, if $t = \max\{s : (i_s,j_{s+1}) \in \mu'\}$ and $\bar{\mu}$ denotes the subset of $\mu''$ containing the edges $(i_s,j_{s+1})$, with $s > t$ ($\bar{\mu} \ne \phi$ because $(i_{p-1},g) \in \bar{\mu}$), then a strong block pivot $P$ in $T$ is defined by the even edge $(i_t,q_t)$, with which $P$ starts, and the edges in $\bar{\mu}$, which define the entering edges of $P$. □

COROLLARY 1. *Let $T$ be an s.s.t. of $G^a$ and $Z(T)$ a matching induced by $T$. If $P$ is an s.b.p. of minimum size in $T$, then $\mu(P)$ is a shortest augmenting path relative to $Z(T)$. Reciprocally, if $\mu$ is any shortest augmenting path relative to $Z(T)$, then $T$ admits an s.b.p. $P$ such that $\mu = \mu(P)$.*

*Proof.* Let $P$ be an s.b.p. in $T$ of minimum size and suppose that there exists an augmenting path $\mu$ relative to $Z(T)$ satisfying $|\mu| < |\mu(P)|$. By Theorem 6, $\mu$ determines an s.b.p., say $P'$, whose size is at most equal to $|\mu|$. Therefore, $s(P') \le |\mu| < |\mu(P)| = s(P)$, contradicting the hypothesis on $P$.

The second part follows directly from Theorem 6, since the s.b.p. $P$ determined by $\mu$ satisfies $\mu(P) \subset \mu$, and the condition on $\mu$ of being a shortest path implies $\mu(P) = \mu$. □

THEOREM 7. *Let $T$ and $T'$ be two s.s.t.'s of $G^a$ and $Z(T), Z(T')$ matchings induced by $T$ and $T'$, respectively. If $r = |Z(T)|, s = |Z(T')|$, and $r < s$, then $T$ admits (at least) $s - r$ disjoint strong block pivots.*

*Proof.* By Theorem 4, it follows that $Z(T)\Delta Z(T')$ contains (at least) $p = s - r$ node-disjoint augmenting paths $\mu_1, \mu_2, \ldots, \mu_p$ relative to $Z(T)$. Theorem 6 implies that each $\mu_k$ determines an s.b.p. $P_k$ which satisfies $\mu(P_k) \subset \mu_k$. This implies that $\mu(P_k)$ are node disjoint, and Theorem 7 follows. □

COROLLARY 2. *Let $T$ be an s.s.t. of $G^a$ and $Z(T)$ a matching induced by $T$. Let $r = |Z(T)|$ and $s$ denote the cardinality of a maximum matching of $G$. Then $T$ admits an s.b.p. of size (at most) $2\lfloor r/(s-r)\rfloor$.*

*Proof.* Since $s > r$, Theorem 7 implies that $T$ admits $s - r$ s.b.p. which contain (at most) the $r$ edges of $Z(T)$. Therefore one of them contains (at most) $\lfloor r/(s - r)\rfloor$ edges of $Z(T)$, and by definition, the size of such an s.b.p. is at most $2\lfloor r/(s - r)\rfloor$. □

THEOREM 8. *Let $T$ be an s.s.t. of $G^a$ and $P$ an s.b.p. of minimum size in $T$. If $T' = T * P$ and $P'$ is any s.b.p. in $T'$, then $s(P') \geq s(P) + 2|\mu(P) \cap \mu(P')|$.*

*Proof.* Let $P = \{P((l_i, h_i), (f_i, h_{i+1})) : i = 1, \ldots, k-1\}$ be an s.b.p. of minimum size in $T$ and $T' = T * P$. From the definition of s.b.p. it follows that $d'(h_i) = 2$ for $i = 2, \ldots, k$ in $T'$ and $d'(h_1) \geq 2$ in $T'$. Furthermore, $f_i$ is the successor of $h_{i+1}$ in $T'$ and every odd (respectively, even) edge of $\mu(h_i, f_i)$ in $T$ is an even (respectively, odd) edge in $T'$. Moreover, the definition of $\mu(P)$ implies that the distance between $h_k$ and $l_1$ in $T'$ is $|\mu'(h_k, l_1)| = |\mu(P)|$.

Let $P' = \{P((l'_i, h'_i), (f'_i, h'_{i+1})) : i = 1, \ldots, t-1\}$ be an s.b.p. in $T'$. If $\mu(P) \cap \mu(P') = \phi$, then $P'$ is also an s.b.p. in $T$, and hence $s(P') \geq s(P)$.

Now assume that $\mu(P) \cap \mu(P') \neq \phi$. The definition of $T'$ implies that $\mu(P) \cap \mu(P')$ is contained in $\mu'(h_k, l_1)$ and that both the highest and lowest edge in $\mu'(h_k, l_1)$ which belong to $\mu(P) \cap \mu(P')$ are even. Moreover, the first edge of $P'$ (starting from $(l'_1, h'_1)$ toward $(f'_{t-1}, h'_t)$) that belongs to $\mu(P) \cap \mu(P')$ is an even edge in $T'$. This property is also satisfied by the last edge of $P'$ that belongs to $\mu(P) \cap \mu(P')$. Therefore, if $(f'_r, h'_{r+1})$ and $(f'_s, h'_{s+1})$ denote such edges, then $P'$ can be partitioned as follows. $P' : P'_1 \cup P'_2 \cup P'_3$, where $P'_1 = \{P((l'_i, h'_i), (f'_i, h'_{i+1})) : i = 1, \ldots, r\}$ and $P'_3 = \{P((l'_i, h'_i), (f'_i, h'_{i+1})) : i = s+1, \ldots, t-1\}$ satisfy $P'_1 \cap \mu(P) = P'_3 \cap \mu(P) = \phi$, and $\mu(P) \cap \mu(P') \subset P'_2$. Furthermore, if $q'_j = |P'_j \cap \mu(P')|$ for $j = 1, 2, 3, t'_1 = |\mu'(f'_s, l_1)|, t'_2 = |\mu'(h'_{r+1}, f_s)|, t'_3 = |\mu'(h_k, h_{r+1})|, P_1 = \{P((l_i, h_i), (f_i, h_{i+1})) : i = 1, \ldots, s-1\}$, and $P_3 = \{P((l_i, h_i), (f_i, h_{i+1})) : i = r+1, \ldots, k-1\}$, then the following hold:

(1) $s(P) = t'_1 + t'_2 + t'_3$, because $\mu'(h_k, l_1) = \mu'(h_k, h_{r+1}) \cup \mu'(h_{r+1}, f'_s) \cup \mu'(f'_s, l_1)$ and, by definition, $s(P) = |\mu'(h_k, l_1)|$.

(2) $q'_1 + t'_3 \geq s(P)$, because $P'_1 \cup P_3$ is an s.b.p. in $T$. Hence, using (1) it follows that $q'_1 \geq t'_1 + t'_2$.

(3) $t'_1 + q'_3 \geq s(P)$, because $P_1 \cup P((l_s, h_s), (f_s, h'_s + 1)) \cup P'_3$ is an s.b.p. in $T$. Therefore, $q'_3 \geq t'_2 + t'_3$.

(4) $t'_1 + q'_2 + t'_3 \geq s(P)$, because $P_1 \cup P'_2 \cup P_3$ is an s.b.p. in $T$. (1) implies that $q'_2 \geq t'_2$.

These relations imply that $s(P') = q'_1 + q'_2 + q'_3 \geq t'_1 + 3t'_2 + t'_3 \geq s(P) + 2|\mu(P) \cap \mu(P')|$ since $t'_2 \geq |\mu(P) \cap \mu(P')|$, completing the proof. □

The above results allow us to give the following algorithm to construct maximum matchings in bipartite graphs.

The basic steps are as follows:

*Step* 0. Start with an s.s.t. $T$ of $G^a$.
*Step* 1. Let $s(P)$ be the size of an s.b.p. $P$ of minimum size in the current tree $T$. Find a maximal set of s.b.p. $\{P_1, \dots, P_t\}$ such that
(a) $s(P_i) = s(P)$, for $i = 1, \dots, t$, and
(b) the $P_i$ are disjoint. If no such set exists, stop. Every matching induced by $T$ is a maximum matching of $G$. Otherwise, go to Step 2.
*Step* 2. Obtain $T' = T * P_1 * P_2 * \cdots * P_t$. Set $T = T'$, and go to Step 1.

We now, consider the complexity of the algorithm. Recall that $G = (M, N, E)$ with $|M| \geq |N|$.

In the implementation, a standard adjacency list is used to represent the bipartite graph, and each tree is represented by some known data structure (e.g., the "three label" given in [9]).

Step 0 requires an amount of work of $0(|M| + |N|)$. The above results imply (due to an argument similar to that of Hopcroft and Karp) that Step 1 (and thus Step 2) is performed at most $0(\sqrt{|N|})$ times. Thus, since Step 2 can be done in $0(|M| + |N|)$ work (the noted data structure for trees allows subsets of nodes to be obtained in work linear in their cardinalities), the complexity of the whole algorithm depends on the work required to perform Step 1. However, by following the method by which Hopcroft and Karp solve the corresponding step of their algorithm, we can also obtain a maximal set of node-disjoint paths in a particular graph $\bar{G} = (\bar{V}, \bar{E})$ associated with $G$ and use such a maximal set of paths to determine the maximal set of s.b.p.

Let $N_0 = C(T), M_1 = \{i \in M : i \text{ is a successor of } j \text{ in } T, j \in N_0\}, N_1 = \{j \in N : (i,j) \in E, i \in M_1, j \in N - N_0\}$, and $E_1 = \{(i,j) : i \in M_1, j \in N_1\}$. (Note that every column node in $N_1$ is of degree 2 in $T$.)

Suppose that for $t = 1, \dots, k$ the sets $M_t, N_t$, and $E_t$ have been obtained. Then $M_{k+1}, N_{k+1}$, and $E_{k+1}$ are defined by $M_{k+1} = \{i : i \text{ is the successor of } j \text{ in } T, j \in N_k\}, N_{k+1} = \{j : (i,j) \in E, i \in M_{k+1}, j \in N - \bar{N}_k\}$, where $\bar{N}_k = \cup\{N_t : t = 0, 1, \dots, k\}$, and $E_{k+1} = \{(i,j) : i \in M_{k+1}, j \in N_{k+1}\}$.

If $k^*$ denotes the least value of the index $k$ for which $N_k \cap Q(T) \neq \phi$, then $\bar{V} = \cup\{M_t \cup N_t : t = 1, \dots, k^*\}$ and $\bar{E} = \cup\{E_t : t = 1, \dots, k^*\}$.

The graph $\bar{G} = (\bar{V}, \bar{E})$, thus defined, can be considered as a layered (directed and acyclic) graph, since $\bar{E} \subset \cup\{M_t \times N_t : t = 1, \dots, k^*\}$ and each node has either in-degree or out-degree equal to 1.

The construction of $\bar{G}$ takes $0(|E|)$ time because each edge in $G$ is processed at most once to become an arc in $\bar{G}$. Furthermore, this construction implies that the strong block pivots of minimum size in $T$ are in one-to-one correspondence with the paths of $\bar{G}$ that begin at nodes in $M_1$ and end at nodes in $N_{k^*} \cap Q(T)$. Moreover, since a maximal set of node-disjoint paths in $\bar{G}$ can be obtained with an amount of work of $0(|\bar{E}|)$ (by applying algorithm $B$ given in [8]), we conclude that Step 1 of the above algorithm can be executed in at most $0(|E|)$ work, which implies the following result.

THEOREM 9. *The algorithm requires* $0(|\sqrt{|V|}|E|)$ *work to find a maximum matching in a bipartite graph with* $|V|$ *nodes and* $|E|$ *edges.*

Even though the above results are similar to those of Hopcroft and Karp, the approach based on s.s.t.'s is different from the one based on augmenting paths because the progress of the present algorithm (from one iteration to the next) does not use

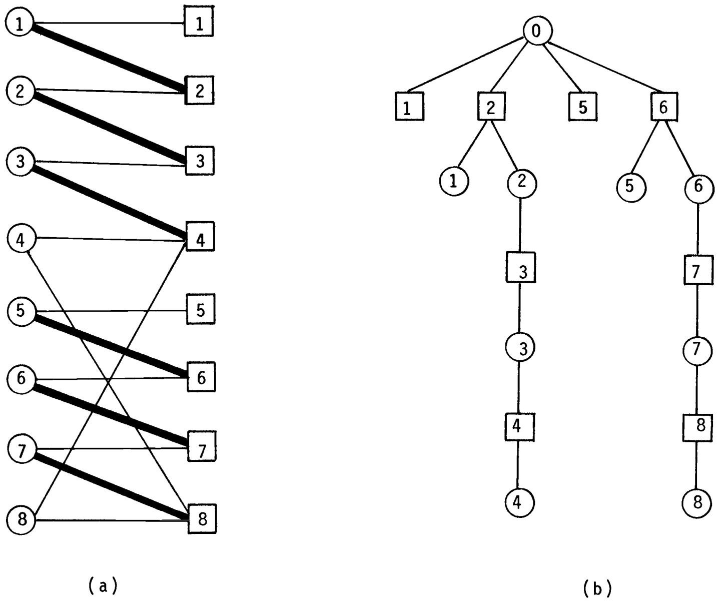


FIG. 1.

any matching of $G$ associated with the current s.s.t., and therefore it does not use (explicitly) augmenting paths. In fact, the main object of this approach is to maintain the structure of the s.s.t. However, since s.s.t.'s do induce some matchings, the above results allow us to show the relation between the concept of s.s.t.'s and augmenting paths.

We end this work with an example to illustrate that the two approaches are different.

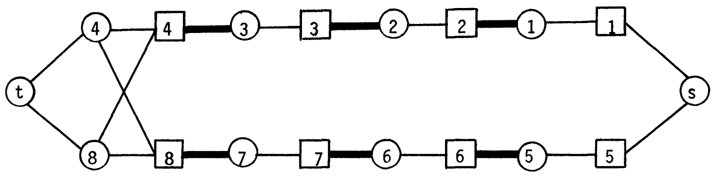
Figure 1(a) shows a bipartite graph $G$ with a matching $Z$ (defined by the heavy edges), and Fig. 1(b) shows a given s.s.t. $T$ of $G^a$.

Figures 2(a) and 2(b) illustrate the auxiliary graphs obtained during Step 1 of the Hopcroft–Karp algorithm and our algorithm, respectively.

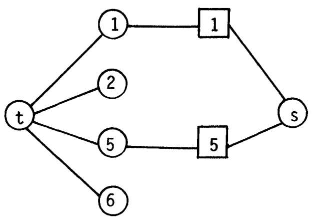
The paths $\mu' = \{(1,1)\}$ and $\mu'' = \{(5,5)\}$ define a maximal set of node-disjoint paths of the graph $\bar{G}$ given in Fig. 2(b), and determine a maximal set of trivial node-disjoint s.b.p.'s $\{P', P''\}$ associated with the s.s.t. $T$ of the Fig. 1(b), where $P' = \{P((1,2),(1,1))\}$ and $P'' = \{P((5,6),(5,5))\}$. Figure 3 illustrates the resulting s.s.t. $T' = T{*}P'{*}P''$ after pivoting in $T((1,1)$ and $(5,5)$ are the entering edges, and the even edges $(1,2)$ and $(5,6)$ in $T$ are the leaving ones).

Note that $Q(T') = \phi$, and therefore $T'$ is closed (the matching induced by $T'$ is a maximum matching of the graph $G$ given in Fig. 1(a)). Furthermore, since $T' = (T - \{(1,2),(5,6)\}) \cup \{(1,1),(5,5)\}$, the new tree $T'$ differs from $T$ only in the leaving and entering edges. In the general case, if $\{P_1, \dots, P_t\}$ is the set of s.b.p.'s determined in Step 1 of the algorithm, then the resulting tree $T' = T * P_1 * \cdots * P_t$ will have $|\mu(P_1)| + \cdots + |\mu(P_t)|$ new edges because the edges in $\mu(P_i)$ that were odd (even) in $T$ become even (odd) in $T'$.

The difference shown by this example is due to the following reason. In the augmenting path approach, the free (row and column) nodes are fixed by the current matching, whereas in the s.s.t. approach (using the matching jargon), column nodes in $Q(T)$ are free and any successor of column nodes in $C(T)$ can be considered as a



(a)



(b)

FIG. 2.

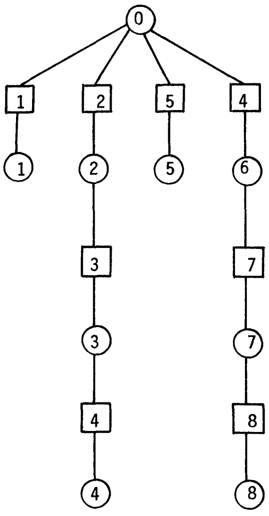


FIG. 3.

free node (but no one of them is fixed to be free) because of the structure of the strong spanning trees.

**Acknowledgments.** The authors would like to thank the referees for their suggestions for improving the presentation of this paper.

## REFERENCES

[1] H. Alt, N. Blum, K. Mehlhorn, and M. Paul, *Computing a maximum cardinality matching in a bipartite graph in time* $0(n^{1.5}\sqrt{m/\log n})$, Inform. Process. Lett., 37 (1991), pp. 237–240.

[2] M. L. Balinski, *A competitive (dual) simplex method for the assignment problem*, Math. Programming, 34 (1986), pp. 125–141.

[3] M. L. Balinski and J. González, *Maximum matching in bipartite graphs via strong spanning trees*, Networks, 21 (1991), pp. 165–179.

[4] C. Berge, *Two theorems in graph theory*, Proc. Natl. Acad. Sci. U.S.A., 43 (1957), pp. 842–844.

[5] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[6] D. Goldfarb, *Efficient dual simplex algorithms for the assignment problem*, Math. Programming, 33 (1986), pp. 187–203.

[7] M. Hall, Jr., *An algorithm for distinct representatives*, Amer. Math. Monthly, 63 (1956), pp. 716–717.

[8] J. E. Hopcroft and R. M. Karp, *An* $n^{5/2}$ *algorithm for maximum matching in bipartite graphs*, SIAM J. Comput., 2 (1973), pp. 225–231.

[9] E. L. Johnson, *Networks and basic solutions*, Oper. Res., 14 (1966), pp. 619–623.

[10] H. W. Kuhn, *The Hungarian method for the assignment problem*, Naval Res. Logist., 2 (1955), pp. 83–97.

[11] R. Z. Norman and M. O. Rabin, *An algorithm for a minimum cover of a graph*, Proc. Amer. Math. Soc., 10 (1959), pp. 315–319.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 196–207, May 1995

# BOUNDS FOR BINARY CODES THAT ARE MULTIPLE COVERINGS OF THE FARTHEST-OFF POINTS *

HEIKKI O. HÄMÄLÄINEN[†], IIRO S. HONKALA[‡], SIMON N. LITSYN[§], AND PATRIC R. J. ÖSTERGÅRD[¶]

**Abstract.** A binary code $C \subseteq \mathbb{F}_2^n$ with $M$ codewords is called an $(n, M, r, \mu)$ multiple covering of the farthest-off points (MCF) if the Hamming spheres of radius $r$ centered at the codewords cover the whole space $\mathbb{F}_2^n$ and every $x \in \mathbb{F}_2^n$ such that $d(x, C) = r$ is covered by at least $\mu$ codewords. The minimum possible cardinality $F(n, r, \mu)$ of such a code is studied and tables of upper bounds on $F(n, r, \mu)$ for $n \leq 16, r \leq 4, \mu \leq 4$ are given.

**Key words.** binary code, covering radius, multiple covering, football pool problem

**AMS subject classifications.** 94B75, 05B40

**1. Introduction.** Denote by $\mathbb{F}_2 = \{0, 1\}$ the field of two elements. The Hamming distance $d(x, y)$ between two words $x, y \in \mathbb{F}_2^n$ is the number of coordinate places in which $x$ and $y$ differ. The weight wt(x) of $x \in \mathbb{F}_2^n$ is the number of 1's in $x$. Denote $B_r(x) = \{y \in \mathbb{F}_2^n | d(y, x) \leq r\}$.

We define that a binary code $C \subseteq \mathbb{F}_2^n$ is an $(n, M, r, \mu)$ multiple covering of the farthest-off point (MCF) if $C$ has $M$ codewords, its covering radius, defined by

$$\mathrm{CR}(C) = \max_{x \in \mathbb{F}_2^n} \min_{c \in C} d(x, c),$$

is at most $r$, and

$$|B_r(x) \cap C| \geq \mu \quad \text{whenever } d(x, C) = r.$$

Given $r$, we say that $y \in \mathbb{F}_2^n$ is covered by $c \in C$ if $d(y, c) \leq r$ and that $y$ is covered by $C$ if $y$ is covered by some $c \in C$. According to the previous definition, the covering radius of an $(n, M, r, \mu)$ MCF $C$ is at most $r$ and every $x$ for which $d(x, C) = r$ is covered by at least $\mu$ codewords of $C$. That is why we call $C$ an MCF, bearing in mind, however, that since the covering radius of $C$ is *at most* $r$, it is actually possible that there are no points $x \in \mathbb{F}_2^n$ whose distance to $C$ is as large as $r$. We denote the minimum possible cardinality of an $(n, \cdot, r, \mu)$ MCF by $F(n, r, \mu)$.

The problem of studying MCFs is closely related to studying multiple coverings. An $(n, M, r, \mu)$ multiple covering (MC) is a code $C \subseteq \mathbb{F}_2^n$ with $M$ codewords such that *every* $x \in \mathbb{F}_2^n$ is covered by at least $\mu$ codewords, i.e., $|B_r(x) \cap C| \geq \mu$ for every $x \in \mathbb{F}_2^n$. The minimum cardinality of an $(n, \cdot, r, \mu)$ MC is denoted by $K(n, r, \mu)$. This problem

* Received by the editors July 14, 1993; accepted for publication (in revised form) February 16, 1994.

[†] Lehtotie 28, 41120 Puuppola, Finland. The research of this author was supported by the Finnish Academy of Sciences.

[‡] Department of Mathematics, University of Turku, 20500 Turku 50, Finland (honkala@sara.cc.utu.fi).

[§] Department of Electrical Engineering–Systems, Tel–Aviv University, Ramat–Aviv, 69978, Israel (litsyn@genius.tau.ac.il). The research of this author was partially supported by a grant from the Ministry of Science and Technology of Israel.

[¶] Department of Computer Science, Helsinki University of Technology, 02150 Espoo, Finland (patric.ostergard@hut.fi).

and bounds on $K(n,r,\mu)$ have been studied, e.g., in a companion paper [9] to this paper, where a table on $K(n,r,\mu)$ for $n \le 16, r \le 4, \mu \le 4$ and further references can be found.

MCFs form perhaps the simplest subclass of the weighted coverings (see [4]–[6]), for which all the weights are not equal. Weighted coverings have been studied in connection with some problems in information theory. Another motivation for studying these problems arises from the well-known combinatorial (generalized) football pool problem (see e.g. [11], [10], and [17]). Assume that we have $n = t + b$ matches, in $t$ of which there are three possible outcomes and in $b$ of which there are only two possible outcomes. We wish to forecast the outcomes of these matches by making $M$ forecasts such that no matter what the outcomes are, there is at least one forecast in which there are at most $r$ incorrect outcomes, i.e., that wins at least an $(r+1)$st prize. Such a set of forecasts is simply a code with covering radius at most $r$. In this context, an MC is a system of forecasts that guarantees at least $\mu$ prizes, each of which is at least the $(r+1)$st prize, whereas an MCF gives us a system of forecasts that wins a prize which is at least the $r$th prize or at least $\mu$ times an $(r+1)$st prize. From a player's point of view, studying MCFs is an even more natural problem, because at least for small values of $\mu$ the $r$th prize will be worth at least $\mu$ times the $(r+1)$st prize. In this paper, we study only the binary case $t = 0$, i.e., in each match the player decides to exclude one of the outcomes, or, equivalently, we have cup matches without draws.

In the definition of an MCF, we have not allowed the possibility that some word appears in the code more than once. However, sometimes we can do better if this is permitted. Suppose $C$ is a collection consisting of $M$ (not necessarily distinct) words in $\mathbb{F}_2^n$, say $c_1, c_2, \ldots, c_M$. We say that these $M$ binary words form an $(n, M, r, \mu)$ MCFR (MCF with repeated words) if for every $x \in \mathbb{F}_2^n$ we have

$$d(x, c_i) \le r - 1 \quad \text{for some } i = 1, 2, \ldots, M,$$

or

$$d(x, c_i) = r \quad \text{for at least } \mu \text{ of the indices } i = 1, 2, \ldots, M.$$

More formally, $C$ is a multiset, i.e., a function from $\mathbb{F}_2^n$ to the set of natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$, and we denote by $C(c)$ the multiplicity of $c$ in $C$. We denote the minimum cardinality of an $(n, \cdot, r, \mu)$ MCFR by $\bar{F}(n,r,\mu)$. An example of a case where $F(n,r,\mu) > \bar{F}(n,r,\mu)$ is given in §2.

In this paper we concentrate almost entirely on constructive upper bounds. In §2 we very briefly discuss some lower bounds. In §3 we discuss various different construction methods, for example, some natural ways of obtaining MCFs from MCs. In §4 we give a table of upper bounds on $F(n,r,\mu)$ for $n \le 16, r \le 4, \mu \le 4$.

We illustrate the different problems discussed above by the following simple example.

*Example* 1. Suppose that there are thirteen football matches, the outcomes of which a player wishes to forecast; the player thinks that he or she knows in advance nine of the outcomes and can exclude one outcome in each of the remaining four matches. He or she wishes to find a suitable set of forecasts that—no matter what the outcomes in the remaining four matches are—ensures a forecast in which there is at most one incorrect result of a game. Then he or she can use the following four forecasts (after choosing a suitable notation for the outcomes):

$$0000, 0111, 1000, 1111.$$

If the player uses each of these four forecasts twice, then he or she is guaranteed to obtain at least two forecasts each with at most one incorrect entry. The same can be achieved even with seven forecasts instead of eight, namely

$$0001, 0010, 0011, 1100, 1100, 0111, 1011,$$

(see [3]). However, by using only the following *six* forecasts,

$$1111, 1111, 1000, 0100, 0010, 0001,$$

the player will always get at least one entirely correct forecast or at least two forecasts with one incorrect entry, as can easily be checked.

This shows that $\bar{F}(4,1,2) \leq 6$. As will be shown later, $\bar{F}(4,1,2) = 6$, but $F(4,1,2) = 7$.

**2. Lower bounds.** According to the sphere covering lower bound for MCFs and MCFRs we have

$$(1) \quad F(n,r,\mu) \geq \bar{F}(n,r,\mu) \geq 2^n \left( \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{r-1} + \frac{1}{\mu}\binom{n}{r} \right)$$

(see [19]). An MCF which attains (1) with equality is called *perfect*.

In the following theorem we give a simple but nontrivial example of a case in which $F(n,r,\mu) > \bar{F}(n,r,\mu)$.

THEOREM 1. $F(4,1,2) = 7 > \bar{F}(4,1,2) = 6$.

*Proof.* By the sphere covering bound we have $\bar{F}(4,1,2) \geq 6$, and it is easy to check that the six words $1111, 1111, 1000, 0100, 0010, 0001$ (see also Example 1 in §1) form a $(4,6,1,2)$ MCFR. Hence $\bar{F}(4,1,2) = 6$.

It is equally easy to check that the seven words $1111, 1100, 0011, 1000, 0100, 0010, 0001$ form a $(4,7,1,2)$ MCF. It therefore remains to show that $F(4,1,2) \geq 7$. Assume on the contrary that there is a $(4,6,1,2)$ MCF. In average the spheres $B_1(x)$ have $6 \cdot 5/16 < 2$ codewords of $C$, hence for some $x \in \mathbb{F}_2^4$, we have $|B_1(x) \cap C| \leq 1$. Then necessarily $x$ itself is a codeword. W.l.o.g. $x = 0000$ and there are no codewords of weight 1. Because of the words of weight 1, the number of codewords of weight 2 is at least 2, i.e., $A_2 \geq 2$. On the other hand, since there are only six codewords, there is a word of weight 2—say 1100—that is not a codeword. But then both the words 1110 and 1101 have to be codewords and $A_2 \leq 3$. If $0011 \notin C$ then also $1011, 0111 \in C$, and $A_2 \leq 1$, a contradiction. Hence $0011 \in C$. By symmetry, we can assume that $1010 \notin C$, but then $1011 \in C$. But this forces $A_2 = 2$, and the codewords of weight 2 should cover all the words of weight 1 once, and therefore the other codeword of weight 2 should be 1100, a contradiction. $\square$

## 3. Constructions.

### 3.1. Adding a parity check bit.

If we have a binary code of length $n$ and covering radius $R$, adding a parity check bit to all the codewords increases both the length and the covering radius by 1, which is disappointing in the sense that simply adding one more identically 0 coordinate would do the same. However, adding a parity check bit does have its advantages, as is shown in the following theorem. We denote the sum (mod 2) of the coordinates of a word $x \in \mathbb{F}_2^n$ by $p(x)$.

THEOREM 2. *Suppose $C$ is an $(n,M,r,\mu)$ MC. Then the extended code $\bar{C} = \{(c,p(c)) \in \mathbb{F}_2^{n+1} | c \in C\}$ is an $(n+1, M, r+1, \lceil \mu(n+1)/(r+1) \rceil)$* MCF.

*Proof.* It is clearly sufficient to show that every point $(x, x') \in \mathbb{F}_2^{n+1}, x \in \mathbb{F}_2^n$, $x' \in \mathbb{F}_2$ that has distance $r+1$ to $\bar{C}$ is covered by at least $\mu(n+1)/(r+1)$ codewords. Clearly, $d(x, C) = r$. Because $(x, x')$ has distance $r+1$ to $\bar{C}$, the word $(x, x')$ has to disagree in the last coordinate with every $(c, p(c)) \in \bar{C}$ for which $d(x, c) = r$, and therefore $x' = p(x) + 1 (\mathrm{mod} 2)$ if $r$ is even and $x' = p(x)$ if $r$ is odd. If $d(x, c) = r+1$ or $r$, then $|B_r(c) \cap B_1(x)| = r+1$, and because $C$ covers $\mu$ times all the points in $B_1(x)$, we have $|B_{r+1}(x) \cap C| \geq \mu(n+1)/(r+1)$. We show that the codewords $(c, p(c))$, where $c \in B_{r+1}(x) \cap C$, will do, i.e., each of them is within distance $r+1$ from $(x, x')$. This is immediate if $d(c, x) = r$. If $d(c, x) = r+1$, then $p(c) = p(x) + 1 = x'$ when $r$ is even and $p(c) = p(x) = x'$ when $r$ is odd, proving our claim. □

COROLLARY 3. *If $C$ is a perfect $(n, M, r, 1)$ MC, then the extended code $\bar{C}$ is a perfect $(n+1, M, r+1, (n+1)/(r+1))$ MCF.*

*Proof.* By Theorem 2, the extended code $\bar{C}$ is an $(n+1, M, r+1, (n+1)/(r+1))$ MCF. Furthermore,

$$\begin{aligned} &M\left(\binom{n+1}{0} + \binom{n+1}{1} + \cdots + \binom{n+1}{r} + \frac{r+1}{n+1}\binom{n+1}{r+1}\right) \\ &= M\left(\binom{n}{0} + \left(\binom{n}{0} + \binom{n}{1}\right) + \cdots + \left(\binom{n}{r-1} + \binom{n}{r}\right) + \binom{n}{r}\right) \\ &= 2 \cdot M\left(\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{r}\right) = 2 \cdot 2^n = 2^{n+1}. \quad \square \end{aligned}$$

*Example* 2. The extended Hamming code is a linear perfect $(2^k, 2^{2^k-1-k}, 2, 2^{k-1})$ MCF.

**3.2. Puncturing.** As we know from the context of covering codes, puncturing a covering code is often useful.

THEOREM 4. *Suppose $C$ is an $(n, M, r, \mu)$ MCF. Then the $M$ words $c' = c(2)c(3) \ldots c(n)$, where $c = c(1)c(2)\ldots c(n) \in C$, form an $(n-1, M, r, 2\mu)$ MCFR $C'$. If no two words of $C$ differ only in the first coordinate, then $C'$ is an MCF.*

*Proof* (cf. [14], proof of Lem. 2). If the Hamming distance from a word $x(2)x(3) \ldots x(n)$ to all the words $c(2)c(3)\ldots c(n)$ is at least $r$, then $d(0x(2)\ldots x(n), C) = r$, $d(1x(2)\ldots x(n), C) = r$, and there are at least $\mu$ different codewords $c_i^0 \in C$ such that $d(c_i^0, 0x(2)\ldots x(n)) = r$ and at least $\mu$ different words $c_i^1 \in C$ such that $d(c_i^1, 1x(2)\ldots x(n)) = r$. Since $d(0x(2)\ldots x(n), C) = r$, all the words $c_i^0$ begin with 0, and similarly the words $c_i^1$ all begin with 1, and consequently we have altogether $2\mu$ different words. Puncturing these words, we see that our claim holds. □

*Example* 3. The $(11, 192, 1, 1)$ MC defined in [7] has the property that no two codewords differ only in the first coordinate. Hence, $F(10, 1, 2) \leq 192$.

**3.3. Piecewise constant codes.** We can also construct MCFs by using the piecewise constant code construction of [7]. By definition a piecewise constant code of length $n = n_1 + n_2 + \cdots + n_i$ consists of all the words $(c_1, c_2, \ldots, c_i)$, where $c_j \in \mathbb{F}_2^{n_j}$, such that $(\mathrm{wt}(c_1), \ldots, \mathrm{wt}(c_i)) \in W$, where $W$ is a given subset of $\mathbb{Z}^i$.

*Example* 4. It is easy to verify that the words of weight $0, 2, 5$, and $7$ in $\mathbb{F}_2^7$ form a $(7, 44, 1, 3)$ MCF.

*Example* 5. Choose $n_1 = 3$ and $n_2 = 2k-1$ and take as codewords all the words

$(c_1, c_2)$, for which $(\mathrm{wt}(c_1), \mathrm{wt}(c_2))$ is one of the following pairs:

| | |
|---|---|
| $0$ | $0$ |
| $2$ | $0$ |
| $1$ | $2k-1$. |
| $3$ | $2k-1$ |

Then it is easy to check that we obtain a $(2k+2, 8, k, 4)$ MCF. Hence $F(2k+2, k, 4) \le 8$ for all $k \ge 1$.

*Example* 6. Choose $n_1 = 4$ and $n_2 = 2k - 1$ and take as codewords all the words $(c_1, c_2)$ such that $(\mathrm{wt}(c_1), \mathrm{wt}(c_2))$ is one of the following four pairs:

| | |
|---|---|
| $1$ | $0$ |
| $4$ | $0$ |
| $0$ | $2k-1$. |
| $3$ | $2k-1$ |

Again, it is easy to check that we obtain a $(2k+3, 10, k, 2)$ MCF and that therefore $F(2k+3, k, 2) \le 10$ for all $k \ge 1$.

*Example* 7. Finally, choose $n_1 = 4$ and $n_2 = 2k - 3$, where $k \ge 2$, and take the words $(c_1, c_2)$ for which $(\mathrm{wt}(c_1), \mathrm{wt}(c_2))$ is one of the following pairs:

| | |
|---|---|
| $1$ | $0$ |
| $0$ | $2k-3$. |
| $4$ | $2k-3$ |

It is easy to verify that these six words form a $(2k+1, 6, k, 4)$ MCF and hence $F(2k+1, k, 4) \le 6$ for all $k \ge 2$. When $k = 1$, we clearly have $F(3, 1, 4) \le 2^3 = 8$. On the other hand, if $C$ is a $(3, 7, 1, 4)$ MCF and $x$ is the only word in $\mathbb{F}_2^3 \setminus C$ then $d(x, C) = 1$, but $|B_1(x) \cap C| \le |B_1(x)| - 1 = 3$. Consequently $F(3, 1, 4) = 8$. This gives another example of a case where the value of the function $\bar{F}$ is smaller. Indeed $\bar{F}(3, 1, 4) = 6$ because the six words $000, 000, 110, 101, 011, 111$ form a $(3, 6, 1, 4)$ MCFR.

The idea of piecewise constant codes applies equally well to MCFRs as shown in the following examples.

*Example* 8. If we take the words of weight 3 in $\mathbb{F}_2^7$, each word of weight 6 twice, and the word of weight 0 four times, we obtain a $(7, 53, 1, 4)$ MCFR.

*Example* 9. If we take the words of weight 3 in $\mathbb{F}_2^6$ and the all-0 and all-1 words each four times, we clearly obtain a $(6, 28, 1, 4)$ MCFR.

*Example* 10. If we take the words of weight 3 and 5 in $\mathbb{F}_2^5$ and the all-0 word three times, we obtain a $(5, 14, 1, 3)$ MCFR.

*Example* 11. If we take all the words in $\mathbb{F}_2^9$ of weight 3 and 6 and the all-0 and all-1 words four times each, we obtain a $(9, 176, 1, 4)$ MCFR. If we instead take all the words of weight $1, 3, 6$, and $8$ in $\mathbb{F}_2^9$ we get a $(9, 186, 1, 4)$ MCF instead.

**3.4. A matrix construction.** The matrix construction of [11], [1], [13], [2] (see also, e.g., [15]) can easily be modified to the case of MCFs and MCFRs. Let $A = (I; D) = (a^{(1)}; \ldots; a^{(N)})$ be an $n \times N$ binary matrix and $I$ the identity matrix and suppose $s^{(1)}, \ldots, s^{(t)}$ are not necessarily distinct words in $\mathbb{F}_2^n$ represented as column vectors.

THEOREM 5. *If every $x \in \mathbb{F}_2^n$ can be represented as a sum of exactly one $s^{(j)}$ and at most $r - 1$ of the columns $a^{(i)}$, or in at least $\mu$ different ways as a sum of exactly one $s^{(j)}$ and $r$ of the columns $a^{(i)}$, then the words in the sets*

$$C_j = \{y \in \mathbb{F}_2^N \mid Ay = s^{(j)}\}, \quad j = 1, 2, \ldots, t$$

*together form an* $(N, t2^{N-n}, r, \mu)$ MCFR $C$. *If all the words* $s^{(j)}$ *are different, then* $C$ *is an* MCF.

*Proof.* Let $z \in \mathbb{F}_2^N$. Then $x = Az$ can be represented as a sum

$$x = a^{(i_1)} + \cdots + a^{(i_k)} + s^{(j)}$$

for some $k \le r$ and some indices $i_1 < \cdots < i_k, j$, and therefore the word $z'$ obtained by adding 1 to the coordinates $i_1, \ldots, i_k$ of $z$ belongs to $C_j$. Furthermore, we know that (a) we can find a representation in which $k < r$ in which case $d(z, C) \le d(z, z') = k < r$, or (b) we can find at least $\mu$ representations in which $k = r$, proving that $C$ is an $(N, t2^{N-n}, r, \mu)$ MCFR. $\square$

This construction turns out to be quite useful. Simulated annealing (see, e.g., [18]), with some suitable modifications, can be used in searching for a suitable matrix $A$ and for the words $s^{(j)}$.

In Table 1 we list all the codes found using this method that are referred to in Table 3. In Table 1 we first give $N$, $M, r, \mu$, and $n$ and then list in hexadecimal $(0 = 0000, 1 = 0001, \ldots, F = 1111)$ the columns $s^{(j)}$ and finally the $N - n$ columns of $D$. As an example, consider the $(16, 192, 3, 1)$ code. The six columns $s^{(j)}$ and the columns of matrix $A$ are

$$\begin{matrix}
0&0&1&1&1&1\\
0&1&0&0&1&1\\
0&1&1&1&0&0\\
1&0&0&0&1&1\\
1&0&0&1&0&1\\
1&0&0&0&1&1\\
0&1&1&0&0&0\\
1&1&0&0&0&1\\
1&0&1&0&1&1\\
0&1&0&0&1&1\\
0&0&1&1&0&1
\end{matrix}
\quad \text{and} \quad
\begin{matrix}
1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&1&0&1&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1
\end{matrix},$$

respectively.

**3.5. Perfect MCFs with $r = 1$.** We have the following proposition.

THEOREM 6 [5]. *A perfect* $(n, \cdot, 1, \mu)$ MCF *exists if* $n = \mu(2^i - 1)$ *for some* $i$.

These MCFs were constructed in the following way. Denote by $V^0$ (respectively, $V^1$) the set of all binary words of length $\mu$ that have even (respectively, odd) weight and by $\mathcal{H}$ the Hamming code of length $2^i - 1$ and let

$$C = \bigcup_{c=(c(1),\ldots,c(2^i-1))\in\mathcal{H}} V^{c(1)} \oplus V^{c(2)} \oplus \cdots \oplus V^{c(2^i-1)}.$$

It is easy to check that this is a perfect $(n, \cdot, 1, \mu)$ MCF.

TABLE 1
*List of* MCF*s found using the matrix method.*

| $N$ | $M$ | $r$ | $\mu$ | $n$ | Columns $s^{(j)}$/columns of $D$ |
|---|---|---|---|---|---|
| 6 | 24 | 1 | 3 | 5 | 0, 3, 4, B, D, E, 11, 16, 17, 18, 1A, 1D/1F |
| 7 | 58 | 1 | 4 | 6 | 1, 3, 4, 9, A, C, F, 11, 12, 16, 17, 18, 1B, 1D, 1E, 20, 22, 25, 26, 2B, 2C, 2F, 33, 34, 35, 38, 39, 3A, 3F/3F |
| 8 | 80 | 1 | 3 | 4 | 0, 3, 6, C, D/8, 9, 9, 7 |
| 8 | 104 | 1 | 4 | 6 | 1, 4, 6, 9, A, F, 11, 12, 17, 18, 1B, 1C, 1D, 20, 23, 25, 27, 29, 2A, 2C, 31, 32, 34, 39, 3E, 3F/20, 1F |
| 9 | 106 | 1 | 2 | 8 | 5, B, C, 13, 16, 19, 1A, 20, 27, 2E, 30, 3B, 3D, 42, 45, 4C, 50, 5B, 5F, 67, 69, 6E, 71, 72, 74, 78, 81, 86, 88, 92, 94, 9F, A3, AA, AD, B5, B6, B9, BC, C1, C8, CF, D6, D7, DB, DD, DE, E3, E4, EA, F0, FD, FF/FF |
| 11 | 368 | 1 | 2 | 8 | 0, 2, 5, B, F, 2C, 32, 35, 37, 39, 3A, 3C, 4C, 53, 54, 58, 5D, 5E, 60, 62, 65, 6B, 6F, 8C, 90, 91, 96, 9B, 9C, 9F, A1, A6, A7, A9, AA, C1, C6, C7, C9, CA, EC, F3, F4, F8, FD, FE/E0, D0, AF |
| 11 | 592 | 1 | 4 | 8 | 0, 6, 7, B, C, 11, 14, 1A, 1D, 23, 25, 26, 28, 2F, 30, 33, 37, 39, 3E, 41, 44, 4A, 4D, 52, 57, 58, 5B, 5E, 62, 67, 69, 6C, 6F, 71, 74, 7A, 7D, 82, 84, 85, 89, 8E, 93, 96, 98, 9F, A1, A4, A7, AA, AD, B1, B2, B5, BB, BC, C3, C6, C8, CF, D0, D5, D9, DA, DC, E0, E5, EB, ED, EE, F3, F6, F8, FF/80, 70, F |
| 12 | 976 | 1 | 3 | 8 | 3, 6, D, 10, 14, 16, 1B, 1C, 22, 25, 28, 29, 2E, 31, 36, 3F, 41, 48, 4F, 52, 55, 58, 5E, 60, 67, 6B, 6C, 73, 78, 7D, 81, 88, 8F, 94, 96, 97, 9A, 9B, A2, A4, A9, AA, AE, B3, B8, BD, C3, C6, CD, D0, D5, D9, DA, DC, E4, E7, E8, EA, F1, F6, FF/E0, D0, CC, B3 |
| 13 | 1248 | 1 | 2 | 9 | 0, 9, E, 14, 1A, 1D, 27, 28, 3C, 42, 4B, 4C, 56, 58, 5F, 6A, 71, 7E, 83, 91, 92, 97, 9B, A4, AD, B2, B3, BA, BB, C5, D3, E0, E6, E9, EF, F4, FD, 105, 113, 122, 12B, 130, 136, 139, 13F, 141, 144, 147, 14D, 155, 164, 165, 16C, 16D, 172, 17B, 186, 188, 18A, 18C, 18F, 198, 19C, 19E, 1A1, 1AE, 1B5, 1C8, 1CA, 1CE, 1D0, 1D9, 1DA, 1DC, 1DE, 1E3, 1F7, 1F8/100, 80, 180, 1FF |
| 13 | 1728 | 1 | 3 | 8 | 2, 4, D, 11, 16, 1A, 1C, 21, 27, 28, 2E, 34, 37, 3B, 40, 47, 49, 4A, 53, 54, 5F, 62, 64, 6D, 72, 79, 7E, 82, 84, 8F, 92, 99, 9C, A1, A6, A8, AB, B0, B5, BF, C0, C5, CB, CE, D3, D5, D6, D8, E3, E6, EC, F0, FA, FD/C0, A0, 50, 3C, F3 |
| 14 | 3072 | 1 | 3 | 8 | 1, 2, F, 13, 1C, 1E, 25, 29, 2E, 30, 35, 3A, 41, 46, 4A, 55, 5A, 5F, 60, 6D, 6E, 71, 73, 7C, 84, 8B, 8C, 90, 97, 99, A2, A7, A8, B6, BB, BD, C7, C8, CD, D2, D4, D9, E3, E4, EB, F6, F8, FF/80, C0, 38, 76, 75, EB |
| 16 | 14336 | 1 | 4 | 8 | 1, 3, 6, F, 13, 14, 1D, 1E, 20, 21, 24, 2A, 35, 36, 38, 3B, 40, 47, 49, 53, 5A, 5C, 63, 66, 6C, 74, 79, 7F, 85, 89, 8A, 92, 97, 98, A2, AC, AF, B1, B6, BD, C4, C7, CB, CD, D0, D1, D5, DE, E2, E5, E8, EE, F0, F2, F7, FB/E0, DC, BA, 16, 6E, B5, 73, C7 |
| 9 | 26 | 2 | 2 | 8 | 0, 1D, 3A, 4B, 61, 76, 86, AB, B1, D3, D8, E7, EC/FF |
| 10 | 56 | 2 | 3 | 7 | 0, 5, 3B, 3E, 5B, 60, 65/39, 75, 53 |
| 11 | 96 | 2 | 3 | 6 | 0, 16, 1E/2E, 3D, 3B, 37, F |
| 11 | 112 | 2 | 4 | 9 | 1B, 62, 64, 68, 6D, 75, 76, 81, 84, 87, 8E, 90, 9C, FB, 104, 109, 10A, 112, 117, 11D, 17B, 19B, 1E3, 1E4, 1EF, 1F1, 1F8, 1FE/1C0, 1A0 |
| 12 | 160 | 2 | 3 | 7 | 2, 1C, 1F, 21, 3E/70, 68, 64, 5C, 3F |
| 13 | 224 | 2 | 2 | 10 | 31, 3A, 3D, 4A, 84, 87, 8A, FA, 135, 13B, 142, 145, 185, 18E, 1F5, 1F8, 236, 240, 24F, 289, 2F3, 2FC, 330, 349, 34C, 383, 3F6, 3FF/300, E0, D0 |
| 13 | 368 | 2 | 4 | 9 | E, 3C, 60, 68, 7B, 7C, 85, 88, A2, D6, D8, ED, 103, 131, 144, 153, 15F, 19D, 1B6, 1BA, 1C7, 1E7, 1E9/1F0, 1CC, 1AA, 156 |
| 14 | 384 | 2 | 2 | 10 | 4, 6, 38, 49, 63, 7D, 18A, 19D, 1AF, 1D7, 1E0, 1E4, 29B, 2A9, 2B5, 2D0, 2D2, 2EE, 301, 332, 336, 34F, 35C, 37B/380, 70, 36C, 363 |
| 14 | 640 | 2 | 4 | 9 | 21, 2E, 54, 5B, 87, 88, B2, BD, E4, EB, 117, 118, 142, 14D, 177, 178, 1A4, 1AB, 1D1, 1DE/100, 80, 40, 20, 10 |
| 15 | 896 | 2 | 3 | 10 | C, 21, 5A, B4, D4, EB, F3, FC, 10B, 113, 134, 154, 18C, 1BA, 1C1, 1F4, 23F, 247, 274, 282, 299, 32C, 34C, 362, 379, 394, 3A7, 3DF/200, 3E0, 19C, 35A, 339 |
| 16 | 1344 | 2 | 2 | 10 | C, 65, 7F, 8F, B1, CF, F1, 158, 16A, 1C1, 1D7, 226, 23C, 2B2, 2CC, 2F0, |

TABLE 1
*Continued*

| $N$ | $M$ | $r$ | $\mu$ | $n$ | Columns $s^{(j)}$/columns of $D$ |
|---|---|---|---|---|---|
| | | | | | 2F2, 31B, 329, 382, 394/3C0, 338, 2B4, 6C, 1DC, 23C |
| 15 | 192 | 3 | 3 | 10 | A9, 16E, 271, 3B2, 3B4, 3B7/380, 340, 38, 2F7, 2EF |
| 16 | 192 | 3 | 1 | 11 | EC, 31A, 515, 541, 6A6, 6EF/700, 680, 5C0, 660, 1F |
| 16 | 448 | 3 | 4 | 10 | 5A, 197, 1CD, 2C0, 2EA, 2FA, 375/380, 370, 2E8, 1E6, 2D5, 1E3 |
| 14 | 40 | 4 | 3 | 12 | 0, 1FB, 2EC, 71A, 783, 935, ABB, CD6, E3A, F4D/B62, 49D |
| 16 | 96 | 4 | 2 | 11 | 0, 4DE, 5FC/79A, 4DE, 671, 15D, 66B |

TABLE 2
*List of linear* MCF*s found using the matrix method.*

| $N$ | $M$ | $r$ | $\mu$ | Generator matrix |
|---|---|---|---|---|
| 10 | 32 | 2 | 2 | 0101110000 |
| | | | | 0011101000 |
| | | | | 1111000100 |
| | | | | 1001100010 |
| | | | | 1110100001 |
| 13 | 256 | 2 | 3 | 1101011010000 |
| | | | | 1010111001000 |
| | | | | 1101100100100 |
| | | | | 0001111100010 |
| | | | | 0110101100001 |
| 15 | 1024 | 2 | 4 | 011111000010000 |
| | | | | 101110100001000 |
| | | | | 110110010000100 |
| | | | | 111010001000010 |
| | | | | 111100000100001 |
| 16 | 256 | 3 | 2 | 1111010010000000 |
| | | | | 0100010101000000 |
| | | | | 1010101000100000 |
| | | | | 1100110000010000 |
| | | | | 0111111000001000 |
| | | | | 0101001100000100 |
| | | | | 0001011100000010 |
| | | | | 1100011000000001 |

**3.6. Other codes.** For the very smallest values of $n$, we can determine the exact values of $F(n, r, \mu)$.

THEOREM 7. *If* $\mu \geq 2$ *then*

$$F(n,r,\mu) = \begin{cases} 1 & \text{if } 1 \leq n < r, \\ 2 & \text{if } r \leq n < 2r \text{ or if } n = 2r \text{ and } \mu = 2, \\ 4 & \text{if } n = 2r \text{ and } \mu \geq 3. \end{cases}$$

*Proof.* The first two cases are simple, and their proofs are omitted. If $n = 2r$, then $F(2r, r, \mu) \leq K(2r, r-1, 1) = 4$ by [7]. It suffices to show that $F(2r, r, \mu) \geq 4$ when $\mu \geq 3$. Assume on the contrary that there is a $(2r, 3, r, \mu)$ code $C$ consisting of three words $0^{2r}, c$, and $c'$. Consider the coordinates in pairs $\{1,2\}, \{3,4\}, \ldots, \{2r-1, 2r\}$. Choose $x = x(1)x(2)\cdots x(n-1)x(n)$ so that $x(1)x(2)$ is a pair not appearing in coordinates 1 and 2 of the codewords of $C$, $x(3)x(4)$ a pair not appearing in coordinates 3 and 4, and so on. Then $d(x, C) = r$. If, e.g., coordinate 1 of $C$ is identically 0, then either 0 or 1—say 0—appears at most once in coordinate 2. Then, choosing $x(1) = 1$ and $x(2) = 0$, we see that $x$ has distance $r$ to at most one word of $C$. Hence we may conclude that no coordinate is identically 0 in $C$. If all three pairs appearing in coordinates 1 and 2 of $C$ are different, then clearly $x$ has distance at most $r$ to at most two of the codewords of $C$. We can therefore assume that the only pairs appearing in $C$ in coordinates 1 and 2 are 00 and 11. Neither $c$ nor $c'$ is the all-0 word and both of them are not the all-1 word, and we can w.l.o.g. assume that $c(1) = c(2) = 1, c(3) = c(4) = 0$, and $c'(3) = c'(4) = 1$. When we now choose $x(3) = x(4) = 1, x(1) = x(2) = 1 + c'(1) = 1 + c'(2)$, and $x(5), \ldots, x(n)$ as before, we see again that $x$ can have Hamming distance $r$ to at most two of the words in $C$. □

*Remark.* Clearly, $\bar{F}(2r, r, 3) = 3$, as can be seen by taking the all-0 word twice and the all-1 word once.

Finally, we mention some examples and present some lengthening methods.

*Example* 12. Take as codewords all the cyclic shifts of the word 11101001000, 00000000000 and their complements to obtain a $(11, 24, 3, 3)$ MCF.

In many cases the amalgamated direct sum (ADS) (see [8]) of an $(n, M, r, \mu)$ MCF and the code $\{000, 111\}$ gives us an $(n+2, M, r+1, \mu)$ MCF. We have checked all these cases separately, and the upper bounds obtained in this way have been marked with $g$ in Table 3.

THEOREM 8. $F(2n, 1, 2\mu) \leq 2^n F(n, 1, \mu)$.

*Proof.* Let $C$ be an $(n, F(n, 1, \mu), 1, \mu)$ MCF and $D = \{(x, x+c) | x \in \mathbb{F}_2^n, c \in C\}$. If $(\alpha, \alpha+\beta) \in \mathbb{F}_2^{2n}$, where $\alpha \in \mathbb{F}_2^n$ has distance 1 to $D$, then we can choose $c \in C$ in $\mu$ different ways so that $d((\alpha, \alpha+\beta), (\alpha, \alpha+c)) = 1$. All the words $\beta + c$ have weight 1, and we find another $\mu$ codewords $(\alpha+(\beta+c), \alpha+\beta) = (\alpha+(\beta+c), \alpha+(\beta+c)+c) \in D$ that have distance 1 to $(\alpha, \alpha+\beta)$. □

THEOREM 9. $F(2n+\mu, 1, \mu) \leq 2^{n+\mu-1} F(n, 1, \mu)$.

*Proof.* Let $C$ be an $(n, F(n, 1, \mu), 1, \mu)$ MCF and $D = \{(x_0, x, x+c) | x_0 \in \mathbb{F}_2^\mu, x \in \mathbb{F}_2^n, c \in C, \mathrm{wt}(x_0 x) \text{ is even}\}$. Let $(\alpha_0, \alpha, \alpha+\beta) \in \mathbb{F}_2^{\mu+2n}$, where $\alpha_0 \in \mathbb{F}_2^\mu, \alpha, \beta \in \mathbb{F}_2^n$, be arbitrary.

Assume first that $d(\beta, C) = 1$ and $d(\beta, c) = 1$ where $c \in C$. The words $\beta$ and $c$ disagree in exactly one—say the $i$th—coordinate. If $\mathrm{wt}(\alpha_0\alpha)$ is even, then $(\alpha_0, \alpha, \alpha+c) \in D$ and $d((\alpha_0, \alpha, \alpha+c), (\alpha_0, \alpha, \alpha+\beta)) = 1$; otherwise $(\alpha_0, \alpha', \alpha'+c) \in D$ and $d((\alpha_0, \alpha', \alpha'+c), (\alpha_0, \alpha, \alpha+\beta)) = 1$ where $\alpha'$ is the word obtained by changing the $i$th coordinate in $\alpha$. The same argument applies to all the $\mu$ different words $c \in C$ such that $d(\beta, c) = 1$ and gives us $\mu$ different words of $D$.

Assume that $\beta \in C$. If $\mathrm{wt}(\alpha_0\alpha)$ is even, then $(\alpha_0, \alpha, \alpha+\beta) \in D$. Finally, if $\mathrm{wt}(\alpha_0\alpha)$ is odd, then $(\alpha_0^i, \alpha, \beta) \in D$ for all $i = 1, 2, \ldots, \mu$, where $\alpha_0^i$ denotes the word obtained by changing the $i$th coordinate in $\alpha_0$. □

*Example* 13. It can be verified that a code $\{(x, x+y) | x, y \in \mathbb{F}_2^7, \mathrm{wt}(x) = 0, 2, 5 \text{ or } 7, \text{ and } \mathrm{wt}(y) = 0 \text{ or } 7\}$ is a $(14, 88, 3, 2)$ MCF.

## 4. Tables.

In Tables 3.1–3.4 we give upper bounds for $F(n, r, \mu)$. In a number of cases better upper bounds are known for $\bar{F}(n, r, \mu)$. For the following upper bounds, see §3.3:

$$\bar{F}(3,1,4) \leq 6, \quad \bar{F}(7,1,4) \leq 53, \quad \bar{F}(6,1,4) \leq 28,$$

$$\bar{F}(5,1,3) \leq 14, \qquad \bar{F}(9,1,4) \leq 176.$$

The proofs of the following bounds are omitted:

$$\bar{F}(8,1,2) \leq 58, \quad \bar{F}(11,1,2) \leq 352, \quad \bar{F}(11,1,3) \leq 508,$$

$$\bar{F}(8,1,4) \leq 96, \qquad \bar{F}(14,3,3) \leq 106.$$

See also the remark after Theorem 7.

*Key to Tables* 3.1.–3.4. The upper bounds for $\mu = 1$ are from [9], [16], and [17].
$a$: $F(n+1, r, \mu) \leq 2F(n, r, \mu)$,
$b$: puncturing (see §3.2),
$g$: ADS of a code with $\{000, 111\}$ (see §3.6),
$l$: linear code (see Table 2),
$m$: matrix construction (see Table 1 and §3.4),
$p$: Theorem 2,
$r$: Theorem 8,
$s$: piecewise constant code (see §3.3),
$t$: $F(n, r, \mu) \leq F(n, r, \mu + 1)$,
$v$: Theorem 1,
$w$: Theorem 9,
$x$: [9],
$y$: perfect weighted covering (see §3.5 and [5]),
$z$: see §3.6.

TABLE 3.1
*Upper bounds for* $F(n, 1, \mu)$.

| $n$ \ $\mu$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1 | 2 $z$ | 2 $z$ | 2 $z$ |
| 2 | 2 | 2 $z$ | 4 $z$ | 4 $z$ |
| 3 | 2 | 4 $t$ | 4 $y$ | 8 $a$ |
| 4 | 4 | 7 $v$ | 8 $t$ | 8 $s$ |
| 5 | 7 | 10 $s$ | 16 $t$ | 16 $a$ |
| 6 | 12 | 16 $y$ | 24 $m$ | 32 $a$ |
| 7 | 16 | 32 $a$ | 44 $s$ | 58 $m$ |
| 8 | 32 | 62 $b$ | 80 $m$ | 104 $m$ |
| 9 | 62 | 106 $m$ | 128 $y$ | 186 $s$ |
| 10 | 120 | 192 $b$ | 256 $a$ | 320 $r$ |
| 11 | 192 | 368 $m$ | 512 $a$ | 592 $m$ |
| 12 | 380 | 640 $w$ | 976 $m$ | 1024 $y$ |
| 13 | 736 | 1248 $m$ | 1728 $m$ | 2048 $a$ |
| 14 | 1408 | 2048 $y$ | 3072 $m$ | 4096 $a$ |
| 15 | 2048 | 4096 $a$ | 6144 $a$ | 8192 $a$ |
| 16 | 4096 | 8192 $a$ | 12288 $a$ | 14336 $m$ |

TABLE 3.2
*Upper bounds for* $F(n, 2, \mu)$.

| $n$ \ $\mu$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 2 | 1 | 2 $z$ | 2 $z$ | 2 $z$ |
| 3 | 2 | 2 $z$ | 2 $z$ | 2 $z$ |
| 4 | 2 | 2 $z$ | 4 $z$ | 4 $z$ |
| 5 | 2 | 4 $t$ | 4 $p$ | 6 $s$ |
| 6 | 4 | 7 $t$ | 7 $p$ | 8 $s$ |
| 7 | 7 | 10 $s$ | 12 $t$ | 12 $p$ |
| 8 | 12 | 16 $t$ | 16 $t$ | 16 $p$ |
| 9 | 16 | 26 $m$ | 32 $a$ | 32 $a$ |
| 10 | 30 | 32 $l$ | 56 $m$ | 62 $p$ |
| 11 | 44 | 64 $a$ | 96 $m$ | 112 $m$ |
| 12 | 78 | 128 $a$ | 160 $m$ | 192 $p$ |
| 13 | 128 | 224 $m$ | 256 $l$ | 368 $m$ |
| 14 | 256 | 384 $m$ | 512 $a$ | 640 $m$ |
| 15 | 480 | 768 $a$ | 896 $m$ | 1024 $l$ |
| 16 | 896 | 1344 $m$ | 1792 $a$ | 2048 $a$ |

TABLE 3.3
*Upper bounds for* $F(n, 3, \mu)$.

| $n$ $\mu$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 3 | 1 | 2 $z$ | 2 $z$ | 2 $z$ |
| 4 | 2 | 2 $z$ | 2 $z$ | 2 $z$ |
| 5 | 2 | 2 $z$ | 2 $z$ | 2 $z$ |
| 6 | 2 | 2 $z$ | 4 $z$ | 4 $z$ |
| 7 | 2 | 4 $t$ | 4 $p$ | 6 $s$ |
| 8 | 4 | 7 $t$ | 7 $p$ | 8 $s$ |
| 9 | 7 | 10 $s$ | 12 $t$ | 12 $g$ |
| 10 | 12 | 16 $t$ | 16 $t$ | 16 $g$ |
| 11 | 16 | 24 $t$ | 24 $z$ | 30 $p$ |
| 12 | 28 | 32 $g$ | 44 $t$ | 44 $p$ |
| 13 | 42 | 64 $a$ | 78 $t$ | 78 $p$ |
| 14 | 64 | 88 $z$ | 128 $t$ | 128 $p$ |
| 15 | 112 | 160 $x$ | 192 $m$ | 256 $a$ |
| 16 | 192 $m$ | 256 $l$ | 384 $a$ | 448 $m$ |

TABLE 3.4
*Upper bounds for* $F(n, 4, \mu)$.

| $n$ $\mu$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 4 | 1 | 2 $z$ | 2 $z$ | 2 $z$ |
| 5 | 2 | 2 $z$ | 2 $z$ | 2 $z$ |
| 6 | 2 | 2 $z$ | 2 $z$ | 2 $z$ |
| 7 | 2 | 2 $z$ | 2 $z$ | 2 $z$ |
| 8 | 2 | 2 $z$ | 4 $z$ | 4 $z$ |
| 9 | 2 | 4 $t$ | 4 $p$ | 6 $s$ |
| 10 | 4 | 7 $t$ | 7 $p$ | 8 $s$ |
| 11 | 7 | 10 $s$ | 12 $t$ | 12 $g$ |
| 12 | 12 | 16 $t$ | 16 $t$ | 16 $g$ |
| 13 | 16 | 24 $t$ | 24 $g$ | 28 $p$ |
| 14 | 28 | 32 $g$ | 40 $m$ | 42 $p$ |
| 15 | 40 | 64 $t$ | 64 $t$ | 64 $p$ |
| 16 | 64 | 96 $m$ | 112 $t$ | 112 $p$ |

**Acknowledgments.** The authors would like to thank Markku Kaikkonen for many useful comments and Tuvi Etzion for interesting discussions. The third author is grateful for the hospitality of the University of Turku during his visit in August 1992. The authors would also like to thank the referees for their many intelligent remarks.

## REFERENCES

[1] A. BLOKHUIS AND C. W. H. LAM, *More coverings by rook domains*, J. Combin. Theory Ser. A, 36 (1984), pp. 240–244.

[2] W. A. CARNIELLI, *Hyper-rook domain inequalities*, Stud. Appl. Math., 82 (1990), pp. 59–69.

[3] R. F. CLAYTON, *Multiple packings and coverings in algebraic coding theory*, Ph.D. thesis, University of California, Los Angeles, CA, 1987.

[4] G. D. COHEN, I. S. HONKALA, AND S. N. LITSYN, *On weighted coverings and packings with diameter one*, Proc. EUROCODE 1992, Udine, Italy, in CISM Courses and Lectures No. 339, Springer-Verlag, Vienna, New York, 1993, pp. 43–49.

[5] G. D. COHEN, S. N. LITSYN, AND H. F. MATTSON, JR., *Binary perfect weighted coverings I, the linear case*, in Sequences II, Capocelli, DeSantis, and Vaccaro, eds., Springer-Verlag, New York, Berlin, 1993, pp. 36–51.

[6] ———, *On perfect weighted coverings with small radius*, in Lecture Notes in Computer Science 573, Springer-Verlag, Berlin, Heidelberg, 1992, pp. 32–41.

[7] G. D. COHEN, A. C. LOBSTEIN, AND N. J. A. SLOANE, *Further results on the covering radius*

*of codes*, IEEE Trans. Inform. Theory, 32 (1986), pp. 680–692.

[8] R. L. Graham and N. J. A. Sloane, *On the covering radius of codes*, IEEE Trans. Inform. Theory, 31 (1985), pp. 385–401.

[9] H. O. Hämäläinen, I. S. Honkala, M. K. Kaikkonen, and S. N. Litsyn, *Bounds for binary multiple covering codes*, Des. Codes and Cryptogr., 3 (1993), pp. 251–275.

[10] Hämäläinen and S. Rankinen, *Upper bounds for football pool problems and mixed covering codes*, J. Combin. Theory Ser. A, 56 (1991), pp. 84–95.

[11] H. J. L. Kamps and J. H. van Lint, *A covering problem*, Colloq. Math. Soc. János Bolyai, 4 (1970), pp. 679–685.

[12] J. H. van Lint, *Recent results on covering problems*, in Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, Proceedings of the 6th International Conference AAECC-6, Rome, Italy, July 1988, T. Mora, ed., Springer-Verlag, Berlin, 1989, pp. 7–21.

[13] J. H. van Lint, Jr., *Covering radius problems*, Master's thesis, Eindhoven University of Technology, 1989.

[14] P. R. J. Östergård, *Further results on $(k, t)$-subnormal covering codes*, IEEE Trans. Inform. Theory, 38 (1992), pp. 206–210 and 1738.

[15] ———, *New upper bounds for the football pool problem for* 11 *and* 12 *matches*, J. Combin. Theory Ser. A, 67 (1994), pp. 161–168.

[16] ———, *New constructions for q-ary covering codes*, Discrete Math., submitted.

[17] P. R. J. Östergård and H. O. Hämäläinen, *A new table of binary/ternary mixed covering codes*, Des. Codes Cryptogr., submitted.

[18] L. T. Wille, *The football pool problem for* 6 *matches: A new upper bound*, J. Combin. Theory Ser. A, 35 (1983), pp. 106–108.

[19] Z. Zhang, *Linear inequalities for covering codes: Part* I*—pair covering inequalities*, IEEE Trans. Inform. Theory, 37 (1991), pp. 573–582.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 208–222, May 1995

# THE EXISTENCE OF HOMOMORPHISMS TO ORIENTED CYCLES *

PAVOL HELL[†] AND XUDING ZHU[†]

**Abstract.** We discuss the existence of homomorphisms of arbitrary digraphs to a fixed oriented cycle $C$. Our main result asserts that if the cycle $C$ is unbalanced then a digraph $G$ is homomorphic to $C$ if and only if (1) every oriented path homomorphic to $G$ is also homomorphic to $C$, and (2) the length of every cycle of $G$ is a multiple of the length of $C$. This answers a conjecture from an earlier paper with H. Zhou and generalizes a result proved there. We also show that this characterization does not hold for balanced cycles. We relate these results to work on the complexity of homomorphism problems.

**Key words.** graph homomorphisms, oriented cycles, homomorphism duality

**AMS subject classifications.** 05C99, 05C20, 05C38

## 1. Introduction.

All the digraphs discussed in this paper are finite unless otherwise specified. A *homomorphism* $G \to H$ of a digraph $G$ to a digraph $H$ is a mapping of the vertex sets $f : V(G) \to V(H)$ which preserves the edges, i.e., such that $xy \in E(G)$ implies $f(x)f(y) \in E(H)$. If such a homomorphism exists, we say $G$ is *homomorphic* to $H$ and write $G \to H$. Otherwise we write $G \not\to H$. Note that these notions can also be applied to undirected graphs by viewing them as symmetric digraphs. If $K_n$ denotes the undirected complete graph on $n$ vertices, then for an arbitrary undirected graph $G$, a homomorphism $G \to K_n$ is just an $n$-colouring of $G$. Because of this fact, it is also common to call a homomorphism $G \to H$ (of general digraphs) an *$H$-colouring* of $G$.

Suppose $g_0 \in V(G)$ is a fixed vertex of $G$, called the root of $G$, and $h_0 \in V(H)$ is the root of $H$. A *rooted homomorphism* of $G$ to $H$ is a homomorphism $h : G \to H$ such that $h(g_0) = h_0$. In this case we write $(G, g_0) \to (H, h_0)$ and say $h$ is a homomorphism of $(G, g_0)$ to $(H, h_0)$. We observe that the composition of rooted homomorphisms $(G, g_0) \to (H, h_0)$ and $(H, h_0) \to (J, j_0)$ is a rooted homomorphism $(G, g_0) \to (J, j_0)$.

An *oriented path* $P$ is a sequence of distinct vertices $[p_0, p_1, \dots, p_n]$, such that, for each $i \in \{0, 1, \dots, n-1\}$, either $p_ip_{i+1} \in E(P)$ (a *forward* edge of $P$) or $p_{i+1}p_i \in E(P)$ (a *backward* edge of $P$), and $P$ has no other edges. The direction in which $P$ is traversed is emphasized by saying that $p_0$ is the *initial vertex* $i(P)$ of $P$, and $p_n$ is the *terminal vertex* $t(P)$ of $P$, respectively.

An *oriented cycle* $C$ is a digraph obtained from an oriented path $P$ by identifying its initial and terminal vertices. Thus an oriented cycle $C$ can be given by a circular sequence of vertices $[c_0, c_1, \dots, c_{m-1}, c_0]$, such that, for each $i \in \{0, 1, \dots, m-1\}$, either $c_ic_{i+1} \in E(C)$ (a *forward* edge of $C$) or $c_{i+1}c_i \in E(C)$ (a *backward* edge of $C$), and $C$ has no other edges. (Subscript addition is taken modulo $m$.) The direction of $C$ which agrees with the forward edges, i.e., $c_0, c_1, c_2, \dots$, is the *positive direction* of $C$, and the opposite direction, i.e., $c_0, c_{m-1}, c_{m-2}, \dots$, is the *negative direction* of $C$. We also use the term *path* (respectively, *cycle*) to mean an oriented path (respectively, an oriented cycle). Since we do not distinguish an initial vertex of an oriented

* Received by the editors November 4, 1992; accepted for publication (in revised form) April 28, 1994.

[†] Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada.

cycle, $[c_0, c_1, \ldots, c_{m-1}, c_0] = [c_1, c_2, \ldots, c_{m-1}, c_0, c_1]$, and we usually choose the most convenient vertex to start listing $C$.

A *directed cycle* (respectively, a *directed path*) is a cycle (respectively, path) in which all edges are in the same direction; if they are all forward edges we speak of a forward directed cycle (respectively, path); if they are all backward edges we speak of a backward directed cycle (respectively, path).

Suppose $G$ is a digraph. A *path* (respectively, a *cycle*) in $G$ is a subgraph of $G$ which is an oriented path (respectively, an oriented cycle). A digraph $G$ is *connected* if any two vertices are joined by some path.

The *length* $l(X)$ of an oriented path or an oriented cycle $X$ is the number of forward edges of $X$ minus the number of backward edges of $X$. Note that the length can be negative. An oriented cycle $C$ is *unbalanced* if $l(C) \neq 0$; otherwise $C$ is *balanced.* A digraph $G$ is balanced if each cycle of $G$ is balanced. For an oriented path $P = [p_0, p_1, \ldots, p_n]$ and two vertices $p_i, p_j \in P$, we define the distance $d_P(p_i, p_j)$ from $p_i$ to $p_j$ as the length of the subpath of $P$ connecting $p_i$ to $p_j$. The *level* of a vertex $p_i$ of $P$ is defined as $\lambda_P(p_i) = d_P(p_0, p_i)$. An oriented path $P$ is *minimal* if $P$ contains no proper subpath $P'$ such that $l(P) = l(P')$.

Let $H$ be a fixed graph or digraph. The *$H$-colouring problem,* or the *homomorphism problem with respect to the target $H$* denoted by *$H$-col*, is the decision problem in which the instance is a graph or digraph $G$ and the question is whether or not $G \to H$. The $H$-colouring problem has, from the algorithmic point of view, received much recent attraction [1]–[4], [8], [9], [14], [22], [30].

For undirected graphs, it was shown in [14] that $H$-col is polynomial for $H$ bipartite and $NP$-complete for all other $H$. No such clear distinction is known for digraphs. We do know many cases of $NP$-complete problems and many nontrivial cases of polynomial problems, [1]–[4], [8], [9], [22], [30]. One easy case is the existence of a homomorphism to a directed path. In fact, we have the following result; cf. [5], [10], [24].

THEOREM 1.1. *Let $P$ be a directed path of length $k$. A digraph $G$ is homomorphic to $P$ if and only if every oriented path homomorphic to $G$ has length at most $k$.*

Theorem 1.1 asserts that paths of length greater than $k$ are the only possible obstructions to a homomorphism to $P$. It is not hard to specify a polynomial algorithm for testing the condition by a breadth first labeling of $G$; cf. [10].

A polynomial algorithm for the existence of a homomorphism to any oriented path was given in [9]. It did not depend on a theorem specifying the obstructions to homomorphisms, but such a theorem was later given in [18].

THEOREM 1.2. *Let $P$ be an oriented path. A digraph $G$ is homomorphic to $P$ if and only if every oriented path homomorphic to $G$ is also homomorphic to $P$.*

This theorem generalizes Theorem 1.1 and can also be shown to imply a polynomial algorithm for $P$-col; cf. [16].

The existence of homomorphisms to oriented cycles appears to be a harder problem. In particular, there exist oriented cycles $C$ for which $C$-col is $NP$-complete; cf. [8, §4]. It is still the case that for directed cycles there is a simple characterization in terms of obstructions, given in Theorem 1.3 below; cf. [5], [10], [24].

THEOREM 1.3. *Let $C$ be a directed cycle of length $k$. A digraph $G$ is homomorphic to $C$ if and only if the length of every oriented cycle in $G$ is divisible by $k$.*

This result also implies a polynomial algorithm via a breadth first labeling modulo $k$; cf. [10].

At this point one may wonder whether or not a general obstruction result analogous to Theorem 1.2 holds for cycles. Let $C$ be an oriented cycle and let $(D_C)$ denote the following statement:

($D_C$) *A digraph $G$ is homomorphic to $C$ if and only if every oriented cycle homomorphic to $G$ is also homomorphic to $C$.*

It is unlikely that ($D_C$) holds for all oriented cycles $C$, though we shall prove it for a large class of oriented cycles. We shall prove in the last section that if ($D_C$) holds then $C$-col is in $NP \cap \mathrm{co}NP$. Thus it is not surprising that when ($D_C$) holds we usually also find a polynomial algorithm for $C$-col. (It is, of course, a priori not clear how to find such an algorithm, but see the added remark at the end of the paper.) This also means that ($D_C$) is unlikely to hold for those cycles $C$ for which $C$-col is $NP$-complete.

If $C$ is a directed cycle then ($D_C$) holds as can be easily seen using Theorem 1.3, since the condition concerning cycle lengths is weaker than the condition in ($D_C$). If $C$ is a cycle obtained from two copies of an oriented path by identifying their two initial vertices and their two terminal vertices, then ($D_C$) can also be seen to hold using Theorem 1.2. In this case we also have a weaker condition, namely, that every oriented path homomorphic to $G$ is also homomorphic to $C$. Indeed, if there is a path $P$ homomorphic to $G$ but not homomorphic to $C$, then the cycle obtained from two copies of $P$ by identifying their two initial vertices and their two terminal verices is homomorphic to $G$ but not to $C$. Another class of cycles $C$ for which ($D_C$) holds is the class of $B$-cycles. A *$B$-cycle* is an oriented cycle obtained from a forward directed path $I$ of length $n$ and a minimal oriented path $J$ of length $n-1$ by identifying their two initial vertices and their two terminal verices. Thus $B$-cycles are particular cycles of length one. The following characterization theorem was proved in [19].

THEOREM 1.4. *Let $C$ be a $B$-cycle. A digraph $G$ is homomorphic to $C$ if and only if every oriented path homomorphic to $G$ is also homomorphic to $C$.*

Thus in this case we also know that ($D_C$) holds, and once again we have a weaker condition. These weaker conditions suggest the following modified statement:

($D'_C$) *A digraph $G$ is homomorphic to $C$ if and only if*

- *every oriented path homomorphic to $G$ is also homomorphic to $C$, and*
- *the length of any cycle of $G$ is a multiple of the length of $C$.*

Note that if $C$ is a directed cycle then the first condition is vacuously satisfied and ($D'_C$) becomes Theorem 1.3. Similarly, when $C$ is a $B$-cycle, the second condition is vacuously satisfied and ($D'_C$) becomes Theorem 1.4. Thus ($D'_C$) is a stronger statement than ($D_C$), yet one which holds in all the cases in which we know ($D_C$) holds. We know that this stronger statement does not hold for all cycles. In §4 we will construct cycles $C$ such that ($D'_C$) fails. In [19] we made the following conjecture.

CONJECTURE 1.5. *If $C$ is an unbalanced oriented cycle, then a digraph $G$ is homomorphic to $C$ if and only if ($D_C$) holds.*

Here we shall prove this conjecture. Note that this implies both Theorem 1.3 and Theorem 1.4, whose proof in [19] is quite complex. Our proof also motivated a polynomial algorithm for $C$-col for any unbalanced $C$ [30]. In fact, such an algorithm has also been discovered independently by Gutjahr [8]. More recently, it has been proved [16] that whenever ($D_C$) holds, there is a polynomial algorithm for $C$-col. (See the remark at the end of this paper.)

The characterization theorem (Theorem 3.1) proved here is of course interesting in its own right, regardless of any polynomial algorithms it may yield or any implications regarding $NP \cap \mathrm{co}NP$. It has already been applied, at least in the special case of $B$-cycles, in proving the multiplicativity of certain oriented cycles [20], thereby completing the classification of multiplicative cycles [20].

**2. Auxiliary results.** We shall first prove some lemmas that are special cases of the conjecture and are needed in the proof of the main theorem. In order not to have to repeat our assumptions everywhere, we specify them explicitly here as follows.

*Assumptions.* Let $C$ be an unbalanced oriented cycle which is not a directed cycle. Assume that $l(C) > 0$ and $n$ is the maximum length of a subpath of $C$. Let $c_0$ be a vertex of $C$ such that $C = [c_0, c_1, \dots, c_{j_0}, \dots, c_{m-1}, c_0]$, where $[c_0, c_1, \dots, c_{j_0}]$ is a minimal path of length $n$.

The Assumptions result in no loss of generality in view of the fact that the result for directed cycles is known. In fact, when $C$ is a directed cycle, $l(C) = n$ and $[c_0, c_1, \dots, c_{j_0}] = [c_0, c_1, \dots, c_0]$. It is easy to see that under the Assumptions we have $n > l(C)$.

LEMMA 2.1. *Let $C$ be the cycle from the Assumptions. For any $1 \leq j \leq m-1$ we have $l([c_0, c_1, \dots, c_j]) > 0$, and for any $j_0 \leq j \leq m$ we have $l([c_0, c_{m-1}, c_{m-2}, \dots, c_j]) \geq 0$.*

*Proof.* Suppose $1 \leq j \leq j_0$. If $l([c_0, c_1, \dots, c_j]) \leq 0$ then $l([c_j, c_{j+1}, \dots, c_{j_0}]) \geq n$, contradicting either the minimality of $[c_0, c_1, \dots, c_{j_0}]$ or the maximality of $n$. Similarly, if $j_0 < j \leq m-1$ and $l[c_0, c_1, \dots, c_j]) \leq 0$, then $l(c_j, c_{j+1}, \dots, c_{m-1}, c_0) > 0$ and $l([c_j, c_{j+1}, \dots, c_0, c_1, \dots, c_{j_0}]) > n$, contradicting the choice of $n$. On the other hand, if $l([c_0, c_{m-1}, c_{m-2}, \dots, c_j]) < 0$ for $j_0 \leq j \leq m-1$, then we also have $l(c_j, c_{j+1}, \dots, c_{m-1}, c_0) > 0$, and we obtain a contradiction as above.

Note that these inequalities imply that $c_0$ has in-degree zero.

We sometimes view $[c_0, c_1, \dots, c_{m-1}, c_0]$ as an oriented path and call it $R^1(C)$; formally, we introduce a new vertex $c_0^1$ and let $R^1(C)$ be the path $[c_0, c_1, \dots, c_{m-1}, c_0^1]$, where $c_{m-1}c_0^1$ or $c_0^1 c_{m-1}$ is an edge of $R^1(C)$ precisely if $c_{m-1}c_0$ or $c_0 c_{m-1}$ is an edge of $C$. This is the path which wraps around $C$ exactly once in the positive direction starting at $c_0$ and also ending at $c_0$. In a similar fashion we define $R^q(C)$ as the path wrapping around $C$ exactly $q$ times in the positive direction, i.e., $R^q(C) = [c_0, c_1, \dots, c_{m-1}, c_0^1, c_1^1, \dots, c_{m-1}^1, c_0^2, \dots, c_0^{q-1}, c_1^{q-1}, \dots, c_{m-1}^{q-1}, c_0^q]$. For convenience we may also write $R^q(C) = [c_0^0, c_1^0, \dots, c_{m-1}^0, c_0^1, c_1^1, \dots, c_{m-1}^1, c_0^2, \dots, c_0^{q-1}, c_1^{q-1}, \dots, c_{m-1}^{q-1}, c_0^q]$. We define the *index function* as $\operatorname{Ind}(c_j^i) = im + j$. For $x, y \in R^q(C)$, we say $x \leq y$ if $\operatorname{Ind}(x) \leq \operatorname{Ind}(y)$ (and $x < y$ if $\operatorname{Ind}(x) < \operatorname{Ind}(y)$). Note that $R^q(C) \to C$ via the obvious homomorphism taking all $c_j^i$ (for $i = 0, 1, \dots, q$) to $c_j$.

Frequently, we will need to consider homomorphisms $h$ of an oriented path $P = [p_0, p_1, \dots, p_n]$ to the oriented cycle $C$. Each such $h$ defines a sequence $[h(p_0), h(p_1), \dots, h(p_n)]$ of vertices of $C$ such that for each $1 \leq i \leq n$, either $h(p_{i-1})h(p_i)$ is an edge of $C$ (if $p_{i-1}p_i$ is an edge of $P$) or $h(p_i)h(p_{i-1})$ is an edge of $C$ (if $p_i p_{i-1}$ is an edge of $P$). We call such a sequence a *walk* of $C$. If $P = [p_0, p_1, \dots, p_n]$ is a minimal path and $h : P \to C$ is a homomorphism, then we call $[h(p_0), h(p_1), \dots, h(p_n)]$ a minimal walk.

Let $P = [p_0, p_1, \dots, p_n]$ be an oriented path and $h : P \to C$ be a homomorphism. We shall also view $h$, in a natural way, as a homomorphism $P \to R^{2q} = [c_0, c_1, \dots, c_{m-1}, c_0^1, c_1^1, \dots, c_{m-1}^1, c_0^2, \dots, c_0^{2q-1}, c_1^{2q-1}, \dots, c_{m-1}^{2q-1}, c_0^{2q}]$ for a large enough $q$. Formally, we define a homomorphism $h' : P \to R^{2q}(C)$ as follows.

We let $h'(p_0) = c_i^q$ if $h(p_0) = c_i$. Note that $c_i^q$ is in the "middle range" of $R^{2q}(C)$. We will define $h'$ so that $h(p_r) = c_j$ will always imply that $h'(p_r) = c_j^a$ for some $a$. If we have already defined $h'(p_r) = c_j^a$ and $h(p_{r+1}) = c_s$, then

$$h'(p_{r+1}) = c_s^{a+1} \quad \text{if } j = m-1 \text{ and } s = 0,$$

$$h'(p_{r+1}) = c_s^{a-1} \quad \text{if } j = 0 \text{ and } s = m-1,$$

$$h'(p_{r+1}) = c_s^a \quad \text{otherwise.}$$

It is easy to see that $h'$ is well defined (as $q$ is large enough), uniquely determined by $h$ (for a fixed $q$), and indeed a homomorphism. We call $h'$ the *induced homomorphism* of $h$. We say $h(P)$ goes in the positive direction of $C$ if $\mathrm{Ind}(h'(p_0)) < \mathrm{Ind}(h'(p_n))$, and $h(P)$ goes in the negative direction if $\mathrm{Ind}(h'(p_0)) > \mathrm{Ind}(h'(p_n))$. Observe that whether $h(P)$ goes in the positive direction or the negative direction of $C$ is independent of the integer $q$, though in the definition of $h'$ we need to choose a fixed integer $q$.

For balanced graphs $G$, the second condition of Conjecture 1.5 is vacuously satisfied. Thus, when we restrict our attention to balanced graphs $G$, Conjecture 1.5 becomes the following statement.

LEMMA 2.2. *Let $G$ be a balanced digraph and $C$ be an unbalanced cycle. Then $G \to C$ if and only if every oriented path homomorphic to $G$ is also homomorphic to $C$.*

*Proof.* Clearly, if $G \to C$ then any path $P$ with $P \to G$ satisfies $P \to C$ by composition. For the converse we shall use Theorem 1.2. Assume that $G$ has $q$ vertices. Then the absolute value of the length of any path in $G$ is less than $q$. Since $G$ is balanced, the same is true for any walk in $G$, and hence also for any path $P$ homomorphic to $G$. Since the length of $C$ is not zero, any homomorphism of such path $P$ to $C$ can wrap around $C$ at most $q$ times in either the positive or negative direction. Now assume that every oriented path homomorphic to $G$ is also homomorphic to $C$. Then any path homomorphic to $G$ is also homomorphic to $R^{2q}(C)$. By Theorem 1.2, $G \to R^{2q}(C)$, and by composition with $R^{2q}(C) \to C$ we have $G \to C$.

COROLLARY 2.3. *Let $T$ be an oriented tree and $C$ be an unbalanced cycle. Then $T \to C$ if and only if every oriented path homomorphic to $T$ is also homomorphic to $C$.*

Lemma 2.2 has a corresponding rooted version.

LEMMA 2.4. *Let $G$ be a balanced digraph and $g_0$ be a fixed vertex of $G$. Let $C$ and $c_0$ be as described in the Assumptions. Then $(G, g_0) \to (C, c_0)$ if and only if $(P, i(P)) \to (G, g_0)$ implies that $(P, i(P)) \to (C, c_0)$ for every oriented path $P$.*

*Proof.* The necessity of the condition is again clear by composition. Suppose the condition is satisfied. We shall construct a homomorphism $(G, g_0) \to (C, c_0)$. Assume again that $G$ has $q$ vertices and consider $R^{2q}(C)$. Let $c^*$ be the "middle vertex" of $R^{2q}$, i.e., $c^* = c_0^q$. Recall that $(R^{2q}, c^*) \to (C, c_0)$. Note also that by Lemma 2.1 and the assumption that $l(C) > 0$, we have $d_{R^{2q}(C)}(c^*, x) > 0$ for any $x$ with $x > c^*$.

We shall first define two mappings $\phi$ and $\psi$ . For $x \in G$, let $\mathcal{P}_x$ be the set of all oriented paths $P$ such that some homomorphism $h : P \to G$ has $h(i(P)) = g_0$ and $h(t(P)) = x$. Consider a path $P \in \mathcal{P}_x$. Since $(P, i(P)) \to (G, g_0)$, we also have $(P, i(P)) \to (C, c_0)$ by our assumption. This implies that $(P, i(P)) \to (R^{2q}(C), c^*)$ by an argument similar to the one given in the proof of the previous lemma. Thus we may define

$$\phi(P) = \min\{h(t(P)) : h \text{ is a homomorphism } (P, i(P)) \to (R^{2q}(C), c^*)\}.$$

Finally, we define $\psi : G \to R^{2q}(C)$ as

$$\psi(x) = \max\{\phi(P) : P \in \mathcal{P}_x\}.$$

Here the terms "max" and "min" are taken with respect to the order determined by the index function.

Next we prove that $\psi$ is a homomorphism $(G, g_0) \to (R^{2q}(C), c^*)$. Let $(x, y) \in E(G)$ be an edge of $G$. We show that $(\psi(x), \psi(y)) \in E(R^{2q}(C))$. First, $\psi(x) \neq \psi(y)$, because for any oriented paths $P_1 \in \mathcal{P}_x$ and $P_2 \in \mathcal{P}_y$, we must have $l(P_1) = l(P_2) - 1$ (as $G$ is balanced) and, therefore, $\phi(P_1) \neq \phi(P_2)$.

Suppose $\psi(x) > \psi(y)$. Let $P \in \mathcal{P}_x$ be an oriented path such that $\phi(P) = \psi(x)$. Let $P'$ be the path obtained from $P$ by adding a new vertex $a = t(P')$ and an edge $t(P)a$. Then, obviously, $P' \in \mathcal{P}_y$. Therefore, $\phi(P') \leq \psi(y)$. Let $h : (P', i(P')) \to (R^{2q}(C), c^*)$ be a homomorphism such that $h(a) = \phi(P')$. We have $h(t(P)) \geq \phi(P) = \psi(x)$, because $h$ restricted to $P$ is a homomorphism from $(P, i(P))$ to $(R^{2q}(C), c^*)$. Since $h(t(P))h(a) \in E(R^{2q}(C))$, we have $\mathrm{Ind}(h(t(P))) \leq \mathrm{Ind}(h(a)) + 1$. Hence $\mathrm{Ind}(h(a)) \leq \mathrm{Ind}(\psi(y)) < \mathrm{Ind}(\psi(x)) \leq \mathrm{Ind}(h(t(P))) \leq \mathrm{Ind}(h(a)) + 1$. Therefore, we must have $h(t(P)) = \psi(x)$ and $h(a) = \psi(y)$, and $\psi(x)\psi(y)$ is an edge of $R^{2q}(C)$.

A similar argument applies for the case $\psi(x) < \psi(y)$. Thus $\psi$ is indeed a homomorphism.

It remains to check that $\psi(g_0) = c^*$. First we have $\psi(g_0) \geq c^*$ because the path $P$ consisting of a single vertex is in $\mathcal{P}_{g_0}$ and $\phi(P) = c^*$. If $\psi(g_0) > c^*$, let $P' \in \mathcal{P}_{g_0}$ be an oriented path such that $\phi(P') = \psi(g_0) > c^*$. Let $h : (P', i(P')) \to (R^{2q}(C), c^*)$ be a homomorphism such that $h(t(P')) = \phi(P')$. Since $l(P') = 0$ (because $G$ is balanced), we must have $d_{R^{2q}(C)}(h(i(P')), h(t(P'))) = d_{R^{2q}(C)}(c^*, \psi(g_0)) = 0$. This is a contradiction with $d_{R^{2q}(C)}(c^*, \psi(g_0)) > 0$ implied by $\psi(g_0) > c^*$ (see the end of the first paragraph of this proof). Therefore, $\psi(g_0) = c^*$.

COROLLARY 2.5. *Let $T$ be an oriented tree and $t_0 \in T$. Let $C$ and $c_0$ be as described in our assumptions. Then $(T, t_0) \to (C, c_0)$ if and only if for any oriented path $P$, $(P, i(P)) \to (T, t_0)$ implies that $(P, i(P)) \to (C, c_0)$.*

It is easy to see (from Lemma 2.2 and Corollary 2.3) that both the above lemma and corollary remain true when $C$ is a directed cycle and $c_0$ is an arbitrary vertex of $C$.

**3. The main theorem.** The following theorem verifies Conjecture 1.5 and is the main result of this paper.

THEOREM 3.1. *Let $C$ be an unbalanced cycle. A digraph $G$ is homomorphic to $C$ if and only if*

- *every oriented path homomorphic to $G$ is also homomorphic to $C$, and*
- *the length of any cycle of $G$ is a multiple of the length of $C$.*

The necessity of the condition is obvious. We shall prove the sufficiency from the following rooted version of the theorem, which is of independent interest.

THEOREM 3.2. *Let $G$ be a digraph and $g_0$ be a fixed vertex of $G$. Let $C$ and $c_0$ be as described in the Assumptions. Then $(G, g_0) \to (C, c_0)$ if and only if*

- *for every oriented path $P$, $(P, i(P)) \to (G, g_0)$ implies that $(P, i(P)) \to (C, c_0)$, and*
- *the length of any cycle of $G$ is a multiple of the length of $C$.*

It is again the case that the above theorem remains valid if $C$ is a directed cycle and $c_0$ is an arbitrary vertex of $C$. On the other hand, we do not know whether Theorem 3.2 remains true if the choice of $c_0$ is unrestricted.

We first prove that Theorem 3.2 implies Theorem 3.1.

LEMMA 3.3. *Let $G$ be a digraph satisfying the two conditions of Theorem 3.1. Let $C$ and $c_0$ be as described in the Assumptions. Then one of the following two situations must occur:*

- *there is a vertex $g_0 \in V(G)$ such that for any oriented path $P$, $(P, i(P)) \to (G, g_0)$ implies $(P, i(P)) \to (C, c_0)$, or*

• *every oriented path $P$ homomorphic to $G$ is also homomorphic to $C \setminus c_0$.*

*Proof.* Suppose the lemma is not true. Then for any vertex $g \in V(G)$, there is an oriented path $P_g$ such that $(P_g, i(P_g)) \to (G, g)$ and $(P_g, i(P_g)) \not\to (C, c_0)$. Also, there is an oriented path $P = [p_1, p_2, \ldots, p_n]$ which is homomorphic to $G$ but not homomorphic to $C \setminus c_0$. Let $h : P \to G$ be a homomorphism. For each $1 \leq j \leq n$, let $P_j$ be an oriented path such that $(P_j, i(P_j)) \to (G, h(p_j))$ and $(P_j, i(P_j)) \not\to (C, c_0)$ (i.e., we write $P_j$ for $P_{h(p_j)}$). Let $T$ be the oriented tree obtained from $P$ by attaching the oriented path $P_j$ to each vertex $p_j$ of $P$, identifying $i(P_j)$ with $p_j$.

Obviously $T \to G$. Thus every oriented path homomorphic to $T$ is also homomorphic to $G$ and, by our assumption, also homomorphic to $C$. By Corollary 2.3 we have $T \to C$.

Let $f : T \to C$ be a homomorphism. Since $P \not\to C \setminus c_0$, there exists $p_j \in P \subset T$ such that $f(p_j) = c_0$. But then $f$ restricted to $P_j \subset T$ is a homomorphism $(P_j, i(P_j)) \to (C, c_0)$, contradicting the assumption that $(P_j, i(P_j)) \not\to (C, c_0)$. This proves Lemma 3.3.

Now suppose that Theorem 3.2 is true and $G$ is a digraph satisfying the conditions of Theorem 3.1. By Lemma 3.3, either there is a vertex $g_0 \in V(G)$ such that $(G, g_0)$ satisfies the conditions of Theorem 3.2, which implies $(G, g_0) \to (C, c_0)$, or every oriented path homomorphic to $G$ is also homomorphic to $C \setminus c_0$, which implies $G \to C \setminus c_0$ by Theorem 1.2. In both cases we have $G \to C$ and, therefore, Theorem 3.2 implies Theorem 3.1.

Next we proceed to prove Theorem 3.2. Since rooted homomorphisms can be composed, it is easy to see that the conditions are necessary for the existence of homomorphisms $(G, g_0) \to (C, c_0)$. Thus suppose the conditions are satisfied.

We first construct an auxiliary digraph $D$ as follows: we take the subpath $[c_0, c_1, c_2, \ldots, c_{m-1}, c_0^1, c_1^1, \ldots, c_{j_0-1}^1]$ of $R^2(C)$ and identify the vertices $c_0$ and $c_0^1$, calling the new vertex $c^*$. Thus $D$ is a copy of the cycle $C$ with an additional oriented path $A = [c^*, c_1^1, \ldots, c_{j_0-1}^1]$ attached to it at vertex $c_0$. The path $A$ is just another copy of the path $[c_0, c_1, c_2, \ldots, c_{j_0-1}]$. Note that $c^*$ has in-degree zero. We define the index function as $\mathrm{Ind}(c_j) = j$ for $1 \leq j \leq m-1, \mathrm{Ind}(c_j^1) = m + j$ for $1 \leq j \leq j_0 - 1$, and $\mathrm{Ind}(c^*) = m$. For $a, b \in V(D)$, we write $a \leq b$ if and only if $\mathrm{Ind}(a) \leq \mathrm{Ind}(b)$ and $a < b$ if $\mathrm{Ind}(a) < \mathrm{Ind}(b)$). We again use the terms "min" and "max" with respect to this order.

Obviously $(D, c^*)$ and $(C, c_0)$ are homomorphically equivalent, i.e., $(D, c^*) \to (C, c_0)$ and $(C, c_0) \to (D, c^*)$. Therefore, $(G, g_0) \to (C, c_0)$ if and only if $(G, g_0) \to (D, c^*)$. Instead of constructing a homomorphism of $(G, g_0) \to (C, c_0)$, we will construct a homomorphism of $(G, g_0) \to (D, c^*)$.

Given a vertex $x \in V(G)$, we let $\mathcal{T}_x$ be the set of triples $(T, t_0, t)$ such that $T$ is an oriented tree, $t_0, t$ are vertices of $T$, and there is a homomorphism $h : T \to G$ with $h(t_0) = g_0$ and $h(t) = x$.

REMARK 3.4. *Let $(G, g_0)$ be a rooted digraph satisfying the conditions of Theorem 3.2. Let $C$ and $c_0$ be as described in the Assumptions. Then for any rooted tree $(T, t_0), (T, t_0) \to (G, g_0)$ implies $(T, t_0) \to (C, c_0)$.*

Indeed, if $(T, t_0) \to (G, g_0)$, then for any oriented path $P$, the existence of a homomorphism $(P, i(P)) \to (T, t_0)$ implies the existence of a homomorphism $(P, i(P)) \to (G, g_0)$ and, therefore, the existence of a homomorphism $(P, i(P)) \to (C, c_0)$ (according to one of the hypotheses). Thus $(T, t_0) \to (C, c_0)$ by Corollary 2.5.

Now we are ready to construct a homomorphism $(G, g_0) \to (D, c^*)$.

Define two mappings $\phi$ and $\psi$ as follows: for $(T, t_0, t) \in \cup_{x \in V(G)} \mathcal{T}_x$, let

$$\phi(T, t_0, t) = \max\{h(t) : h \text{ is a homomorphism } (T, t_0) \to (D, c^*)\},$$

and for $x \in V(G)$, let

$$\psi(x) = \min\{\phi(T, t_0, t) : (T, t_0, t) \in \mathcal{T}_x\}.$$

Note that $\phi$ and $\psi$ are well defined: for any $x$ and any $(T, t_0, t) \in \mathcal{T}_x$ we have $(T, t_0) \to (G, g_0)$, and hence $(T, t_0) \to (C, c_0)$ by the above remark. Thus $(T, t_0) \to (D, c^*)$. It is also clear that for any $x \in V(G)$, the set $\mathcal{T}_x$ is not empty, since $G$ is connected.

We now proceed to prove that $\psi$ is a homomorphism $(G, g_0) \to (D, c^*)$.

First we need some lemmas that will help us restrict the possible images of a vertex of $G$ under a homomorphism of $G$ to $D$.

Let $l(C) = k$. For $x \in V(D)$, let $\lambda(x)$ be the length of the path $[c^*, c_1, c_2, \ldots, x]$ if $x \in C$ and the length of the path $[c^*, c_1^1, \ldots, x]$ if $x \in A$. By the remarks at the beginning of §2, we have that $\lambda(x) > 0$ for all $x \neq c^*$.

Consider any oriented path $P$ and any homomorphism $h : (P, i(P)) \to (D, c^*)$. The image $h(P)$ of $P$ under $h$ is a walk of $D$. Since homomorphism of paths preserves distances, we have $\lambda_P(x) = \lambda_{h(P)}(h(x))$ for any $x \in P$. The walk $h(P)$ may wind around $C$ several times. Since $l(C) = k$, we have $\lambda_{h(P)}(h(x)) = \lambda(h(x)) + tk$ for some integer $t$ ($t$ can be positive, negative, or zero). We state this important fact as a lemma.

LEMMA 3.5. *Let $(D, c^*)$ be the rooted digraph defined above and $k = l(C)$. For any oriented path $P$ and any homomorphism $h : (P, i(P)) \to (D, c^*)$, we have $\lambda_P(x) \equiv \lambda(h(x)) (\text{mod} k)$ for all $x \in P$.*

COROLLARY 3.6. *Suppose that $H$ is a connected digraph and $h_0 \in V(H)$ is a fixed vertex of $H$; suppose further that $h_1 : (H, h_0) \to (D, c^*)$ and $h_2 : (H, h_0) \to (D, c^*)$ are two homomorphisms. Then $\lambda(h_1(x)) \equiv \lambda(h_2(x)) (\text{mod} k)$ for all $x \in H$.*

Note that the path $I = [c^*, c_1, \ldots, c_{j_0}]$ is the only minimal path of length $n$ in $D$ which starts at $c^*$.

LEMMA 3.7. *Let $(D, c^*)$ be the auxiliary digraph constructed from $(C, c_0)$. Let $n$ be the maximum length of a subpath of $C$ and $k$ be the length of $C$. If $X = [x_0, x_1, \ldots, x_t]$ is a minimal walk of $D$ of length $n$ and $\lambda(x_0) \equiv 0 (\text{mod} k)$, then $x_0 = c^*$ and $X \subset I$.*

*Proof.* Let $X = [x_0, x_1, \ldots, x_t]$ be a minimal walk of $D$ of length $n$ with $\lambda(x_0) = 0 (\text{mod} k)$. First we show that $x_0 = c^*$. Otherwise, suppose $x_0 \neq c^*$. It is easy to see (by Lemma 2.1) that $D \setminus c^*$ contains no path of length $n$, and thus we have $c^* = x_j$ for some $0 \leq j \leq t$. Since $c^* \neq x_0$, we have $\lambda(x_0) > 0$. Since $X$ is minimal, $d_X(x_0, c^*) > 0$. However, if $X$ goes in the negative direction of $C$, then $d_X(x_0, c^*) = -\lambda(x_0) < 0$. Therefore $X$ must go in the positive direction of $C$. Thus $0 < d_X(x_0, c^*) = k - \lambda(x_0)$, which implies $\lambda(x_0) < k$. Thus $0 < \lambda(x_0) < k$, contradicting the assumption that $\lambda(x_0) \equiv 0 (\text{mod} k)$. Therefore $x_0 = c^*$. Since $I$ is the only minimal path of length $n$ in $D$ which starts at $c^*$, we see that $X \subset I$.

COROLLARY 3.8. *Suppose that $P$ is an oriented path and $B$ is a minimal subpath of $P$ of length $n$. If there is a homomorphism $h : (P, i(P)) \to (D, c^*)$ such that $h(B) = I$, then for any homomorphism $h' : (P, i(P)) \to (D, c^*)$, we must also have $h'(B) = I$.*

*Proof.* Let $h' : (P, i(P)) \to (D, c^*)$ be a homomorphism. Obviously $h'(B)$ is a minimal walk of $D$ of length $n$. By Corollary 3.6, $\lambda(h'(i(B))) = 0 (\text{mod} k)$. Therefore, $h'(i(B)) = c^*$ and $h'(B) = I$ by Lemma 3.7.

In the following three lemmas, we assume that $xy$ is an edge of $G$. We shall prove that $\psi(x)\psi(y)$ is an edge of $D$.

LEMMA 3.9. *Let $\psi : V(G) \to V(D)$ be the mapping defined just after Remark 3.4. If $xy \in E(G)$ then $\psi(x) \neq \psi(y)$.*

*Proof.* Otherwise, suppose $\psi(x) = \psi(y) = a$. Let $(T', t'_0, t') \in \mathcal{T}_x$ be a triple such that $\phi(T', t'_0, t') = \psi(x)$, and let $(T'', t''_0, t'') \in \mathcal{T}_y$ be a triple such that $\phi(T'', t''_0, t'') = \psi(y)$.

Let $T$ be the tree obtained from the disjoint union of $T'$ and $T''$ by adding the edge from $t'$ to $t''$. Then it is easy to see that $(T, t'_0, t'), (T, t''_0, t') \in \mathcal{T}_z$ and $(T, t'_0, t''), (T, t''_0, t'') \in \mathcal{T}_y$. By the definition of $\psi(y)$ we have $\phi(T, t''_0, t'') \geq \psi(y) = \phi(T'', t''_0, t'')$. Since any homomorphism $h : (T, t''_0) \to (D, c^*)$ restricted to $T''$ is a homomorphism from $(T'', t''_0)$ to $(D, c^*)$, we have $\phi(T, t''_0, t'') \leq \phi(T'', t''_0, t'')$. Therefore $\phi(T, t''_0, t'') = \phi(T'', t''_0, t'') = a$. Similarly, $\phi(T, t'_0, t') = \phi(T', t'_0, t') = a$. Let $h : T \to D$ be a homomorphism such that $h(t'') = a$, and let $h' : T \to D$ be a homomorphism such that $h'(t') = a$. Suppose $h(t') = b$ and $h'(t'') = c$. Then $(b, a) \in E(D)$ and $(a, c) \in E(D)$, because $(t', t'') \in E(T)$ and $h, h'$ are homomorphisms. Therefore, $a$ has positive in-degree and positive out-degree. In particular, $a \neq c^*$ and, therefore, $a$ has in-degree one and out-degree one.

To obtain the final contradiction, we consider two cases.

*Case* 1. Suppose that $a = c_1$. Let $h : (T, t''_0) \to (D, c^*)$ be a homomorphism such that $h(t'') = \phi(T, t''_0, t'') = c_1$. Then $h(t') = c^*$. Delete from $T$ all the vertices $t$ such that $h(t) = c^*$, and let $B$ be the component which contains $t''$ after the deletion. If $c_{j_0} \notin h(B)$ then $h(B)$ is contained in $I \setminus c_{j_0}$. Now define a mapping $h' : T \to D$ as follows:

$$h'(t) = h(t) \quad \text{if } t \notin B, \quad \text{and}$$
$$h'(t) = c_j^1 \quad \text{if } t \in B \text{ and } h(t) = c_j.$$

It is easy to see that $h'$ is a homomorphism and $h'(t''_0) = c^*, h'(t'') = c_1^1$. However, $\text{Ind}(c_1^1) = m + 1 > \text{Ind}(c_1)$. This contradicts the fact that $\phi(T, t''_0, t'') = c_1$. Therefore $c_{j_0} \in h(B)$. Let $s \in B$ be a vertex such that $h(s) = c_{j_0}$, and for which the unique path $P$ connecting $t'$ and $s$ in $T$ has no other vertex $u$ with $h(u) = c_{j_0}$. Thus $P$ is a minimal path of length $n$.

Now let $h'' : (T, t'_0) \to (D, c^*)$ be a homomorphism such that $h''(t') = \phi(T, t'_0, t') = c_1$. By Corollary 3.8, $h''(P) = I$, which implies that $h''(t') = c^*$, a contradiction.

*Case* 2. Assume that $a \neq c_1$. Then for the two neighbours $b$ and $c$ of $a$ we have $b < a < c$. As $a$ has in-degree one and out-degree one, assume that $ba \in E(D)$ and $ac \in E(D)$. (A similar argument applies to the case $ab \in E(D)$ and $ca \in E(D)$).

By the definition of $\psi(x)$, we have $\phi(T, t''_0, t') \geq \psi(x) = a$. Let $h : (T, t''_0) \to (D, c*)$ be a homomorphism such that $h(t') = \phi(T, t''_0, t')$. Then $(h(t'), h(t'')) \in E(D)$ implies that $h(t'') > a$. This contradicts the fact that $\phi(T, t''_0, t'') = a$ and proves that $\psi(x) \neq \psi(y)$.

LEMMA 3.10. *Let $\psi : V(G) \to V(D)$ be the mapping defined just after Remark 3.4. If $xy \in E(G)$ and $\psi(x) < \psi(y)$ then $\psi(x)\psi(y) \in E(D)$.*

*Proof.* We proceed as in the proof of Lemma 3.9, constructing $(T', t'_0, t')$, $(T'', t''_0, t'')$, and $T$. Recall that $\phi(T, t'_0, t') = \psi(x)$ and $\phi(T, t'_0, t'') \geq \psi(y)$. Let $h : (T, t'_0) \to (D, c^*)$ be a homomorphism such that $h(t'') = \phi(T, t'_0, t'') \geq \psi(y)$. By the definition of $\phi$ we have $h(t') \leq \phi(T, t'_0, t') = \psi(x)$. Therefore $h(t') \leq \psi(x) < \psi(y) \leq h(t'')$. However, $h(t')h(t'') \in E(D)$, and hence $\text{Ind}(h(t'')) \leq \text{Ind}(h(t')) + 1$. Therefore, we must have $h(t') = \psi(x), h(t'') = \psi(y)$, and thus $\psi(x)\psi(y) \in E(D)$.

LEMMA 3.11. *Let $\psi : V(G) \to V(D)$ be the mapping defined just after Remark 3.4. If $xy \in E(G)$ and $\psi(x) > \psi(y)$, then $\psi(x)\psi(y) \in E(D)$.*

*Proof.* Again, let $(T', t_0', t')$, $(T'', t_0'', t'')$, and $T$ be defined as in the proof of Lemma 3.9; thus we again have $\phi(T, t_0'', t'') = \psi(y)$ and $\phi(T, t_0'', t') \geq \psi(x)$. Let $h : (T, t_0'') \to (D, c^*)$ be a homomorphism such that $h(t') = \phi(T, t_0'', t') \geq \psi(x)$. As in the proof above, we have $h(t'') \leq \psi(y) < \psi(x) \leq h(t')$ and $h(t')h(t'') \in E(D)$. If $\mathrm{Ind}(h(t')) \leq \mathrm{Ind}(h(t'')) + 1$ then the same argument shows that $\psi(x)\psi(y) = h(t')h(t'') \in E(D)$. Otherwise, we must have $h(t') = c^*, h(t'') = c_1$, and $c_1 \leq \psi(y) < \psi(x) \leq c^*$. In what follows we prove that in this case we must have $\psi(y) = c_1$ and $\psi(x) = c^*$ and, therefore, $\psi(x)\psi(y) \in E(D)$.

Let $K = \{t \in T : h(t) = c^*\}$ and let $B$ be the component of $T \setminus K$ which contains $t''$. With the same argument as in Case 1 of the proof of Lemma 3.9, we find a vertex $s \in B$ such that the unique path $P$ in $T$ joining $t'$ and $s$ is a minimal path of length $n$. Let $h' : (T, t_0') \to (D, c^*)$ be a homomorphism such that $h'(t') = \phi(T, t_0', t') = \psi(x)$. Since $h(t') = c^*$, we have $h(P) = I$. By Corollary 3.8, we have $h'(P) = I$ as well. Therefore $h'(t') = c^* = \psi(x)$.

To show that $\psi(y) = c_1$, we let $h'' : (T, t_0'') \to (D, c^*)$ be a homomorphism such that $h''(t'') = \phi(T, t_0'', t'') = \psi(y)$. Again, by Corollary 3.8, $h''(P) = I$ and $h''(t') = c^*$. Observe that $t''$ is the vertex adjacent to $t'$ in $P$; we have $h''(t'') = c_1$. Now Lemma 3.11 is proved.

This completes the proof that $\psi : G \to D$ is a homomorphism. To complete the proof of Theorem 3.2, we still need the following lemma.

LEMMA 3.12. *The homomorphism $\psi$ satisfies $\psi(g_0) = c^*$.*

*Proof.* First, we observe that $\psi(g_0) \leq c^*$, because for the tree $T^*$ consisting of a single vertex $t_0$, we have $(T^*, t_0, t_0) \in \mathcal{T}_{g_0}$ and $\phi(T^*, t_0, t_0) = c^*$.

Assume that $\psi(g_0) < c^*$, and let $(T, t_0, t) \in \mathcal{T}_{g_0}$ be a triple such that $\phi(T, t_0, t) = \psi(g_0)$. Thus, there exists a homomorphism $h_1 : T \to G$ with $h(t_0) = h(t) = g_0$ and a homomorphism $h_2 : T \to D$ with $h_2(t_0) = c^*$ and $h_2(t) = \phi(T, t_0, t) = \psi(g_0)$. By identifying $c_i^1$ with $c_i$, we view $h_2$ as a homomorphism of $T$ to $C$ with $h_2(t_0) = c_0$. We shall proceed to construct a homomorphism $h : T \to C$ with $h(t_0) = h(t) = c_0$, which can be viewed as a homomorphism of $T$ to $D$ with $h(t_0) = h(t) = c^*$, in contradiction to $\phi(T, t_0, t) < c^*$.

To construct $h$ we shall use $h_1, h_2$, and a third homomorphism $h_3 : T \to C$ with $h_3(t) = c_0$. Such a homomorphism exists by Remark 3.4, since $(T, t) \to (G, g_0)$ via $h_1$. We shall construct $h$ by letting it equal $h_2$ on part of the tree $T$ and $h_3$ on the rest of the tree $T$. For this purpose we need the following claim.

*Claim.* Let $P$ be the unique path of $T$ connecting $t_0$ to $t$. Then there is a vertex $t^*$ of $P$ such that $h_2(t^*) = h_3(t^*)$.

Since $h_1(P)$ is a closed walk of $G$, we have $l(P) = l(h_1(P)) = sk$ for an integer $q$. Now we consider four cases.

*Case* 1. Suppose that $s \leq -1$. Since $h_2(t_0) = c_0$ and $l(h_2(P)) \leq -k$, it is easy to see from Lemma 2.1 and the minimality of $I$ that $h_2(P)$ must wind around $C$ in the negative direction at least once. Therefore, there is a minimal subpath $X$ of $P$ such that $h_2(X) = I$. By Corollary 3.8, $h_3(X) = I$ and $h_2(i(X)) = h_3(i(X)) = c_0$. Thus, in this case we let $t^* = i(X)$.

This argument also shows that the claim follows whenever there exists a vertex $v \in P$ which has level less than or equal to $-k$.

*Case* 2. Suppose that $s \geq 2$. The composition $\xi = \psi \circ h_1$ of the two homomorphisms $\psi$ and $h_1$ is a homomorphism of $T$ to $D$. Since $\xi(t_0) = \xi(t) = \psi(g_0)$ and $C$ is a cycle of length $k$, we see that $\xi(P)$ is a closed walk of $C$ which winds around $C$ at least twice. This implies that there is a minimal subpath $X$ such that $\xi(X) = I$, and again, by Corollary 3.8, $h(i(X)) = h'(i(X))$. In this case we also let $t^*$ be $i(X)$.

*Case* 3. Suppose that $s = 0$. We choose a large integer $q$ and consider the induced homomorphisms of $h_2$ and $h_3$, namely $h'_2, h'_3 : P \to R^{2q}(C)$ (cf. §2). As $h_2(t_0) = h_3(t) = c_0$, we have $h'_2(t_0) = c_0^q$ and $h'_3(t) = c_0^a$ for some $a$. Let $h''_3 : P \to R^{2q}(C)$ be the homomorphism defined as $h''_3(x) = c_j^{r+q-a}$ if $h'_3(x) = c_j^r$. In other words, $h''_3$ is obtained from $h'_3$ by shifting the image so that $t$ is sent to $c_0^q$. It is obvious that if $h'_2(t^*) = h''_3(t^*)$ for some $t^* \in P$, then $h_2(t^*) = h_3(t^*)$.

Note that Lemma 2.1 implies that $d_{R^{2q}(C)}(c_0^q, v) > 0$ for any $v \in R^{2q}(C)$ with $v > c_0^q$ (recall that the order is defined by the index function). Since $h'_2(P)$ and $h''_3(P)$ are walks of $R^{2q}(C)$ of length zero, we must have $h'_2(t) < c_0^q$ and $h''_3(t_0) < c_0^q$. Thus we have $h'_2(t_0) > h''_3(t_0)$ and $h'_2(t) < h''_3(t)$. Let $x$ be the last vertex of $P$ such that $h'_2(x) \geq h''_3(x)$, and let $y$ be the next vertex of $P$. Thus we have $h'_2(y) < h''_3(y)$. If $h'_2(x) = h''_3(x)$ then we let $t^* = x$ and the claim follows. Assume that $h'_2(x) > h''_3(x)$. Observe that either $xy$ is an edge of $P$, which implies that $h'_2(x)h'_2(y)$ and $h''_3(x)h''_3(y)$ are edges of $R^q(C)$, or $yx$ is an edge of $P$, which implies that $h'_2(y)h'_2(x)$ and $h''_3(y)h''_3(x)$ are edges of $R^q(C)$. In any case, we have $\mathrm{Ind}(h'_2(x)) \leq \mathrm{Ind}(h'_2(y))+1$ and $\mathrm{Ind}(h''_3(y)) \leq \mathrm{Ind}(h''_3(x)) + 1$. Therefore, we must have $h'_2(x) = h''_3(y)$ and $h'_2(y) = h''_3(x)$. This is a contradiction, since $R^{2q}(C)$ has no digons.

*Case* 4. Suppose that $s = 1$. Again, we let $\xi = \psi \circ h_1$ be the composition of the two homomorphisms $\psi$ and $h_1$. Since $\xi(P)$ is a closed walk of $C$ of length $k$, it winds around $C$ exactly once in the positive direction of $C$. Therefore, the induced homomorphism $\xi' : P \to R^{2q}(C)$ satisfies $\xi'(t_0) = c_r^q$ and $\xi'(t) = c_r^{q+1}$, where $c_r = \xi(t_0) = \psi(g_0) = h_2(t)$. In particular, there is a vertex $v \in P$ such that $\xi'(v) = c_0^{q+1}$.

Recall that $h_2(t_0) = c_0$. Suppose $h_2(P)$ goes in the negative direction of $C$. Then the induced homomorphism (for some large $q$) $h'_2 : P \to R^{2q}(C)$ satisfies $h'_2(t_0) = c_0^q$ and $h'_2(t) = c_r^s$ for some $s \leq q-1$. Thus the distance $d_{R^{2q}(C)}(c_0^q, c_r^s) = l(P) = k$. However, $d_{R^{2q}(C)}(c_0^q, c_0^{s+1}) = (-k)\cdot(q-(s+1)) \leq 0$. Therefore, $d_{R^{2q}(C)}(c_0^{s+1}, c_r^s) \geq k$. This implies that $d_{R^{2q}(C)}(c_r^q, c_0^{q+1}) \leq -k$, and hence the level of the vertex $v$ in $P$ is less than or equal to $-k$. We have already shown that in this case the claim is true (see the remark at the end of the proof of Case 1).

Thus we may assume that $h_2(P)$ goes in the positive direction of $C$, i.e., $h'_2(t_0) = c_0^q$ and $h'_2(t) > c_0^q$. If there is a vertex $v$ of the tree $T$ such that $h'_2(v) = c_{j_0}^q$, then the path $P'$ of $T$ connecting $t_0$ to $v$ has length $n$. Thus it contains a minimal subpath $B$ of length $n$ such that $h'_2(B) = [c_0^q, c_1^1, \ldots, c_{j_0}^q]$. This implies that $h_2(B) = I$ and, therefore, $h_3(B) = I$ by Corollary 3.8. In this case we let $t^* = i(B)$. On the other hand, suppose that there is no vertex $v$ of $T$ such that $h'_2(v) = c_{j_0}^q$. We delete all the vertices $x$ of $T$ such that $h_2(x) = c_0$. Let $B$ be the component which contains $t$ (recall that $h_2(t) = \psi(g_0) \neq c^*$). Then there is no vertex $x \in B$ such that $h_2(x) = c_{j_0}$, and hence $h_2(B)$ is contained in $I \backslash c_{j_0}$. As in the proof of Lemma 3.9, we can shift the image of $B$ to $A$ (recall that $A$ is just another copy of $I \setminus \{c_{j_0}\}$ attached to $c^*$ in the auxiliary digraph $D$). The new homomorphism shows that $\phi(T, t_0, t) \geq c^*$, contradicting our assumption.

Now the claim is proved and we can define a homomorphism $h'' : (T, t_0) \to (D, c^*)$ as follows: let $B$ be the component of $T \setminus t^*$ which contains $t$, and let

$$\begin{aligned} h''(x) &= h'(x) \quad \text{if } x \in B, \\ h''(x) &= h(x) \quad \text{if } x \notin B. \end{aligned}$$

Then $h''$ is obviously a homomorphism and $h''(t_0) = h'(t) = c^*$. Therefore, $\psi(g_0) = c^*$

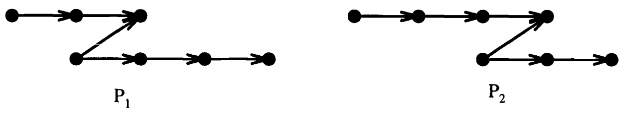


FIG. 1. *Example paths $P_1$ and $P_2$.*

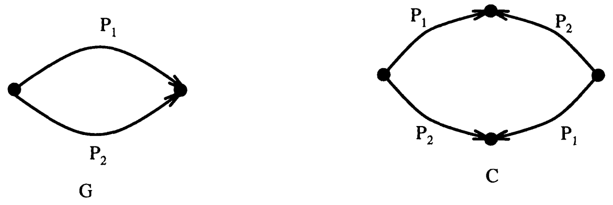


FIG. 2. *Cycles G and C.*

and $(G, g_0) \to (D, c^*)$. This completes the proof of Lemma 3.12, as well as that of Theorem 3.2.

**4. General oriented cycles.** We first construct an example to show that $(D'_C)$ does not always hold.

Let $P_1$ and $P_2$ be two minimal paths of the same length such that $P_1 \not\to P_2$ and $P_2 \not\to P_1$. Such paths are easy to construct; Fig. 1 gives one such pair of paths.

Let $G$ be obtained by identifying $t(P_1)$ with $t(P_2)$ and $i(P_1)$ with $i(P_2)$ (cf. the left graph in Fig. 2, where the directed edge labeled $P_i$ represents the path $P_i$). Let $C$ be obtained in a similar way from two copies of $P_1$ and two copies of $P_2$, as depicted in the right graph of Fig. 2.

It is easy to see that $G \not\to C$, yet $G$ satisfies the hypotheses of $(D'_C)$. Thus $(D'_C)$ does not hold for this cycle $C$. Note that this does not show that $(D_C)$ fails for $C$, since $G$ does not satisfy the hypotheses of $(D_C)$. However, Theorem 4.1 below suggests that $(D_C)$ may fail for any cycle $C$ such that $C$-col is $NP$-complete; cf. Fig. 3.

THEOREM 4.1. *If $(D_C)$ holds for an oriented cycle $C$ then $C$-col is in $NP \cap$co$NP$.*

*Proof.* Obviously each problem $C$-col is in the class $NP$. Let $C$ be a fixed oriented cycle for which $(D_C)$ holds, i.e., such that a digraph $G$ is homomorphic to $C$ if and only if every cycle homomorphic to $G$ is also homomorphic to $C$.

In order to prove that $C$-col is also in co$NP$, we shall prove the following two statements (in Lemmas 4.2 and 4.3).

(1) There is an algorithm which decides whether or not $X \to H$ in time $O(|E(X)| \cdot |E(H)| \cdot |V(H)|)$ for any oriented cycle $X$ and any digraph $H$.

(2) Let $H$ be a fixed digraph. If there is a cycle $X$ which is homomorphic to a digraph $G$ but not to $H$, then there is such a cycle $X'$ with $O(|V(G)|)$ edges.

It is not difficult to see that these two statements imply that $C$-col is in the class co$NP$. Indeed, if $G$ is a digraph with $G \not\to C$, then there is a cycle $X$ which is homomorphic to $G$ but not to $C$. Thus by (2), there is such a cycle $X'$ with $O(|V(G)|)$ edges. By (1), it can be verified in time $O(|V(G)| \cdot |E(G)|^2)$ that $X'$ is indeed homomorphic to $G$ and not to $C$ (observe that the size of $C$ is a constant).

Let $W = [w_0, w_1, \ldots, w_{m-1}, w_m]$ be an oriented path, $H$ be any digraph, and $h_0 \in V(H)$ be a fixed vertex of $H$. The *canonical labeling* of $W$ by $(H, h_0)$ is the unique mapping $\lambda$ of $W$ to the subsets of $V(H)$ for which

$$\begin{aligned}\Lambda(w_0) &= \{h_0\},\\ \Lambda(w_{i+1}) &= \{v \in V(H) : uv \in E(H) \text{ for some } u \in \Lambda(w_i)\}\\ &\quad \text{if } w_i w_{i+1} \in E(W),\\ \Lambda(w_{i+1}) &= \{v \in V(H) : vu \in E(H) \text{ for some } u \in \Lambda(w_i)\}\\ &\quad \text{if } w_{i+1} w_i \in E(W).\end{aligned}$$

LEMMA 4.2. *Let $W = [w_0, w_1, \ldots, w_{m-1}, w_m]$ be an oriented path and $X$ be the oriented cycle obtained from $W$ by identifying $w_0$ with $w_m$. Let $H$ be any digraph and let $h_0$ be a fixed vertex of $H$. Then $(X, w_0) \rightarrow (H, h_0)$ if and only if $h_0 \in \Lambda(w_m)$ in the canonical labeling of $W$ by $H$.*

*Proof.* Suppose $h : (X, w_0) \rightarrow (H, h_0)$ is a homomorphism. Then it is easy to show by induction on $j$ that $h(w_j) \in \Lambda(w_j)$ for all $0 \leq j \leq m$. Since $h(w_m) = h(w_0) = h_0$, we have $h_0 \in \Lambda(w_m)$.

On the other hand, suppose $h_0 \in \Lambda(w_m)$ in the canonical labeling of $W$ by $(H, h_0)$. A homomorphism $h : (X, w_0) \rightarrow (H, h_0)$ can be constructed as follows: Let $h(w_m) = h(w_0) = h_0$. If $h(w_j) = v_j \in \Lambda(w_j)$ has been chosen, then let $h(w_{j-1}) = v_{j-1}$, where $v_{j-1}$ is an element of $\Lambda(w_{j-1})$ such that either $(v_{j-1}, v_j) \in E(H)$ or $(v_j, v_{j-1}) \in E(H)$, according to whether $(w_{j-1}, w_j) \in E(W)$ or $(w_j, w_{j-1}) \in E(W)$. Such an element exists by the definition of the canonical labeling. It is clear that the mapping $h$ is a homomorphism.

The canonical labeling of $W$ by $(H, h_0)$ can be found in time $O(|E(W)| \cdot |E(H)|)$. Thus it can be determined in time $O(|E(W)| \cdot |E(H)|)$ whether or not $(X, w_0) \rightarrow (H, h_0)$. In order to determine whether or not $W \rightarrow H$, it is enough to determine whether or not $(W, w_0) \rightarrow (H, h)$ for some $h \in V(H)$. Therefore, it suffices to find the canonical labeling of $W$ by $(H, h)$ for each of the vertices $h \in V(H)$. Thus it can be determined in time $O(|E(X)| \cdot |E(H)| \cdot |V(H)|)$ whether or not $X \rightarrow H$.

LEMMA 4.3. *Let $V(H) = k$. If there exists an oriented cycle $X$ homomorphic to $G$ but not to $H$, then there exists such a cycle $X$ with $|V(X)| \leq 2^{k^2} \cdot |V(G)|$.*

*Proof.* Suppose that $X$ is obtained from the oriented path $W = [w_0, w_1, \ldots, w_m]$ by identifying $w_0$ with $w_m$, and $X \rightarrow G$ and $X \not\rightarrow H$. Let $f : X \rightarrow G$ be a homomorphism. For each vertex $h \in V(H)$, let $\Lambda_h$ be the canonical labeling of $W$ by $(H, h)$. By the previous lemma, $h \notin \Lambda_h(w_m)$ for any $h \in V(H)$ (otherwise we would have $(X, w_0) \rightarrow (H, h)$, and hence $X \rightarrow H$). If $m > 2^{k^2} \cdot |V(G)|$, then by the pigeon hole principle there are two vertices $w_i, w_j$ of $W$(with $i < j$) such that $f(w_i) = f(w_j)$ and $\Lambda_h(w_i) = \Lambda_h(w_j)$ for all $h \in V(H)$. (The mappings $f$ and $\{\Lambda_h : h \in V(H)\}$ can be viewed as a single mapping of $W$ into the set $V(G) \times 2^{V(H)} \times 2^{V(H)} \times \cdots \times 2^{V(H)}$ of size $2^{k^2} \cdot |V(G)|$.) Let $W' = [w_0, w_1, \ldots, w_i, w_{j+1}, \ldots, w_m]$ (i.e., $W'$ is obtained from $W$ by deleting all the vertices $w_{i+1}, \ldots, w_{j-1}$ and identifying $w_i$ with $w_j$), and let $X'$ be the cycle obtained from $W'$ by identifying $w_0$ with $w_m$. Then, obviously, $W' \rightarrow G$, and for the canonical labeling $\Lambda'_h$ of $W'$ by $(H, h)$ we have $\Lambda'_h(w_t) = \Lambda_h(w_t)$ for all $w_t \in W'$ and all $h \in V(H)$. Therefore, $h \notin \Lambda'_h(w_m)$ for all $h \in V(H)$ and $X' \not\rightarrow H$.

COROLLARY 4.4. *Let $G$ be a digraph and $C$ be an oriented cycle with $k$ edges. If there exists a cycle homomorphic to $G$ but not to $C$, then there exists such a cycle $X$ with $|V(X)| \leq 2^{k^2} \cdot |V(G)|$.*

Since $C$ is fixed, $2^{k^2}$ is a constant. So the size of $X$ is $O(|V(G)|)$. Thus we have proved statements (1) and (2), as well as Theorem 4.1.

It follows from Corollary 4.4 and Theorem 3.1 that $C$-col is in $NP \cap \text{co}NP$ whenever $C$ is unbalanced. In fact, it follows from [8], [30] that $C$-col is polynomial

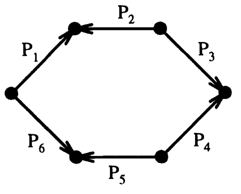


FIG. 3. *Gutjahr's cycle $C$ for which $C$-col is $NP$-complete.*

for unbalanced cycles $C$. This can also be derived from Theorem 3.1 using a technique explained in [16]. On the other hand, suppose that $P_i$ $(i = 1,2,3,4,5,6)$ are minimal oriented paths of length $n$ such that $P_i \not\to P_j$ whenever $i \neq j$. (Such paths are easy to construct using the technique apparent in Fig. 1.) Gutjahr proved that $C$-col is $NP$-complete for the balanced cycle $C$ depicted in Fig. 3 [8].

**5. Remark on new results.** In [16], we shall argue that statements like $(D_C)$ can be viewed as "duality" properties of graph homomorphisms. In the terminology of [16], our main result here asserts that unbalanced cycles have *cycle duality*, and Theorem 1.2 asserts that oriented paths have *path duality*. We have recently considered more general duality statements: A digraph $H$ is said to have *treewidth-$k$ duality* if a digraph $G$ is homomorphic to $H$ if and only if every oriented partial $k$-tree homomorphic to $G$ is also homomorphic to $H$. Since oriented paths are partial 1-trees and oriented cycles are partial 2-trees, we have many examples of graphs with treewidth-$k$ duality. It is proved in [16] that if $H$ has treewidth-$k$ duality (for any $k$), then $H$-col is polynomial. This allows us to conclude, from our main theorem, that $C$-col is polynomial for each unbalanced cycle $C$. Polynomial algorithms for this problem have previously been proposed by X. Zhu [30] (motivated by the main technique of this paper), and independently by W. Gutjahr [8]. (It also allows us to conclude from Theorem 1.2 that $P$-col is polynomial for each oriented path $P$; this was first proved in [9].) T. Feder and M. Vardi have recently shown that the class of graphs with treewidth-$k$ duality corresponds exactly with $H$-col problems they call *of bounded width* [6], which admit polynomial Datalog algorithms. Most recently, Feder has shown that for all oriented cycles $C$, the problem $C$-col is either polynomial or $NP$-complete [7]. J. Nešetřil and X. Zhu [25] have shown that there exist balanced oriented cycles which have no treewidth-$k$ duality for any integer $k$. This implies, in particular, that $(D_C)$ does not hold for these cycles.

## REFERENCES

[1] J. BANG-JENSEN AND P. HELL, *On the effect of two cycles on the complexity of colouring*, Discrete Appl. Math., 26 (1990), pp. 1–23.

[2] J. BANG-JENSEN, P. HELL, AND G. MACGILLIVRAY, *The complexity of colouring by semicomplete digraphs*, SIAM J. Discrete Math., 1 (1988), pp. 281–298.

[3] ———, *On the complexity of colouring by superdigraphs of bipartite graphs*, Discrete Math., 109 (1992), pp. 27–44.

[4] ———, *Hereditarily hard colouring problems*, Discrete Math., to appear.

[5] S. BLOOM AND S. BURR, *On unavoidable digraphs in orientation of graphs*, J. Graph Theory, 11 (1987), pp. 453–462.

[6] T. FEDER AND M. VARDI, *Monotone monadic SNP and constraint satisfaction*, Proc. 25th ACM STOC, 1993, pp. 612–622.

[7] T. FEDER, *Classification of homomorphisms to oriented cycles*, 1993, manuscript.

[8] W. GUTJAHR, *Graph colorings*, Ph.D. thesis, Free University, Berlin, 1991.

[9] W. GUTJAHR, E. WELZL, AND G. WOEGINGER, *Polynomial graph colourings*, Discrete Appl.

Math., 35 (1992), pp. 29–46.

[10] R. HÄGGKVIST ET AL., *On multiplicative graphs and the product conjecture*, Combinatorica, 8 (1988), pp. 71–81.

[11] R. HÄGGKVIST AND P. HELL, *On A-mote universal graphs*, European J. Combin., 14 (1993), pp. 23–27.

[12] F. HARARY, *Graph Theory*, Addison–Wesley, Reading, MA, 1969.

[13] P. HELL, *An Introduction to the category of graphs*, Annals New York Acad. Sci., 328 (1979), pp. 120–136.

[14] P. HELL AND J. NEŠETŘIL, *On the complexity of H-colouring*, J. Combin. Theory Ser. B, 48 (1990), pp. 92–110.

[15] ———, *Homomorphisms of graphs and their orientations*, Monatsh. Math., 85 (1978), pp. 39–48.

[16] P. HELL, J. NEŠETŘIL, AND X. ZHU, *Duality and polynomial testing of tree homomorphisms*, Trans. Amer. Math. Soc., to appear.

[17] ———, *Duality of graph homomorphisms*, in Combinatorics, Paul Erdös Is Eighty, vol. 2, Bolyai Society for Mathematical Studies, 1994.

[18] P. HELL AND X. ZHU, *Homomorphisms to oriented paths*, Discrete Math., 132 (1994), pp. 107–114.

[19] P. HELL, H. ZHOU, AND X. ZHU, *Homomorphisms to oriented cycles*, Combinatorica, 13 (1993), pp. 421–433.

[20] ———, *Multiplicativity of oriented cycles*, J. Combin. Theory Ser. B, 60 (1994), pp. 239–253.

[21] P. KOMÁREK, *Some new good characterizations of directed graphs*, Časopis Pro Pěstovani Matematiky (Prague), 51 (1984), pp. 348–354.

[22] G. MACGILLIVRAY, *On the complexity of colouring by vertex-transitive and arc-transitive digraphs*, SIAM J. Discrete Math., 4 (1991), pp. 397–408.

[23] H. A. MAURER, J. H. SUDBOROUGH, AND E. WELZL, *On the complexity of the general coloring problem*, Inform. and Control, 51 (1981), pp. 123–145.

[24] J. NEŠETŘIL AND A. PULTR, *On classes of relations and graphs determined by subobjects and factorobjects*, Discrete Math., 22 (1978), pp. 287–300.

[25] J. NEŠETŘIL AND X. ZHU, *On bounded treewidth duality of graphs*, 1994, manuscript.

[26] E. WELZL, *Symmetric graphs and interpretations*, J. Combin. Theory Ser. B, 37 (1984), pp. 235–244.

[27] H. ZHOU, *Homomorphism properties of graph products*, Ph.D. thesis, Simon Fraser University, Burnaby, British Columbia, Canada, 1988.

[28] ———, *Characterization of the homomorphic preimages of certain oriented cycles*, SIAM J. Discrete Math., 6 (1993), pp. 87–99.

[29] X. ZHU, *Multiplicative structures*, Ph.D. thesis, The University of Calgary, Calgary, Alberta, Canada, 1990.

[30] X. ZHU, *A polynomial algorithm for homomorphisms to unbalanced oriented cycles*, 1991, manuscript.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 223–250, May 1995

# CHERNOFF–HOEFFDING BOUNDS FOR APPLICATIONS WITH LIMITED INDEPENDENCE *

JEANETTE P. SCHMIDT†, ALAN SIEGEL‡, AND ARAVIND SRINIVASAN§

**Abstract.** Chernoff–Hoeffding (CH) bounds are fundamental tools used in bounding the tail probabilities of the sums of bounded and independent random variables (r.v.'s). We present a simple technique that gives slightly better bounds than these and that more importantly requires only *limited independence* among the random variables, thereby importing a variety of standard results to the case of limited independence for free. Additional methods are also presented, and the aggregate results are sharp and provide a better understanding of the proof techniques behind these bounds. These results also yield improved bounds for various tail probability distributions and enable improved approximation algorithms for jobshop scheduling. The limited independence result implies that a reduced amount and weaker sources of randomness are sufficient for randomized algorithms whose analyses use the CH bounds, e.g., the analysis of randomized algorithms for random sampling and oblivious packet routing.

**Key words.** Chernoff–Hoeffding bounds, large deviations, randomized algorithms, derandomization, limited independence, correlation inequalities, deterministic simulation

**AMS subject classifications.** 60F10, 60E15, 60G50, 60C05, 68Q99, 68R99, 62H10, 05A10

## 1. Introduction.

The most fundamental tools used in bounding the tail probabilities of the sums of bounded and independent random variables are based on techniques initiated by Chernoff [11] and generalized by Hoeffding [17] more than 30 years ago. They are frequently used in the design and analysis of randomized algorithms and derandomization and in the probabilistic method. We present a simple method which generalizes somewhat the classical method for proving the Chernoff Hoeffding (CH) bounds in the case of bounded random variables confined to the interval [0, 1]. More importantly, this approach requires only *limited independence* among the random variables (r.v.'s) and thereby imports a variety of standard results to the case of limited independence for free. This and related bounds lead to a variety of applications ranging from improved bounds for tail probability distributions to new algorithmic results.

The limited independence result implies that sources of randomness that are

* Received by the editors October 13, 1992; accepted for publication (in revised form) April 14, 1994. A preliminary version of this work appeared in *ACM–SIAM Symposium on Discrete Algorithms*, 1993.

† Computer Science Department, Polytechnic University, Brooklyn, New York 11201. The research of this author was supported in part by National Science Foundation grants CCR-9110255 and CCR-9305873 and the Center for Advanced Technology of the New York State Science and Technology Foundation.

‡ Computer Science Department, Courant Institute of Mathematical Science, New York University, New York, New York 10012. The research of this author was supported in part by National Science Foundation grants CCR-8906949 and CCR-8902221.

§ Department of Computer Science, Cornell University, Ithaca, New York 14853. Current addresses, Center for Discrete Mathematics and Theoretical Computing (DIMACS), Rutgers University, Piscataway, New Jersey 08855, and School of Mathematics, Institute for Advanced Study, Princeton, New Jersey 08540. The research of this author was supported in part by the Army Research Office through the Mathematical Sciences Institute of Cornell University and by National Science Foundation Presidential Young Investigator award CCR-89-96272 with matching support from United Parcel Service and Sun Microsystems.

weaker than the standard model of unbiased and independent bits are sufficient for any algorithm whose analysis uses the CH bounds. It also provides a better understanding of the proof techniques behind these bounds and gives improved bounds for various tail probability distributions. Via standard techniques, it leads to a simple analysis of algorithms for such classical problems as random sampling. The formulation also leads to approximation algorithms with better approximation guarantees for certain problems.

Given $n$ r.v.'s $X_1, X_2, \ldots, X_n$, suppose we want to upper bound the "upper tail" probability $\Pr(X \geq a)$, where $X \doteq \sum_{i=1}^n X_i, \mu \doteq E[X], a = \mu(1+\delta)$, and $\delta > 0$. The classical idea behind the CH bounds (see, for instance, Chernoff [11], Hoeffding [17], Raghavan [35], and Alon, Spencer, and Erdös [3]) is as follows. For any fixed $t > 0, \Pr(X \geq a) = \Pr(e^{tX} \geq e^{at}) \leq E[e^{tX}]/e^{at}$ by Markov's inequality. Computing an upper bound $u(t)$ on $E[e^{tX}]$ and minimizing $u(t)/e^{at}$ over $t > 0$ gives an upper bound for $\Pr(X \geq a)$.

An important situation in computation is the one in which $X_i \in \{0,1\}, i = 1, 2, \ldots, n$. For this case, we construct a class of functions of $X$ that is as easy to analyze and includes the class $\{e^{tX} : t > 0\}$ and do the above minimization over this class. In the process, we discover that $X_1, X_2, \ldots, X_n$ *need only be* $h(n, \mu, \delta)$*-wise independent* for a suitably defined function $h(\cdot,\cdot,\cdot)$, which is typically much less than $n$ for many algorithms; recall that a set of r.v.'s $V$ exhibit $k$-wise independence if any subset of $k$ or fewer r.v.'s from $V$ are jointly independent, which is to say that their joint probability distribution function is simply the product of the individual distributions. One reason for the use of the $e^{tX}$ function in the classical methods is that $E[e^{tX}]$ generates all higher moments of $X$; using only a constant number of higher moments, for instance, gives weak bounds. However, in the binary case, the first $n$ moments are sufficient to generate *all* higher moments, which motivates our method. Interestingly, this formulation can also be applied to general $X_i$ that take arbitrary values in the interval [0, 1], even though it is *not* true that the first $n$ moments of $X = \sum_{i=1}^n X_i$ determine all higher ones.

The results have many applications to tail probability distributions. They imply similar limited independence results when $X_1, X_2, \ldots, X_n$ take values in the interval [0, 1]; this can be extended to bounded r.v.'s by scaling their ranges to [0, 1]. In the case of the hypergeometric distribution (sampling without replacement), they provide an elementary mechanism to attain slightly better bounds than those implied in [17] and by Chvátal [13]. The method also yields good upper bounds for the tail probabilities of the sums of r.v.'s with limited independence.

These constructions also provide pointers to further improvement of the independence bounds. For example, we will take the liberty of redirecting somewhat the estimation method as appropriate when attaining improved tools for analyzing the behavior of the sum of $k$-wise independent r.v.'s. The results simplify and sharpen some of the analyses done in [39] and [40]. In particular, we derive good upper bounds on $E[((\sum_{i=1}^n X_i) - E[\sum_{i=1}^n X_i])^k]$, where $X_1, X_2, \ldots, X_n$ are $k$-wise independent r.v.'s, each of which lies in the interval [0, 1]; this leads to better independence bounds than our $h(n, \mu, \delta)$ when $\delta < 1$. We also prove good bounds on the probability of *exactly* $r$ successes in a sequence of $k$-wise independent Bernoulli trials, which shows that even with modest independence, probabilities and conditional probabilities are close to the fully independent case in situations such as hashing.

The sufficiency of limited independence has several computational applications. First, it means that any random process whose analysis uses the CH bounds can be simulated with a weaker random source than one which outputs unbiased and independent bits. Next, via known constructions of r.v.'s with limited independence using fewer random bits (Joffe [19], Carter and Wegman [9], Mehlhorn and Vishkin [26], Alon, Babai and Itai [1], Siegel [43]), we can reduce the randomness required for certain algorithms. One simple example is that of *random sampling*; given a universe $U$ and a subset $X \subseteq U$, the problem is to estimate the fraction of objects of type $X$ in $U$ such that the absolute error of the output is at most $\delta$ with probability at least $1-\varepsilon$ for given error parameters $\delta$ and $\varepsilon$. The new constructions imply that if $R$ independent samples are required to yield the desired bound, then it in fact suffices for those $R$ samples to be $k^*$-wise independent, for $k^* = O(\log(\frac{1}{\varepsilon}))$. These samples can be generated by $O(\log(\frac{1}{\varepsilon}))$ random samples from $U$ using standard methods. Note that the above construction is not optimal with regard to the number of random bits used (see Bellare, Goldreich, and Goldwasser [6] for an optimal construction), but is extremely simple. It is also easily parallelizable, whereas it is not known how to parallelize other schemes for reducing randomness, e.g., random walks on expanders. It has come to our attention that, via weaker bounds on the $k$th moment, essentially the same bounds for the random sampling problem have been obtained by Bellare and Rompel [7]. We believe that there should be additional applications yielding reduced randomness. A spectrum of explicit constructions of oblivious routing algorithms on the butterfly with varying time-randomness parameters is among the results of Peleg and Upfal [32]; our limited independence result directly matches these bounds on the hypercube and, we believe, should extend to other interconnection networks.

Finally, we combine the method of conditional probabilities [34], [44] with the new construction to obtain two results. We get a much faster implementation of the sequential jobshop scheduling algorithm of Shmoys, Stein, and Wein [41]. It is comparable in time complexity to the speedups of Plotkin, Shmoys, and Tardos [33] and Stein [45], but more importantly, the approximation bound it presents is better than those of [33] and [45]. Here, we show that a problem can be derandomized directly, thereby avoiding the bottleneck step of solving a huge linear program. We also prove an "exact partition" result for set discrepancy and derive a polynomial-time algorithm for it.

The organization of the paper is as follows. Section 2 presents the new formulation and its applications to tail probabilities. Section 3 presents applications of these results to computation.

**2. The basic method and applications to tail probabilities.** In this section, we introduce the method, discuss its implications to the tail probabilities of various distributions, and analyze some related approaches. We also prove probability bounds for exactly $r$ successes in a sequence of Bernoulli trials under limited independence. As discussed in §1, the basic idea used in the CH bounds is as follows. Given $n$ r.v.'s $X_1, X_2, \ldots, X_n$, we want to upper bound the upper tail probability $\Pr(X \geq a)$, where $X \doteq \sum_{i=1}^{n} X_i, \mu \doteq E[X], a = \mu(1+\delta)$, and $\delta > 0$. For any fixed $t > 0$,

$$\Pr(X \geq a) = \Pr(e^{tX} \geq e^{at}) \leq \frac{E[e^{tX}]}{e^{at}};$$

by computing an upper bound $u(t)$ on $E[e^{tX}]$ and minimizing $u(t)/e^{at}$ over $t > 0$, we can upper bound $\Pr(X \geq a)$. When $X_1, X_2, \ldots, X_n$ are binary, we construct a class of functions of $X$ that includes the class $\{e^{tX} : t > 0\}$ and do the minimization over this class; in the process, we discover that $X_1, X_2, \ldots, X_n$ *need only be* $h(n, \mu, \delta)$*-wise independent* for a function $h(\cdot,\cdot,\cdot)$ that will be defined in equation (3) of the next section.

*Notation.* If $x$ is real and $r$ is a positive integer, then $\binom{x}{r}$ will denote, as usual, $(x(x-1)\cdots(x-r+1))/r!$ with $\binom{x}{0} \doteq 1$.

**2.1. Estimating tail probabilities of binary random variables.** The CH bounds are frequently used when the r.v.'s $X_1, X_2, \ldots X_n$ are binary and *independent.* In this section, we first assume that $X_1, X_2, \ldots X_n$ are 0–1 independent r.v.'s with $\Pr(X_i = 1) = p_i, 1 \le i \le n$; the independence assumption will be relaxed later and the results will be extended to r.v.'s $X_i$ with $0 \le X_i \le 1$ in §2.2. Let $X \doteq \sum_{i=1}^n X_i$, and $\mu \doteq E[X] = \sum_{i=1}^n p_i$. We want good upper bounds on $\Pr(X \ge \mu(1+\delta))$ for $\delta > 0$. Chernoff [11] implicitly showed that for identically distributed 0–1 variables $X_1, X_2, \ldots X_n$ and for $a > \mu$,

$$\min_t \frac{E[e^{tX}]}{e^{at}} \le L(n,\mu,a) = \left(\frac{\mu}{a}\right)^a \left(\frac{n-\mu}{n-a}\right)^{n-a}.$$

Hoeffding [17] extended this by showing that $L(n,\mu,a)$ is an upper bound for the above minimum even if the $X_i$'s are not identically distributed and range between 0 and 1. Replacing $a$ with $\mu(1+\delta)$ in the Hoeffding estimate $L(\cdot,\cdot,\cdot)$ gives, for $\delta > 0$,

$$\Pr(X \ge \mu(1+\delta)) \le F(n,\mu,\delta) \doteq \frac{(1 + \frac{\mu\delta}{(n-\mu(1+\delta))})^{n-\mu(1+\delta)}}{(1+\delta)^{\mu(1+\delta)}}.$$

Since $L(n,\mu,a)$ is symmetric with respect to $(a,\mu)$ and $(n-a, n-\mu)$, the Hoeffding estimate also shows that

$$\begin{aligned}\Pr(X \le \mu(1-\delta)) &= \Pr(n - X \ge n - \mu(1-\delta)) \le F(n,\mu,-\delta)\\ &\doteq \frac{(1 - \frac{\mu\delta}{(n-\mu(1-\delta))})^{n-\mu(1-\delta)}}{(1-\delta)^{\mu(1-\delta)}}\end{aligned}$$

The following simple upper bounds for $F(n,\mu,\delta)$ and $F(n,\mu,-\delta)$ are sufficient to derive most of the useful approximations that have appeared in the literature [17], [4], [35], [3].

$$F(n,\mu,\delta) \le G(\mu,\delta) \doteq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$$

(see, for example, [35]);

for $\delta < 1$, $G(\mu,\delta) \le e^{-\delta^2\mu/3}$ [4]; for $\delta > 2e-1$, $G(\mu,\delta) \le 2^{-(1+\delta)\mu}$ [35],

and

$$F(n,\mu,-\delta) \le G(\mu,-\delta) \le e^{-\mu\delta^2/2}$$

[17], [4], [35], [3].

At the heart of these estimates are the simple calculations associated with the multiplicative nature of $E[e^{\Sigma X_i}]$. Recall that $e^{tX} = \sum_{i=0}^\infty (t^i/i!)X^i$. Consider $X^2$, for instance. $X^2 = (X_1 + X_2 + \cdots + X_n)^2 = \sum_{i=1}^n X_i^2 + 2\sum_{1 \le i_1 < i_2 \le n} X_i X_j = \sum_{i=1}^n X_i + 2\sum_{1\le i_1<i_2\le n} X_i X_j$, since $X_i^2 = X_i$ for $X_i \in \{0,1\}$. Similarly, other higher powers of the $X_i$'s are unnecessary, implying that a form simpler and more useful than functions of the form $\{e^{tX} : t > 0\}$ might exist. There are many ways to formalize this. We define for $z = (z_1, z_2, \ldots, z_n) \in \Re^n$ a family of symmetric multilinear polynomials $S_j(z), j =$

$0, 1, \ldots, n$, where $S_0(z) \equiv 1$, and for $1 \leq j \leq n, S_j(z) \doteq \sum_{1 \leq i_1 < i_2 \cdots < i_j \leq n} z_{i_1} z_{i_2} \cdots z_{i_j}$. We start with the following simple lemma.

LEMMA 1. *Suppose $z_1, z_2, \ldots, z_n$ take on binary values. Then for any positive integer $j$, $(z_1 + z_2 + \cdots + z_n)^j \equiv \sum_{i=1}^{\min(j,n)} a_i S_i(z_1, z_2, \ldots, z_n)$ for some* nonnegative *integers $a_1, a_2, \ldots, a_{\min(j,n)}$.*

The proof of Lemma 1 is trivial and is omitted. The converse of Lemma 1 also is true; if $z = (z_1, z_2, \ldots, z_n) \in \{0, 1\}^n$, then for any $j, j = 0, 1, \ldots, n$,

$$\forall (u_0, \ldots, u_j) \in \Re^{j+1} \quad \exists (v_0, \ldots, v_j) \in \Re^{j+1} : \sum_{i=1}^{j} u_i S_i(z) \equiv \sum_{i=1}^{j} v_i (z_1 + z_2 + \cdots + z_n)^i.$$

So, the two forms polynomial of $z_1 + z_2 + \cdots + z_n$ and linear combination of $S_0(z)$, $S_1(z), \ldots, S_n(z)$ are equivalent. Note that if the binary random variables $X_1, X_2, \ldots$, $X_n$ are independent, then $E[S_i(X_1, X_2, \ldots, X_n)]$ is explicitly available; $E[S_i(X_1, X_2, \ldots, X_n)] = S_i(p_1, p_2, \ldots, p_n)$, where $p_j = \Pr(X_j = 1)$. This explains our preference for the $S_i$'s.

Since the expansion $e^{tZ} = \sum_{i=1}^{\infty} (t^i/i!) Z^i$ converges for all $t$ and $Z$ and since all the coefficients $(t^i/i!)$ are positive if $t > 0$, we get the following corollary.

COROLLARY 1. *Let $z_1, z_2, \ldots, z_n$ take on binary values. Then for any $t > 0$, there exist* nonnegative *reals $a_0, a_1, \ldots, a_n$ such that $e^{t(z_1+z_2+\cdots+z_n)} \equiv \sum_{i=0}^{n} a_i S_i(z_1, z_2, \ldots, z_n)$.*

One reason for the use of the function $e^{tX}$ in the CH bounds is the need for higher moments of $X$. In particular, the *moment-generating function* of $X$ is defined to be $E[e^{tX}] = \sum_{i=0}^{\infty} (t^i/i!) E[X^i]$; its derivatives generate all higher moments of $X$. Moreover, the use of moment-generating functions embeds the problem of attaining probability estimates in a space rich with algebraic structure and convex inequalities. (More about the computational aspects of such an alternative approach can be found in [42].) The need for higher moments is due to the fact that a direct application of Markov's or Chebyshev's inequality to upper bound $\Pr(X \geq E[X] \cdot (1 + \delta))$ leads to weak bounds. Higher moments and exponentials give dramatically better estimates. However, when $X$ is the sum of random bits $X_1, X_2 \ldots, X_n$, Lemma 1 and Corollary 1 imply that *all* the higher moments of $X$ can be linearly generated by $\{E[S_i(X_1, X_2, \ldots, X_n)] : i = 0, 1, \ldots, n\}$. Equivalently, they are also generated linearly by *any n* higher moments of $X$.

Thus, we now consider functions of the form $\sum_{i=0}^{n} y_i S_i(X_1, X_2, \ldots, X_n)$, where $y_0, y_1, \ldots, y_n \geq 0$, instead of restricting ourselves to those of the form $e^{tX}$, for some $t > 0$. Indeed, by Corollary 1, we will be considering a class of functions which includes the class $\{e^{tX} : t > 0\}$. For any $y = (y_0, y_1, \ldots, y_n) \in \Re_+^{n+1}$ and $z = (z_1, z_2, \ldots, z_n) \in \Re^n$, define $f_y(z) \doteq \sum_{i=0}^{n} y_i S_i(z)$. With this notation, we can restate Corollary 1 as $\forall t > 0 \ \exists y \in \Re_+^{n+1} : f_y(X_1, X_2, \ldots, X_n) = e^{tX}$. Assume $a \doteq \mu(1 + \delta)$ to be integral. Note that for any nonnegative integer $m, X = m$ iff $f_y(X_1, X_2, \ldots, X_n) = \sum_{i=0}^{m} y_i \binom{m}{i}$ and hence,

$$\Pr(X \geq a) = \Pr\left( f_y(X_1, \ldots, X_n) \geq \sum_{i=0}^{a} y_i \binom{a}{i} \right) \leq \frac{E[f_y(X_1, \ldots, X_n)]}{\sum_{i=0}^{a} y_i \binom{a}{i}} \tag{1}$$
$$= \frac{\sum_{i=0}^{n} y_i S_i(p_1, p_2, \ldots, p_n)}{\sum_{i=0}^{a} y_i \binom{a}{i}}.$$

Thus, our goal now is to minimize this upper bound over $(y_0, y_1, \ldots, y_n) \in \Re_+^{n+1}$. To accomplish this, note that $y_{a+1}, y_{a+2}, \ldots, y_n$ must all be set to 0 since they contribute nonnegative terms to the numerator and nothing to the denominator. Next, note that the right-hand side of inequality (1) is minimized by setting $y_i = 1$ if $i = j^*$ and 0 otherwise, where $j^*$ is the integer at which $S_i(p_1, p_2, \ldots, p_n)/\binom{a}{i}$ is minimized over the range $i = 0, 1, \ldots, a$. To get a better handle on this minimum, we need the following lemma.

LEMMA 2. *For any $i > 0$ and $s \geq 0, S_i(z_1, z_2, \ldots, z_n)$ is maximized by setting $z_1 = z_2 = \cdots = z_n = \frac{s}{n}$, when subject to the constraints that $(z_1, z_2, \ldots, z_n) \in \Re_+^n$ and $\sum_{j=1}^n z_j = s$.*

*Proof.* Suppose $z_p < z_q$ for some vector $z$ satisfying the constraints $(z_1, z_2, \ldots, z_n) \in \Re_+^n$ and $\sum_{j=1}^n z_j = s$. Then, set $z'_p = z_p + \varepsilon$ and $z'_q = z_q - \varepsilon$ for any $\varepsilon < z_q - z_p$, and set $z'_j = z_j$ for all indices $j, j \notin \{p, q\}$. It is easy to verify that $z\prime$ satisfies the above constraints and that $S_i(z') > S_i(z)$. Hence $S_i(z)$ is maximized at $z = z^*$, where $z_j^* = \frac{s}{n}$ for $j = 1, 2, \ldots, n$. □

Inequality (1) and Lemma 2 imply that if $p \doteq (\sum_{i=1}^n p_i)/n = \frac{\mu}{n}$, then for any $y \in \Re_+^{n+1}$,

$$\Pr(X \geq a) \leq \frac{\sum_{i=0}^n y_i \binom{n}{i} p^i}{\sum_{i=0}^a y_i \binom{a}{i}}. \tag{2}$$

Since $\left(\binom{n}{i+1} p^{i+1} / \binom{a}{i+1}\right) / \left(\binom{n}{i} p^i / \binom{a}{i}\right) = \frac{(n-i)p}{a-i}$, which is less than, equal to, or greater than 1 depending on if $i$ is less than, equal to, or greater than $\frac{a-np}{1-\mu/n}$, $\binom{n}{i} p^i / \binom{a}{i}$ is minimized at

$$i^* = h(n, \mu, \delta) \doteq \left\lceil \frac{a - \mu}{1 - \mu/n} \right\rceil = \left\lceil \frac{\mu\delta}{1 - \mu/n} \right\rceil. \tag{3}$$

So, the right-hand side of (2) is minimized at $y = y^*$, where $y_i^* = 1$ if $i = i^*$, and 0 otherwise. Hence, we get

$$\begin{aligned}\Pr(X \geq \mu(1+\delta)) \leq U_1(n, p_1, \ldots, p_n, \delta) &\doteq \frac{S_{j^*}(p_1, p_2, \ldots, p_n)}{\binom{\mu(1+\delta)}{j^*}} \leq U_2(n, \mu, \delta) \\ &\doteq \frac{\binom{n}{i^*}\left(\frac{\mu}{n}\right)^{i^*}}{\binom{\mu(1+\delta)}{i^*}}.\end{aligned}$$

$U_1(n, p_1, \ldots, p_n, \delta)$ is guaranteed to be better than any estimate based on the CH method, since we have considered a larger class of functions. Also, the upper bound $U_2(n, \mu, \delta)$ on $U_1(n, p_1, \ldots, p_n, \delta)$ is better than any such estimate which depends only on $\mu$ and which is oblivious to the actual values of $p_1, p_2, \ldots, p_n$; this includes $F(n, \mu, \delta)$ and $G(\mu, \delta)$.

But most importantly, note that these new bounds will hold even if $X_1, X_2, \ldots, X_n$ are only $h(n, \mu, \delta)$-wise independent. This is because each term in $S_k(X_1, X_2, \ldots, X_n)$ is of the form $X_{i_1}, X_{i_2}, \ldots, X_{i_k}$ for any integer $k$, and, hence, $E[S_k(X_1, X_2, \ldots, X_n)]$ will be the same for $k$-wise independent $X_1, X_2, \ldots, X_n$ as for completely independent $X_1, X_2, \ldots, X_n$. Because $\mu(1+\delta) \leq n, h(n, \mu, \delta) \leq n$; in typical algorithmic situations, $h(n, \mu, \delta) \ll n$. This will be of great use later on.

THEOREM 1. *Let bits $X_1, X_2, \ldots, X_n$ be random with* $\Pr(X_i = 1) = p_i, X = \sum_{i=1}^n X_i$, *and $\mu = E[X] = \sum_{i=1}^n p_i$. Suppose further that the $X_i$'s are $k$-wise independent for $k \geq h(n, \mu, \delta)$. Then for any $\delta > 0$,*

$$\Pr(X \geq \mu(1+\delta)) \leq U_1(n, p_1, \ldots, p_n, \delta) \leq U_2(n, \mu, \delta).$$

*Furthermore,* $U_2(n,\mu,\delta) \le F(n,\mu,\delta) \le G(\mu,\delta)$, *i.e., the CH upper bounds hold even if the* $X_i$*'s are only* $h(n,\mu,\delta)$*-wise independent.*

Our results also imply upper tail bounds for r.v.'s with smaller independence than $h(n,\mu,\delta)$.

LEMMA 3. *Let* $X_1, X_2, \ldots, X_n$ *be binary r.v.'s with* $X \doteq \sum_{i=1}^n X_i$ *and* $\mu \doteq E[X]$. *Then for any* $\delta > 0, \Pr(X \ge \mu(1+\delta)) \le \binom{n}{k}(\mu/n)^k/\binom{\mu(1+\delta)}{k}$, *if the* $X_i$'s *are* $k$*-wise independent for any* $k < h(n,\mu,\delta)$.

*Proof.* Set $y_i = 0$ for $i \ne k$ and $y_k = 1$ in (2). □

It turns out that $U_2(n,\mu,\delta)$ is almost the same as $F(n,\mu,\delta)$.

THEOREM 2. *Given* $n$ *random bits* $X_1, X_2, \ldots, X_n$, *let* $X = \sum_{i=1}^n X_i, \mu = E[X]$, *and* $p \doteq \mu/n$. *Then for any* $\delta > 0$,

(I) *If the* $X_i$*'s are* $\lceil \mu\delta \rceil$*-wise independent, then* $\Pr(X \ge \mu(1+\delta)) \le G(\mu,\delta)$, *where*

$$G(\mu,\delta) = \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \le \begin{cases} e^{-\mu\delta^2/3} & \text{if } \delta < 1; \\ e^{-\mu\delta \ln(1+\delta)/2} \le e^{-\mu\delta/3} & \text{if } \delta \ge 1. \end{cases}$$

(II) *If the* $X_i$*'s are* $\lceil \mu\delta/1-p \rceil$*-wise independent, then* $\Pr(X \ge \mu(1+\delta)) \le F(n,\mu,\delta)$.

(III) *If the* $X_i$*'s are* $\lceil \mu\delta/p \rceil = \lceil n\delta \lceil$*-wise independent, then* $\Pr(X \le \mu(1-\delta)) \le F(n,\mu,-\delta)$, *where*

$$F(n,\mu,\delta) \le \begin{cases} e^{-\mu\delta^2/(2(1-p))} & \text{if } p \le 1/2; \\ e^{-2p\mu\delta^2} & \text{if } p > 1/2. \end{cases}$$

*Proof.* The first claim follows by setting $k = \lceil \mu\delta \rceil \le h(n,\mu,\delta)$ in Lemma 3. The only interesting case is that $k < h(n,\mu,\delta)$. We apply the inequality $\binom{n}{k}(\mu/n)^k/\binom{a}{k} \le Q(n,k,a) \doteq (\frac{n}{a})^k(\mu/n)^k(\frac{n}{n-k})^{n-k}(\frac{a-k}{a})^{a-k}$, valid for any $a < n$; this inequality follows by induction on $k$ combined with the fact that the function $(1-\frac{1}{x})^{x-1}$ is nonincreasing for $x > 1$. Let $a = (1+\delta)\mu$ and $k' = \mu\delta$. Then,

$$\begin{aligned} Q(n,k',a) &= \frac{(n/(n-\mu\delta))^{n-\mu\delta}}{(1+\delta)^{\mu(1+\delta)}} \\ &= \left(\frac{1}{1+\delta}\right)^{\mu(1+\delta)}\left(1+\frac{\mu\delta}{n-\mu\delta}\right)^{n-\mu\delta} \le \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \\ &= G(\mu,\delta). \end{aligned}$$

It is easy to verify that $Q(n,x,a)$ is nonincreasing for $x \le h(n,\mu,\delta)$, and hence the bound $G(\mu,\delta)$ established for $k'$ also holds for $k = \lceil \mu\delta \rceil$. The upper bounds for $G(\mu,\delta)$ are either straightforward or have been established in [4], [35], [3].

The second claim follows immediately from Theorem 1, while the third claim follows by obtaining lower tail bounds from the upper tail of $\sum_{i=1}^n (1-X_i)$ and importing the upper tail bounds established in [17]. By Theorem 1, these bounds hold with independence $h(n,\mu,\delta)$. □

As we will show in §2.3, bounds almost as good as $G(\mu,\delta)$ and $F(n,\mu,-\delta)$ hold with the much smaller independence $k = \lfloor \mu\delta^2 \rfloor$ when $\delta < 1$.

**2.2. Tail probabilities of bounded random variables.** We now show that almost the same results hold for arbitrary r.v.'s which take values in [0, 1]. Analogous bounds for bounded r.v.'s that are constrained to lie in other intervals can be obtained by a linear transformation of their ranges to [0, 1]. Given arbitrary r.v.'s $X_i$ such that $0 \leq X_i \leq 1, i = 1, 2, \ldots, n$, we wish to upper bound $\Pr(X \geq \mu(1+\delta))$, where $X = \sum_{i=1}^n X_i, \mu = E[X]$, and $\delta > 0$. Hoeffding [17] has proved upper tail bounds for bounded r.v.'s, assuming full independence among the $X_i$'s; the main point of interest here again is that partial independence suffices, giving bounds almost as good as Hoeffding's. Almost all of the work is done by the following lemma.

LEMMA 4. *Let $z_i$ be real numbers, with $0 \leq z_i \leq 1, i = 1, 2, \ldots, n$, and suppose that $a \geq 0, j \leq \lfloor a \rfloor$ and $\sum_{i=1}^n z_i \geq a$. Then,*

$$S_j(z_1, z_2, \ldots, z_n) \geq \binom{a}{j}.$$

*Proof.* We will consider only the case $\sum_{i=1}^n z_i = a$; then the upper bound will directly follow if $\sum_{i=1}^n z_i > a$. If $0 < z_p \leq z_q < 1$ for $p \neq q$, then $S_j(z)$ decreases if we set $z_p := z_p - \varepsilon$ and $z_q := z_q + \varepsilon$ for any $\varepsilon < \min(z_p, 1 - z_q)$. Thus, if $S_j(z)$ is minimized at $z^*$ in the domain $[0,1]^n$ under the constraint that $\sum_{i=1}^n z_i = a$, then $0 < z_i^* < 1$ for at most one $i, 1 \leq i \leq n$.

If $a$ is integral, then $z_i^* \in \{0,1\}, i = 1, 2, \ldots, n$, and hence $S_j(z^*) = \binom{a}{j}$. Otherwise, suppose $a$ is nonintegral; let $a_1 = \lfloor a \rfloor$ and $a_2 = a - a_1$. Hence, $z_p^* = a_2$ for some index $p$ and $z_i^* \in \{0,1\}$ for $i \in \{1, 2, \ldots, n\} - \{p\}$; thus,

$$S_j(z^*) = \binom{a_1}{j} + a_2 \cdot \binom{a_1}{j-1},$$

and we need to show that this is at least $\binom{a}{j}$, i.e., that

$$[a_1]_j + a_2 j \cdot [a_1]_{j-1} \geq [a_1 + a_2]_j = [a]_j,$$

where $[x]_r$ denotes $x(x-1)\cdots(x-r+1)$ and $[x]_0 \doteq 1$. This is easily seen by induction on $j$, as follows. Equality clearly holds for $j = 1$. For $j > 1, [a_1]_j + a_2 j \cdot [a_1]_{j-1} = [a_1]_j + (j-1)a_2[a_1]_{j-1} + a_2[a_1]_{j-1}$. Since $a_2 < 1, a_2[a_1]_{j-1} \geq a_2[a_1 + a_2 - 1]_{j-1}$ and, hence,

$$\begin{aligned}
[a_1]_j + a_2 j \cdot [a_1]_{j-1} &\geq a_1([a_1 - 1]_{j-1} + (j-1)a_2[a_1 - 1]_{j-2}) + a_2[a_1 + a_2 - 1]_{j-1} \\
&\overset{\text{ind hyp.}}{\geq} a_1[a_1 + a_2 - 1]_{j-1} + a_2[a_1 + a_2 - 1]_{j-1} \\
&= [a_1 + a_2]_j. \qquad \square
\end{aligned}$$

By essentially the same analysis as before, we get the following theorem.

THEOREM 3. *Given $n$ arbitrary r.v.'s $X_1, X_2, \ldots, X_n$ with $0 \leq X_i \leq 1$ and $E[X_i] = p_i$, let $X \doteq \sum_{i=1}^n X_i$ and $\mu \doteq E[X]$. Then if $X_1, X_2, \ldots, X_n$ are $k$-wise independent for $k \geq h(n, \mu, \delta)$, then $\Pr(X \geq \mu(1+\delta)) \leq U_1(n, p_1, \ldots, p_n, \delta) \leq U_2(n, \mu, \delta)$ for any $\delta > 0$.*

*Proof.* From Lemma 4, we have that for any $a > 0$ and for *nonnegative* $y_0, y_1, \ldots, y_n$,

$$\Pr(X \geq a) \leq \Pr\left(f_y(X_1, \ldots, X_n) \geq \sum_{i=1}^{\lfloor a \rfloor} y_i \binom{a}{i}\right) \leq \frac{E[f_y(X_1, \ldots, X_n)]}{\sum_{i=1}^{\lfloor a \rfloor} y_i \binom{a}{i}},$$

and the rest of the proof follows as before. $\square$

*Remark.* The methods of §2.1 were motivated by the fact that if $X$ is the sum of $n$ 0–1 r.v.'s, then any $n$ higher moments of $X$ linearly generate all the higher moments of $X$. However, note that if r.v's $X_1, X_2, \dots, X_n$ take arbitrary values in the interval [0, 1] and if $X = \sum_{i=1}^n x_i$, then such a result is *not* true, in fact, no bound can be put on the number of higher moments needed to generate all the moments of $X$. However, the intuition gained from §2.1 has helped us obtain a large deviation bound for $X$, which is as good as the known bound [17]. This is despite the fact that we have not considered all the higher moments of $X$; one of the original motivations for Chernoff's consideration of $E[e^{tX}]$ was that it generates all the higher moments of $X$. A possible interpretation of our result of this subsection is that it pinpoints the "crucial" higher moments.

**2.3. Redirecting the method.** Recall that in §2.1 we introduced the class of functions $S_0(z), S_1(z), \dots, S_n(z)$ and generalized Chernoff's idea by working with *nonnegative* linear combinations of these functions. A natural generalization of this is to allow arbitrary linear combinations; however, the corresponding optimization problem, described below, seems hard to analyze.

Suppose we have $n$ *binary* r.v.'s $X_1, X_2, \dots, X_n$ with $\Pr(X_i = 1) = p_i$ and with $X = \sum_{i=1}^n X_i$ and we want good upper bounds on $\Pr(X \geq a)$, where $a > E[X]$, when the $X_i$'s are $k$-wise independent. As before, let

$$f_y(X_1, X_2, \dots, X_n) = \sum_{i=0}^{n} y_i S_i(X_1, X_2, \dots, X_n),$$

with the further restriction that $y_i = 0$ for $i \geq k+1$ to capture the idea of $k$-wise independence. Note that $f_y(X_1, X_2, \dots, X_n)$ is a function of $X$;

$$f_y(X_1, X_2, \dots, X_n) = g_y(X) \doteq \sum_{i=0}^{\min(k,X)} y_i \binom{X}{i}.$$

If $g(t) \geq 0$ for $t = 0, 1, \dots, n$ (so that Markov's inequality can be applied) and if $g_y(b) \geq g_y(a)$ for $b \geq a$, then

$$\Pr(X \geq a) \leq \Pr(g_y(X) \geq g_y(a)) \leq \frac{E[g_y(X)]}{g_y(a)} = \frac{\sum_{i=0}^k y_i S_i(p_1, p_2, \dots, p_n)}{g_y(a)}.$$

We can scale the $y_i$'s so that $g_y(a) = 1$, and thus we get the following linear program with $y_0, y_1, \dots, y_k$ being arbitrary real variables.

LP$(a, k, p_1, p_2, \dots, p_n)$:

Minimize $\sum_{i=0}^k y_i S_i(p_1, p_2, \dots, p_n)$ subject to

- $g_y(j) \geq 0, j = 0, 1, \dots, n,$
- $g_y(a) = 1$, and
- $g_y(b) \geq 1, b = a+1, a+2, \dots, n.$

Unfortunately, we have been unable to analytically compute the optimum of this linear program. However, we now consider an important case in which some of the multipliers are negative and that is a feasible solution to the above LP; our results generalize a result of [22], [8], [27]. We use the $k$th moment inequality

$$\Pr(|X - E[X]| \geq \delta E[X]) \leq \frac{E[|X - E[X]|^k]}{(\delta E[X])^k},$$

which is attributable in various formulations and generalizations to Chebyshev, Markov, and Loève [18] and has been used to attain probability deviation estimates for over a century. Note that if $X = X_1 + X_2 + \cdots X_n$, where the $X_i$'s are random bits, then $(X - E[X])^k$ is a linear combination of $S_0(X_1, X_2, \ldots, X_n), S_1(X_1, X_2, \ldots, X_n), \ldots, S_k(X_1, X_2, \ldots, X_n)$, with some of the multipliers being negative. We derive good upper bounds on $E[|X - E[X]|^k]$, where $X = \sum_{i=1}^n X_i$, with the $X_i$'s being $k$-wise independent r.v.'s that satisfy $|X_i - E[X_i]| \leq 1$; this yields bounds that are better than those given in Theorem 1 and Lemma 3 when $k \ll h(n, \mu, \delta)$ and $\delta < 1$. Moreover, the large-deviation bounds derived in Theorem 5 for $k$-wise independent r.v.'s agree with the simple exponential forms of the large-deviation bounds most often cited for sequences of fully independent Bernoulli trials.

Theorem 4 is similar in spirit and proof to Lemma 4.19 of [22] for identically distributed $X_i$ and constant $k$, but the present result is somewhat tighter even in the case of identically distributed $X_i$, especially if $X = \sum_{i=1}^n X_i$ has small variance. A slightly weaker form of a special case of one of the inequalities proven in Theorem 4 was also obtained in [8], and some related formulas were given in [27]. The proof of Theorem 4, as well as related proofs presented elsewhere, is based upon estimates for the $k$th moment of $X$. Estimates related to ours, but for a more general class of r.v.'s, were established in [28]. That formulation, however, is considerably more complicated than ours and is not as tight for the cases specifically considered here. In particular, Theorem 5 cannot be derived from the bounds in [28] for the $k$th moment. Other related work was done by Gladkov [15] (with later improvements in [16]). He showed that if $Y_1, Y_2, \ldots, Y_n$ are *independent* r.v.'s, with $Y_i$ having the same distribution as $X_i$ and with $Y \doteq Y_1 + Y_2 + \cdots + Y_n$, then as $n \to \infty$, the convergence of $Y$ to the normal distribution implies a comparable convergence for $X$ provided $k$ is sufficiently large.

THEOREM 4. *Let $X_1, \ldots, X_n$ be a sequence of $k$-wise independent r.v.'s that satisfy $|X_i - E[X_i]| \leq 1$. Let $X = \sum_{i=1}^n X_i$, with $E[X] = \mu$, and let $\sigma^2[X]$ denote the variance of $X$ so that $\sigma^2[X] = \sum_{i=1}^n \sigma^2[X_i]$ (provided $k \geq 2$, which we require). Then the following hold for any even $k$.*

(I) *For*[1] $C \geq \sigma^2[X], \Pr(|X - \mu| \geq T) \leq \sqrt{2}\cosh(\sqrt{k^3/36C})(kC/eT^2)^{k/2}$.

(II) *For* $2 \leq k \leq 3(\sigma^2[X])^{1/3}, \Pr(|X - \mu| \geq T) \leq 2(k\sigma^2[X]/eT^2)^{k/2}$.

(III) *For* $C \geq \max\{k, \sigma^2[X]\}, \Pr(|X - \mu| \geq T) \leq (kC/e^{2/3}T^2)^{k/2}$.

*Proof.* We use the $k$th moment inequality

$$\Pr(|X - \mu| \geq T) \leq \frac{E[|X - \mu|^k]}{T^k}. \tag{4}$$

For even $k, E[|X - \mu|^k] = E[(X - \mu)^k]$. Most of our effort will therefore be invested in estimating the $k$th moment of $X - \mu$. Let $p_i = E[X_i]$, for $1 \leq i \leq n$. Then

$$\begin{aligned} E[(X - \mu)^k] &= E\left[\left(\sum_{i=1}^n (X_i - p_i)\right)^k\right] \\ &= \sum_{i_1 + i_2 + \cdots + i_n = k} \binom{k}{i_1, \ldots, i_n} E[(X_1 - p_1)^{i_1}] \cdots E[(X_n - p_n)^{i_n}]. \end{aligned} \tag{5}$$

[1] Recall that $\cosh(x) = (e^x + e^{-x})/2$. Throughout this manuscript, $e$ denotes, as usual, the base of the natural logarithm.

Clearly, $E[X_i - p_i] = 0$ and any term in (5) that has some $i_j = 1$ must be 0. More generally, $|E[X_i - p_i]^\ell| \le \sigma^2[X_i]$ for any $\ell \ge 2$, since $|X_i - p_i| \le 1$ and therefore $|X_i - p_i|^\ell \le |X_i - p_i|^2$; hence $|E[X_i - p_i]^\ell| \le |E[X_i - p_i]^2| = \sigma^2[X_i]$. Thus,

$$\begin{aligned}
E[(X-\mu)^k] &= \sum_{\ell=1}^{k/2-1} \sum_{\substack{j_1+j_2+\cdots+j_{k/2-\ell}=k \\ j_i \ge 2}} \binom{k}{j_1, j_2, \ldots, j_{k/2-\ell}} \\
&\quad \times \sum_{i_1<i_2<\cdots<i_{k/2-\ell}} \prod_{r=1}^{k/2-\ell} E[X_{i_r} - p_{i_r}]^{j_r} \\
&\le \sum_{\ell=1}^{k/2-1} \sum_{\substack{j_1+j_2+\cdots+j_{k/2-\ell}=k \\ j_i \ge 2}} \binom{k}{j_1, j_2, \ldots, j_{k/2-\ell}} \\
&\quad \times \sum_{i_1<i_2<\cdots<i_{k/2-\ell}} \prod_{r=1}^{k/2-\ell} \sigma^2[X_{i_r}] \\
&\le \sum_{\ell=0}^{k/2-1} \sum_{\substack{j_1+j_2+\cdots+j_{k/2-\ell}=k \\ j_i \ge 2}} \binom{k}{j_1, j_2, \ldots, j_{k/2-\ell}} \frac{(\sigma^2[X])^{k/2-\ell}}{(k/2-\ell)!}.
\end{aligned} \tag{6}$$

Estimate (6) comes about because the summation

$$\sum_{i_1<i_2<\cdots<i_{k/2-\ell}} \prod_{r=1}^{k/2-\ell} \sigma^2[X_{i_r}]$$

is maximized when all the $\sigma^2[X_{i_c}]$ are equal (Lemma 2) and hence is at most

$$\binom{n}{k/2-\ell}\left(\frac{\sigma^2[X]}{n}\right)^{k/2-\ell} \le \frac{(\sigma^2[X])^{k/2-\ell}}{(k/2-\ell)!}.$$

Let $T_0, T_1, \ldots T_{k/2-1}$ denote the $k/2$ terms in summation (6); hence,

$$T_\ell = \sum_{\substack{j_1+j_2+\cdots+j_{k/2-\ell}=k \\ j_i \ge 2}} \binom{k}{j_1, j_2, \ldots, j_{k/2-\ell}} \frac{(\sigma^2[X])^{k/2-\ell}}{(k/2-\ell)!} \tag{7}$$

and

$$T_0 = \binom{k}{2, 2, \ldots, 2} \frac{(\sigma^2[X])^{k/2}}{(k/2)!}. \tag{8}$$

There are exactly $\binom{k/2+\ell-1}{2\ell}$ terms (i.e., possible sets of assignments for $j_1, j_2, \ldots, j_{k/2-\ell}$) in $T_\ell$, since $\sum j_i = k$ and $j_i \ge 2$. For each such assignment of values, $\sum_{k=1}^{k/2-\ell}(j_k - 2) = 2\ell$, hence $\prod_{k=1}^{k/2-\ell} j_k! \ge 2^{k/2-\ell}3^{2\ell}$ (and equality holds only when $\ell \le k/6$ and exactly $2\ell$ of the $k/2-\ell$ values equal 3 while the remaining values equal 2). Hence $\binom{k}{j_1, j_2, \ldots, j_{k/2-\ell}} \le (2/9)^\ell \binom{k}{2,2,\ldots,2}$ and

$$T_\ell \le \binom{k/2+\ell-1}{2\ell}\left(\frac{2}{9\sigma^2[X]}\right)^\ell \frac{(k/2)!}{(k/2-\ell)!} T_0.$$

Since $\binom{k/2+\ell-1}{2\ell}\frac{(k/2)!}{(k/2-\ell)!} < (k/2)^{3\ell}/(2\ell)!$, we have $E[(X-\mu)^k] \le T_0 \sum_{\ell=0}^{k/2-1} \frac{k^{3\ell}}{(36\sigma^2[X])^\ell (2\ell)!}$. Using Taylor's series for $\cosh(x) = (e^x + e^{-x})/2 = \sum_j x^{2j}/(2j)!$, we see that

$$\sum_{\ell=0}^{k/2-1} \frac{(k^3/36\sigma^2[X])^{2\ell/2}}{(2\ell)!} \le \cosh\left(\sqrt{\frac{k^3}{36\sigma^2[X]}}\right).$$

Consequently

$$E[(X-\mu)^k] \le \cosh\left(\sqrt{\frac{k^3}{36\sigma^2[X]}}\right) T_0. \tag{9}$$

$T_0$ is readily bounded by expanding (8) to get $T_0 = k!(\sigma^2[X])^{k/2}/(2^{k/2}(k/2)!)$. We may apply a strong version of Stirling's formula [37]

$$(r/e)^r \sqrt{2\pi r} e^{1/(12r+1)} \le r! \le (r/e)^r \sqrt{2\pi r} e^{1/(12r)},$$

which is valid for all $r \ge 1$, to bound both $k!$ and $(k/2)!$. This yields $T_0 \le \sqrt{2}(k\sigma^2[X]/e)^{k/2}$. Substituting for $T_0$ in (9) gives

$$E[(X-\mu)^k] \le \sqrt{2}\cosh\left(\sqrt{\frac{k^3}{36\sigma^2[X]}}\right)\left(\frac{k\sigma^2[X]}{e}\right)^{k/2}, \tag{10}$$

which establishes the desired bound for $E[(X-\mu)^k]$.

Estimate (6) is an increasing function of $\sigma^2[X]$, and the estimate in (10) exceeds (6). Therefore, $\sigma^2[X]$ can be replaced by any $C \ge \sigma^2[X]$ in (10). The proof of (I) is completed by applying this estimate to (4).

All other bounds are special cases of (I). When $k \le 3(\sigma^2[x])^{1/3}$, we use $C = \sigma^2[X]$ in (I) and overestimate $\cosh(\sqrt{k^3/36\sigma^2[X]})$ by $\cosh(\sqrt{3/4}) < \sqrt{2}$.

(III) is easily verified for $k = 2$ by applying Chebyshev's inequality,

$$\Pr(|X-\mu| \ge T) \le \frac{\sigma^2[X]}{T^2} \le \frac{2\sigma^2[X]}{e^{2/3}T^2}.$$

For $k \ge 4$ we may replace $C$ by $\max\{\sigma^2[X], k\}$ in (I) and overestimate $\cosh(\sqrt{k^3/36C})$ by $\cosh(\frac{k}{6})$. Since $\cosh(x) < e^x/\sqrt{2}$ for $x \ge \frac{1}{2}$, we get $\cosh(\frac{k}{6}) \le e^{k/6}/\sqrt{2}$ and, hence,

$$\Pr(|X-\mu| \ge T) \le \left(\frac{kC}{e^{2/3}T^2}\right)^{k/2}.$$

This concludes the proof of estimate (III). □

We now combine the results of Theorem 4 and Theorem 2 to establish Chernoff-like bounds [11], [17], where the independence $k$ might even be much smaller than the deviation we wish to bound.

THEOREM 5. *If $X$ is the sum of $k$-wise independent r.v.'s, each of which is confined to the interval* $[0,1]$ *with $\mu = E[X]$, then the following hold:*

(I) *For $\delta \le 1$,*

(a) *if $k \le \lfloor \delta^2\mu e^{-1/3} \rfloor$, then* $\Pr(|X-\mu| \ge \delta\mu) \le e^{-\lfloor k/2 \rfloor}$, *and*

(b) *if $k = \lfloor \delta^2\mu e^{-1/3} \rfloor$, then* $\Pr(|X-\mu| \ge \delta\mu) \le e^{-\lfloor \delta^2\mu/3 \rfloor}$.

(II) *For $\delta \ge 1$,*

(a) *if* $k \le \lfloor \delta\mu e^{-1/3} \rfloor$, *then* $\Pr(|X-\mu| \ge \delta\mu) \le e^{-\lfloor k/2 \rfloor}$, *and*
(b) *if* $k = \lfloor \delta\mu e^{-1/3} \rfloor$, *then* $\Pr(|X-\mu| \ge \delta\mu) \le e^{-\lfloor \delta\mu/3 \rfloor}$.
(III) *For* $\delta \ge 1$ *and* $k = \lceil \delta\mu \rceil$,

$$\Pr(|X-\mu| \ge \delta\mu) \le G(\mu,\delta) \le e^{-\delta \ln(1+\delta)\mu/2} < e^{-\delta\mu/3}.$$

*Proof.* (I). To prove that (1a) holds we apply Theorem 4(III) with $C=\mu, T=\delta\mu$ and $k=\lfloor \delta^2\mu/e^{1/3} \rfloor$, which is permissible since $\mu \ge k$ and $\mu \ge \sigma^2[X]$ for variables in the range [0, 1]. When $k=\lfloor \delta^2\mu/e^{1/3} \rfloor$ and $k$ is even, this gives a bound of

$$\Pr(|X-\mu| \ge \delta\mu) \le \left(\frac{k}{e^{2/3}\delta^2\mu}\right)^{k/2} \le e^{-k/2} \le e^{-\lfloor \delta^2\mu/3 \rfloor}, \quad \text{since } 2e^{1/3} < 3.$$

If $k$ is odd, we follow a calculation similar to that above, but only use independence $k-1$. This gives

$$\Pr(|X-\mu| \ge \delta\mu) \le \left(\frac{k-1}{e^{2/3}\delta^2\mu)}\right)^{k-1/2} \le \left(\frac{k-1}{ek}\right)^{(k-1)/2} \le e^{-(k-1)/2}e^{-(k-1)/2k}.$$

Since $k \ge 2$ for the desired bound to give anything less than 1, $e^{-(k-1)/2k} \le e^{-1/3}$ and hence,

$$\Pr(|X-\mu| \ge \delta\mu) \le e^{-(k-1)/2+1/3)} \le e^{-\lfloor \delta^2\mu/3 \rfloor}.$$

In (II) we follow the same iterations as in (I), but set $C=\delta\mu$ and $k \le \lfloor \delta\mu/e^{1/3} \rfloor$ for (IIa); (IIb) we use $k=\lfloor \delta\mu/e^{1/3} \rfloor$ or $\lfloor \delta\mu/e^{1/3} \rfloor - 1$, depending on the parity of $\lfloor \delta\mu/e^{1/3} \rfloor$. In (III) we reiterate the result of Theorem 2, combined with Theorem 3. $\square$

*Remark.* The proofs of parts (I) and (III) of Theorem 5 also point out the relative merits of the basic method (§§2.1 and 2.2) versus its redirection of this subsection. The basic method of using nonnegative linear combinations of the symmetric polynomials $S_i$ gives better probability bounds when $\delta$, the relative deviation from the mean, is greater than 1; it yields the probability bound of $\exp(-\Theta(\delta\ln(1+\delta)\mu))$ in this case. On the other hand, the formulation involving the $k$th moment inequality gives a much smaller bound on the amount of independence needed when $\delta < 1$.

**2.4. Probability bounds for exactly $r$ successes under limited independence.** Some applications require estimates for the probability that *exactly* $r$ successes occur in cases where the occurrence of at least $r$ successes is not too improbable. The following theorem shows how and when this can be done. It also provides relative errors, which can be useful for estimating conditional probabilities.

THEOREM 6. *Let* $X_1, X_2, \ldots, X_n$ *and* $Y_1, Y_2, \ldots, Y_n$ *be Bernoulli trials with probabilities of success* $E[X_i] = E[Y_i] = p_i$. *We let the* $Y_i$*'s be independent, but only require the* $X_i$*'s to be k-wise independent. Let* $p(r) = \Pr(\sum_i Y_i = r)$ *and* $p_k(r) = \Pr_k(X = r)$, *where the subscript k denotes the k-wise independent trials. Let* $P(r) = \sum_{\ell \ge r} p(\ell)$, *and* $P_k(r) = \sum_{\ell \ge r} p_k(\ell)$.

(I) *If* $r \le k$, *then* $|p_k(r) - p(r)| \le \binom{n}{k}\binom{k}{r}\left(\frac{\mu}{n}\right)^k$.
(II) *If* $k \ge e\mu + \ln(\frac{1}{p(0)}) + r + D$, *then* $|p_k(r) - p(r)| \le e^{-D}p(r)$.
(III) *If* $k \ge e\mu + \ln(\frac{1}{p(0)}) + r + D$, *then* $|P_k(r) - P(r)| \le (1-P(r))e^{-D}$.
(IV) *If* $r \ge (1+\delta)\mu + k$, *then* $P_k(r) \le (1+\delta)^{-k}, P(r) \le (1+\delta)^{-k}$, *and, hence* $|P_k(r) - P(r)| \le (1+\delta)^{-k}$. *Although* (IV) *holds for all values of k, it is meant to be used for* $k \ll \lceil \delta\mu \rceil$.

(V) *If* $r \geq (1+\delta)\mu + k$ *and* $k \geq \lceil \delta\mu \rceil$, *then* $P_k(r) \leq G(\mu,\delta), P(r) \leq G(\mu,\delta)$, *and, hence,* $|P_k(r) - P(r)| \leq G(\mu,\delta)$.

*Proof.* For an arbitrary event $A$, we may use standard inclusion–exclusion to estimate the probability of the event $[A \wedge [\wedge_{\ell \notin \{i_1,\ldots,i_r\}}(X_\ell = 0)]]$. The probability $p(r)$ can be expressed in terms of events $A = [\wedge_{j\in\{i_1,\ldots,i_r\}}(X_j = 1)]$, which admits a simple estimation as follows.

$$p_k(r) = \sum_{i_1<i_2<\cdots<i_r} \Pr_k\left(\left(\bigwedge_{j\in\{i_1,\ldots,i_r\}} (X_j = 1)\right) \wedge \left(\bigwedge_{\ell\notin\{i_1,\ldots,i_r\}} (X_\ell = 0)\right)\right)$$
$$= \sum_{i_1<i_2<\cdots<i_r} \sum_{l=0}^{n-r} \sum_{\substack{i_{r+1}<\cdots<i_{r+l}\\ i_{r+1},\ldots,i_{r+l}\notin\{i_1,\ldots,i_r\}}} (-1)^l \Pr_k\left(\bigwedge_{j\in\{i_1,\ldots,i_{r+1}\}} (X_j = 1)\right)$$
$$= \sum_{l=0}^{n-r} \sum_{i_1<\cdots<i_{r+l}} (-1)^l \binom{r+l}{r} \Pr_k\left(\bigwedge_{j\in\{i_1,\ldots,i_{r+l}\}} (X_j = 1)\right).$$

Truncating the outer summation at $l = k - r$ introduces an error that is bounded by the last term of the truncated sum. Let $p_k^T(r)$ and $p^T(r)$ denote these truncated sums, in the respective cases of $k$-wise and full independence. Since the first $k - r$ terms in the outer summation are the same for both fully and $k$-wise independent r.v.'s, $p_k^T(r) = p^T(r)$. Furthermore, $\Pr_k(\wedge_{j\in\{i_1,\ldots,i_k\}}(X_j = 1)) = \prod_{j=1}^k p_{i_j}$. Hence,

$$p_k(r) = p^T(r) - (-1)^{k-r}\delta_k \binom{k}{r} \sum_{i_1<\cdots<i_k} \prod_{j=1}^k p_{i_j}$$

for some $\delta_k \in [0,1]$, and an identical inequality holds without the $k$ subscripts. Hence,

$$|p_k(r) - p(r)| \leq \binom{k}{r} \sum_{i_1<\cdots<i_k} \prod_{j=1}^k p_{i_j}. \tag{11}$$

$\sum_{i_1<\cdots<i_k}\prod_{j=1}^k p_{i_j}$ is maximized when all $p_{i_j}$ are equal (Lemma 2) and, hence,

$$|p_k(r) - p(r)| \leq \binom{k}{r}\binom{n}{k}\frac{(p_1 + p_2 + \cdots + p_n)^k}{n^k} = \binom{n}{k}\binom{k}{r}(\mu/n)^k;$$

(I) follows.

To get multiplicative error bounds, let $Q_r = \sum_{i_1<i_2<\cdots<i_r}\prod_{\ell=1}^r (p_{i_\ell}/(1-p_{i_\ell}))$ and define the summation in the error estimate of equation (11) by $R_k = \sum_{i_1<\cdots<i_k}\prod_{j=1}^k p_{i_j}$. Observe that $R_k$ is the expected number of size $k$ sets of successes among $n$ trials so that $z$ successes total accounts for $\binom{z}{k}$ such sets. In the fully independent case, $p(r) = p(0)\sum_{i_1<i_2<\cdots<i_r}\prod_{\ell=1}^r(p_{i_\ell}/(1-p_{i_\ell})) = p(0)Q_r$. Furthermore, $R_r \leq Q_r$, and $\binom{k}{r}R_k \leq R_r \times R_{k-r}$. It follows that

$$|p_k(r) - p(r)| \leq \binom{k}{r} R_k \leq R_r \times R_{k-r} \leq Q_r R_{k-r} \leq \frac{p(r)\mu^{k-r}}{p(0)(k-r)!},$$

since $R_{k-r} \leq \binom{n}{k-r}(\frac{\mu}{n})^{k-r} \leq \frac{\mu^{k-r}}{(k-r)!}$.

For $k \geq e\mu - \log(p(0)) + r + D$, the factor $(\mu)^{k-r}/(p(0)(k-r)!)$ is bounded by $((k+\log(p(0)) - D - r)/(k-r))^{k-r}e^{-\log(p(0))} \leq e^{-D}$, which establishes (II).

(III) is immediate, since

$$P_k(r) = 1 - \sum_{\ell<r} p_k(\ell)$$

and, hence,

$$|P_k(r) - P(r)| \leq \sum_{\ell<r} |p_k(\ell) - p(\ell)| \leq \sum_{\ell<r} p(\ell)e^{-D} = (1 - P(r))e^{-D}.$$

Finally, suppose that $r = (1+\delta)\mu + k$. Then by Lemma 3,

$$\begin{aligned} P_k(r) &\leq \frac{\mu^k}{k!\binom{r}{k}} \leq \frac{\mu^k}{r(r-1)(r-2)\cdots(r-k+1)} \\ &\leq (1+\delta)^{-k}. \end{aligned} \tag{12}$$

(IV) is completed by observing that (12) also holds with $P(r)$ substituted for $P_k(r)$. (V) is an immediate consequence of Theorem 2. This concludes the proof of Theorem 6. □

It is worth pointing out that parts (I)–(III) of Theorem 6 are not strong when $\mu > \sqrt{n}$, since it follows from the work of Linial and Nisan [24] that $\Pr(X = \ell) = \Pr(Y = \ell)(1 + O(e^{-2(k-\ell)/\sqrt{n}}))$ independently of $\mu$, which gives a much sharper bound in this case. Also, independently of our work, a result similar to part (I) of Theorem 6 has been proven by Even et al. [14]; they have shown that $|p_k(0) - p(0)| \leq 2^{-\Omega(k)}$.

Theorem 6, in fact, achieves its greatest strength when $\mu$ is small, say $\mu = o(n)$, or even $\mu = O(1)$. Such instances are not unusual when pseudorandom integers are being generated uniformly in the range $[0, n]$, and a successful trial corresponds to just a few different values. This is precisely the usual circumstance in, for instance, hashing [40], [52]. As an example, consider the (uniformly distributed) random placement of $n$ balls among $n$ slots. The expected number of items in slot 1 is only 1. The probability $p(0)$ that no item lands in a given slot is about $\frac{1}{e}$. Theorem 6 shows that if the independence $k$ is $e + 1 + r + D$, the probability that exactly $r$ items land in that slot is the same in the $k$-wise independent case as in the fully independent case up to a multiplicative factor of $(1 + e^{-D})$ or less. Suppose that, during the placements of balls $l$ through $m$, exactly $r$ balls fall into slot $j$ for $1 \leq l \leq m < n, r \leq m - l + 1$. Let $\chi^r_{[l,m]}$ denote this event (with the dependence upon $j$ understood). The conditional probability that, under $k$-wise independence, ball $m+1$ also falls into slot $j$ is $\Pr_k(\chi^1_{[m+1,m+1]}|\chi^r_{[l,m]})$. If we use 1 degree of freedom for the $(m+1)$st ball, it will be uniformly distributed, while the previous $m$ balls will enjoy $(k-1)$wise independence. We may estimate the conditional probability as $\Pr_k(\chi^r_{[l,m]}|\chi^1_{[m+1,m+1]}) \times \Pr(\chi^1_{[m+1,m+1]})/\Pr_{k-1}(\chi^r_{[l,m]})$. Since both $\Pr_{k-1}(\chi^r_{[l,m]})$ and $\Pr_k(\chi^r_{[l,m]}|\chi^1_{[m+1,m+1]})$ can be estimated by $\Pr(\chi^r_{[l,m]})$ $(1 \pm \mathrm{err}_{k-1})$ where $\mathrm{err}_{k-1}$ is the relative error that results from the limited independence, we see that $\Pr_k(\chi^1_{[m+1,m+1]}|\chi^r_{[l,m]})$ is very close to $\frac{1}{n}$, with a relative error that is approximately $2\mathrm{err}_{k-1}$. For $k \geq 1 + e + 1 + r + D$, the resulting accuracy is about $1 \pm 2e^{-D}$. Thus even with modest independence, this process behaves "as expected" much of the time; that is, the corresponding conditional probabilities for $k$-wise independence are very close to the ones for full independence, in many cases.

**2.5. How close to optimal are our results?** It is known that the standard CH bounds are optimal in general to within a constant factor in the exponent, since we know by the Central Limit Theorem that as $n \to \infty$, the tail of the scaled sum of i.i.d. r.v.'s tends to the tail of a normal distribution, and hence we cannot significantly improve the tail probabilities presented by Theorem 5. However, what about the independence we obtain? Can it be reduced further to get the same tail probabilities?

To answer this, we note that the tail probabilities presented by Theorem 5 for $k$-wise independent r.v.'s are of the form $e^{-c \cdot k}$. However, $n$ $k$-wise independent r.v.'s can be obtained from a sample space of size

$$\sum_{i=0}^{\lfloor k/2 \rfloor} \binom{n}{i} \approx (O(n/k))^{\lfloor k/2 \rfloor}$$

as shown for binary unbiased r.v.'s by Chor et al. [12] and for general r.v.'s by Alon, Babai, and Itai [1]. Noting next that any nonzero probability in a sample space of size $t$ is at least $\frac{1}{t}$, we see that to obtain a tail probability of the form $e^{-c \cdot k}$, we need at least $\Omega(\frac{k}{\log(n/k)})$-wise independence. Thus, the independence we obtain cannot generally be reduced by more than a factor of $O(\log n)$.

However, by using results from the newly developing theory of *approximating probability distributions* (Naor and Naor [29], Azar, Motwani, and Naor [5], Alon et al. [2], Even et al. [14], and Chari, Rohatgi, and Srinivasan [10]), we obtain optimal results in the case where the $X_i$'s are binary with $\Pr(X_i = 1) = \frac{1}{2}$. A sample space $X$ for $n$-bit vectors was defined to be $k$-wise $\varepsilon$-biased by Naor and Naor [29] (see also Vazirani [49]) if

$$\forall S \subseteq \{1, 2, \ldots, n\}, 1 \leq |S| \leq k, \left| \Pr\left( \bigoplus_{i \in S} x_i = 1 \right) - \Pr\left( \bigoplus_{i \in S} x_i = 0 \right) \right| \leq \varepsilon,$$

where $\oplus$ denotes the XOR function and $x_i$ denotes the $i$th bit of an $n$-bit string $x$ picked uniformly at random from $X$. One property of such a sample space is that $\forall \ell, \ell = 1, 2, \ldots, k, \forall \{i_1, i_2, \ldots, i_\ell\} \subseteq \{1, 2, \ldots, n\}, \forall b_1 b_2 \cdots b_\ell \in \{0,1\}^\ell$,

$$\left| \Pr(x_{i_1} = b_1, x_{i_2} = b_2, \ldots, x_{i_\ell} = b_\ell) - \frac{1}{2^\ell} \right| \leq \varepsilon. \tag{13}$$

$X$ is $\varepsilon$-biased if it is $n$-wise $\varepsilon$-biased. Constructions of $k$-wise $\varepsilon$-biased sources of size poly$(k, \log n, \frac{1}{\varepsilon})$ were presented in [29], [2]. Such sample spaces have been shown to have several applications to explicit constructions and derandomization, mainly because probabilistic analyses may be expected to be robust under small perturbations of the probabilities. It is easy to see how our methods can be used to derive large deviation bounds for $x_1 + x_2 + \cdots + x_n$; from inequality (13), it follows that for a $k$-wise $\varepsilon$-biased source $X$,

$$\forall \ell \leq k \; E[S_\ell(x_1, x_2, \ldots, x_n)] \leq \binom{n}{\ell} \cdot \left( \frac{1}{2^\ell} + \varepsilon \right),$$

and hence, by picking $\varepsilon \leq 1/2^\ell$, this quantity can at most be double its unbiased value of $\binom{n}{\ell} \cdot 1/2^\ell$. Thus, for a $k$-wise $\varepsilon$-biased random source with $k = h(n, \frac{n}{2}, \delta) = n\delta$ and $\varepsilon = 2^{-k}$,

$$\Pr\left( \sum_{i=1}^{n} x_i \geq \frac{n}{2}(1 + \delta) \right) \leq 2 \cdot U_2\left(n, \frac{n}{2}, \delta\right). \tag{14}$$

Because such a source can be generated using poly$(k, \log n)$ random bits, we see that this result is optimal as long as $k = \Omega(\log n)$; if $k = O(\log n)$, then the probability space is polylogarithmic in size and should in most cases be dispensable via brute-force search of the sample space. Similar results hold when the $X_i$'s are binary with their probabilities of being 1 being the same negative power of 2 (not necessarily 1/2) using identical methods.

**2.6. Upper tail bounds for some other distributions.** Suppose we have random bits $X_1, X_2, \ldots, X_n$ with some arbitrary distribution. Let $X = \sum_{i=1}^n X_i$ and $\mu = E[X]$. Then, for any $a > \mu$, the methods of §2.1 yield

$$\Pr(X \geq a) \leq \frac{\sum_{i=0}^a y_i E[S_i(X_1, X_2, \ldots, X_n)]}{\sum_{i=0}^a y_i \binom{a}{i}}, \quad \forall (y_0, y_1, \cdots y_1) \in \Re_+^{a+1}.$$

The following theorem is immediate.

THEOREM 7. *Given $n$ random bits $X_1, X_2, \ldots, X_n$ with $X = \sum_{i=1}^n X_i$ and $\mu = E[X]$, suppose $z_j$ is an upper bound on $E[S_j(X_1, X_2, \ldots, X_n)], j = 1, 2, \ldots, n$. Then, if $a = \mu(1+\delta)$ for $\delta > 0$, the following hold:*

(I) $\Pr(X \geq a) \leq \sum_{i=0}^a y_i z_i / \sum_{i=0}^a y_i \binom{a}{i}, \forall (y_0, y_1, \ldots, y_a) \in \Re_+^{a+1}$.

(II) *If $X_1, X_2, \ldots, X_n$ are $k$-wise independent, then* $\Pr(X \geq a) \leq \min_{i=1,2,\ldots,k} z_i / \binom{a}{i}$.

As an example of a distribution that benefits from the above, consider the *self-weakening r.v.'s* defined and used in [31]; random bits $X_1, X_2, \ldots, X_n$ are defined to be self-weakening with parameter $\lambda$ in [31] if for all $j$ and for all distinct indices $X_{i_1}, X_{i_2}, \ldots, X_{i_j}, E[\prod_{\ell=1}^j X_{i_\ell}] \leq \lambda \prod_{\ell=1}^j E[X_{i_\ell}]$; note that $z_j \leq \lambda \binom{n}{j} (\frac{\mu}{n})^j$ in this case. Hence, Theorem 7 directly implies one of the main theorems of [31], which states that if $X_1, X_2, \ldots, X_n$ are self-weakening random bits with parameter $\lambda$ with $X = \sum_{i=1}^n X_i$ and $\mu = E[X]$, then for any $\delta > 0, \Pr(X \geq \mu(1+\delta))$ is at most $\lambda$ times any CH-type upper bound on the corresponding probability had the $X_i$ been independent, with the same individual distributions. It was the work of [31] which primarily motivated the methods of §2.1. Furthermore, the applications sketched in §3.2 use Theorem 7.

Theorem 7 helps improve the known upper tail probability bounds for the *hypergeometric distribution*, an important source of self-weakening random variables. Suppose $n$ balls are picked at random *without replacement* from an urn containing $M$ red balls and $N - M$ balls of other colors. Let $X$ be the number of red balls picked in the random sample and let $p \doteq \frac{M}{N}$. Then for $\delta > 0$, a special case of a result of Hoeffding [17] (see Chvátal [13] for another proof) implies that

$$\Pr(X \geq np(1+\delta)) \leq F(n, np, \delta). \tag{15}$$

We prove the following strengthened version of inequality (15).

LEMMA 5. *Suppose a random set of $n$ balls is picked from an urn containing $M$ red balls and $N - M$ balls of other colors. If $X$ denotes the number of red balls picked, $p = \frac{M}{N}, \delta > 0$, and $k \doteq h(n, np, \delta) = o(N)$, then*

$$\Pr(X \geq np(1+\delta)) \leq U_2(n, np, \delta) e^{-\Theta(k^2/M)} \leq F(n, np, \delta) e^{-\Theta(k^2/M)}.$$

*Proof.* (Sketch). Number the balls picked as $1, 2, \ldots, n$ and let $X_i$ be the indicator r.v. for the event that ball $i$ was red. Then $X = \sum_{i=1}^n X_i$ and

$$\Pr(X \geq a) \leq U_3(n, np, \delta) \doteq \frac{E[S_k(X_1, X_2, \ldots, X_n)]}{\binom{a}{k}},$$

where $a \doteq \lceil np(1+\delta)\rceil$. For distinct indices $i_1, i_2, \ldots, i_k$, $E[\prod_{j=1}^k X_{i_j}] = \Pr(\wedge_{j=1}^k X_{i_j} = 1) = \prod_{i=0}^{k-1} \frac{M-i}{N-i}$; hence,

$$\frac{U_3(n, np, \delta)}{U_2(n, np, \delta)} = \left(\frac{N}{M}\right)^k \cdot \prod_{i=0}^{k-1}\left(\frac{M-i}{N-i}\right)$$
$$= \prod_{i=1}^{k-1}\left(1 - \left(\frac{N}{M} - 1\right)\frac{i}{N-i}\right) \leq e^{-\left(\frac{N}{M}-1\right)\sum_{i=1}^{k-1}\frac{i}{N-i}},$$

which is $e^{-\Theta(k^2/M)}$ if $k = o(N)$. □

*Remark.* Note that sampling without replacement produces r.v.'s which are self-weakening with parameter 1. Lemma 5 gives good improvements over inequality (15) in many interesting cases, e.g., consider the case $p$ = constant, $\delta$ = constant, and $\Omega(M^{0.5+\varepsilon}) \leq n = o(N)$ for any fixed $\varepsilon > 0$.

Also, the CH bounds [11], [17], [35], [3] depend only on $\mu$, not on the actual values of $p_i$, and give the upper bound $F(n, \mu, \delta) \geq U_2(n, \mu, \delta)$. We know for any $\delta > 0$ that $\Pr(X \geq \mu(1+\delta)) \leq U_1(n, p_1, \ldots, p_n, \delta) = S_k(p_1, \ldots p_n)/\binom{\mu(1+\delta)}{k}$, where $k = h(n, \mu, \delta)$. By Lemma 2, this is maximized by setting $p_i = \mu/n$ for all $i$, subject to the constraint that $E[X] = \mu$. However, if the values $p_i$ are different, we get bounds formally superior to $U_2(n, \mu, \delta)$ and $F(n, \mu, \delta)$. Suppose, for example, that $\mu = n/2, p_i = \varepsilon, i = 1, 2, \ldots, n/2$, and $p_i = 1 - \varepsilon, i = \frac{n}{2} + 1, \ldots, n$, where $0 < \varepsilon \leq \frac{1}{2}$. Then

$$\frac{U_1(n, p_1, p_2, \ldots, p_n, \delta)}{U_2(n, \mu, \delta)} \leq f(\varepsilon) \doteq 2^k\left(\sum_{i=0}^{k}\binom{n/2}{i}\binom{n/2}{k-i}\varepsilon^i(1-\varepsilon)^{k-i}\right)\Big/\binom{n}{k},$$

where $k = h(n, \mu, \delta)$. Note that $f(0) \leq e^{-\Theta(k^2/n)}$ by Lemma 5 and $f(\varepsilon)$ can get arbitrarily close to $f(0)$ since $f(\cdot)$ is continuous.

*Remark.* This particular result can only increase the constant factor in the exponent of $U_2(n, \mu, \delta)$. However, it is a small step towards better understanding of the dependence of $\Pr(X \geq \mu(1+\delta))$ on $n, p_1, \ldots, p_n$, and $\delta$. Similar improvements can also be made in the case of nonbinary r.v.'s. An alternative approach might be to derive CH bounds for a sum of Bernoulli trials as a function of the variance as well as $\mu, a$, and $n$, as in [42].

A final application is to the *semirandom* source introduced by Santha and Vazirani [38]. A random source that outputs bits $X_1, X_2, \ldots, X_n$ is defined to be $\varepsilon$-semirandom in [38] if

$$\forall i \; 1/2 - \varepsilon \leq \Pr(X_i = 1 | X_1, X_2, \ldots, X_{i-1}) \leq 1/2 + \varepsilon,$$

i.e., the random bits can be correlated, but only to a limited extent, independently of the past history. Despite its seemingly weak nature, such a model has been shown to be able to simulate complexity classes such as RP (Vazirani and Vazirani [50], [51]), and the study of a generalization of this model by Zuckerman [54] has led to significant results recently (Nisan and Zuckerman [30], Wigderson and Zuckerman [53]). Noting that for such a source,

$$E\left[\prod_{j=1}^{k} X_{i_j}\right] \leq \left(\frac{1}{2} + \varepsilon\right)^k$$

for all $k \geq 1$ and all distinct indices $i_1, i_2, \ldots, i_k$, we see that

$$\Pr\left(\sum_{i=1}^{n} X_i \geq n\left(\frac{1}{2}+\varepsilon\right)(1+\delta)\right) \leq U_2\left(n, n\left(\frac{1}{2}+\varepsilon\right), \delta\right), \quad \forall\delta > 0$$

for an $\varepsilon$-semirandom source.

Inequality (14) shows another application of our techniques.

**3. Applications to computation.** The most striking point of Theorems 1 and 5 is that bounds as good as the CH bounds can be obtained with small independence. This implies, for any analysis that relies on the CH bounds, much weaker requirements on the random sources used. We now present some further computational applications of the new results.

**3.1. Reduced independence for randomized algorithms.** There are known constructions of r.v.'s with limited independence using a small number of random bits. Examples include: the construction of [19], the use of universal hash functions [9] to generate $|F|$ many $k$-wise independent random elements from a finite field $F$ using $O(k \log |F|)$ random bits, and the result of [1] using coding techniques [25], which gives a polynomial (in $n$) time algorithm to construct $n$ $k$-wise independent and unbiased random bits given $O(k \log n)$ independent unbiased bits for any $k, k \leq n$. Combining these with our result on reduced independence for the CH bounds, we get a major reduction in the amount of randomness needed for several randomized algorithms.

**3.1.1. Reduced randomness of random sampling.** In random sampling, we have a huge finite universe $U$ and a subset $W \subseteq U$, and we want to estimate the fraction $f^* = |W|/|U|$. Given error parameters $\delta, \varepsilon > 0$, the method used is to pick a random sample $S$ of size $N(\delta, \varepsilon)$ from $U$ and output the fraction $f$ of type $W$ elements in $S$; $N(\delta, \varepsilon)$ must be such that $\Pr(|f^* - f| \geq \delta) \leq \varepsilon$. This is required, for instance, in PAC learning [47] and running BPP algorithms. What was known so far is that $N(\delta, \varepsilon) = O(1/\delta^2)\log(\frac{1}{\varepsilon})$ with all the samples being independent. We can improve this to the following theorem.

THEOREM 8. *Given a universe $U$, a subset $W \subset U$, and error parameters $\delta, \varepsilon > 0$, suppose a set $S$ of $O(1/\delta^2)\log(\frac{1}{\varepsilon})$ random samples with $O(\log(\frac{1}{\varepsilon}))$-wise independence are picked from $U$. Then, if $f^*$ and $f$ are the respective fractions of type $W$ elements in $U$ and $S$, $\Pr(|f^* - f| \geq \delta) \leq \varepsilon$ will hold.*

*Proof.* Consider a randomized algorithm which looks at a random set of samples $S$ from $U$, and outputs the ratio $f$ of type $W$ elements in $S$. We now look at the random properties of $S$ that are required for the claim $\Pr(|f^* - f| \geq \delta) \leq \varepsilon$ to hold.

Let $|S| = n$. In the notation of Theorem 5, we want to claim that $\Pr(|X - \mu| \geq \delta'\mu) \leq \varepsilon$, where $\mu = E[X], X = nf$, and $\delta' = \delta/f^*$. We apply Theorem 5 with $X = fn$ and $\mu = f^*n$ and choose $k^*$ consistent with parts (Ia) and (IIa). For such a suitable choice of independence $k^*$ among the elements of $S$, $\Pr(|X - \mu| \geq \delta'\mu)$ can be bounded by $e^{-\lfloor k^*/2 \rfloor}$. Hence, it suffices to choose $k^* \geq 2\lceil \ln(\frac{1}{\varepsilon})\rceil$, for the above probability to be bounded by $\varepsilon$. To compute the required sample size $n$, we distinguish between the two cases $\delta' \leq 1$ and $\delta' > 1$. Then, from part (Ia) of Theorem 5, the probability that $|X - \mu|$ is greater than $\delta'\mu$ is bounded by $e^{-\lfloor k^*/2 \rfloor}$ provided that $k^* \leq \lfloor \delta'^2 \mu e^{-1/3} \rfloor$, and $k^* \geq 2\lceil \ln(\frac{1}{\varepsilon})\rceil$ for this probability to be bounded by $\varepsilon$. It therefore suffices to choose $\delta'^2\mu e^{-1/3} \geq 2\lceil \ln(\frac{1}{\varepsilon})\rceil$. This will hold if $n\delta^2 e^{-1/3}/(2f^*) \geq \lceil \ln(\frac{1}{\varepsilon})\rceil$. Because $e^{-1/3}/2 > 1/3$, it suffices to choose $n \geq N_1(\delta, \varepsilon) = (3f^*/\delta^2)\lceil \ln(\frac{1}{\varepsilon})\rceil$. If $\delta' > 1$, then a similar analysis using Theorem 5(IIa) implies that $N_2(\delta, \varepsilon) = \frac{3}{\delta}\lceil \ln(\frac{1}{\varepsilon})\rceil$ many samples with $2\lceil \ln(\frac{1}{\varepsilon})\rceil$-wise independence suffice to satisfy the error bounds.

Note that since both $f^*$ and $\delta$ are clearly bounded by 1, the number of samples and independence needed in both the above cases can be upper bounded by $N_3(\delta, \varepsilon) = (3/\delta^2)\lceil \ln(\frac{1}{\varepsilon})\rceil$ and $k^* = 2\lceil \ln(\frac{1}{\varepsilon})\rceil$. □

By Theorem 5 the choice for the independence that minimizes the error bound is an increasing function of the sample size; however, increasing the sample space size when given a fixed independence will also reduce the error probability; a proof of this claim follows. In the proof of Theorem 5, parts (I) and (II) were derived from part (III) of Theorem 4 with $C = \mu$. Note that in the current problem, $C = nf^*$ and $T = n\delta$, where $n$ is the number of random samples picked (in the notation of Theorem 4). When the independence $k$ is fixed, the bound given by part (III) of Theorem 4 decreases with $n$ for these values of $C$ and $T$, and the claim follows.

Theorem 4 hence also allows an estimate for the required size of a sample space with $k$-wise independent variables, in case $k < 2\lceil \ln(\frac{1}{\varepsilon})\rceil$.

THEOREM 9. *Given a universe $U$, a subset $W \subseteq U$, and error parameters $\delta, \varepsilon > 0$, suppose that $S$ is a sample space of $U$ whose elements are $k$-wise independent for some even $k$. Then, if $f^*$ and $f$ are the respective fractions of type-$W$ elements in $U$ and $S$, for $\Pr(|f^* - f| \geq \delta) \leq \varepsilon$ to hold, it is sufficient to choose $|S| \geq ck/\delta^2\varepsilon^{(2/k)}$ for some constant $c$.*

*Proof.* Use part (III) of Theorem 4 by setting $C = n$, where $n = |S|$. □

Theorems 8 and 9 imply "reduced randomness" results for random sampling if the universe $U$ has some properties. For instance, if $U$ is a finite field and the field operations can be done in polynomial (in $\frac{1}{\delta}$ and $\log(\frac{1}{\varepsilon})$) time, then any number of $k$-wise independent samples from $U$ can be generated from $k$ independent random samples [19], [9]. Also, via weaker bounds on the $k$th moment, it has been independently shown in [7] that essentially the same bounds as those given in Theorem 8 can be obtained for random sampling; [7] also shows how iterated sampling can be used to decrease the number of random bits at the expense of a controlled increase in the sample size.

The above constructions are not optimal with regard to the minimum number of random bits used. Using random walks on expander graphs to generate the random bits, it is shown in [6] that $O(\log(|U|) + \log(\frac{1}{\delta}))$ random bits are necessary and sufficient for this problem. Our construction has the advantage of being elementary and parallelizable.

### 3.1.2. Reduced randomness for oblivious permutation routing.

We now show how our results directly imply bounds that match the explicit constructions of algorithms with reduced randomness of Peleg and Upfal [32] for *oblivious permutation routing on fixed interconnection networks* (see also [20], [35], [36], [46], [48]).

Given some interconnection network with $N$ nodes and a permutation $\sigma : \{1, 2, \ldots, N\} \rightarrow \{1, 2, \ldots, N\}$, the problem is to route a packet $\nu_i$ residing at each node $i$ to its destination $\sigma(i)$ so that the total time taken is small. Furthermore, the routing must be *oblivious* in that the path $P_\sigma(x)$ chosen for a packet $x$ must be "independent" of the path $P_\sigma(y)$ chosen for any other packet $y$. (See [32] for a precise definition when randomized routing protocols are allowed.) Explicit constructions of algorithms with a spectrum of time-randomness parameters are among the results proved in [32] for the degree-4 butterfly network; these are also extendable to other networks. (See Karloff and Raghavan [20] for a protocol for the hypercube with slightly weaker bounds.) Here, we show how our results of §2.1 directly imply the bounds of [32] for the hypercube; we believe that similar results should hold for other interconnection networks.

Consider the implementation of Valiant's two-phase scheme [46] (see also Valiant and Brebner [48]) on a hypercube with $N = 2^n$ nodes. Phase (I) Each vertex $i$ picks a random $\rho(i) \in \{1, 2, \ldots, N\}$ as an *intermediate destination* for $\nu_i$ and routes $\nu_i$ there. Phase (II) Each packet $\nu_i$ is routed to its final destination $\sigma(i)$. We now follow the discussion of the standard aspects of this from [35]. Assume FIFO queues at each edge, and that Phase (I) routes $\nu_i$ from $i$ to $\rho(i)$ by "correcting" its bits from left to right, assuming that the nodes of the hypercube are indexed by $n$ bits, and Phase (II) "corrects" bits right to left. So, Phase (II) is like "running Phase (I) backwards," and so we consider Phase (I) alone here. It is shown in [35] that the time taken for packet $\nu_i$ in Phase (I) is at most

$$n + \sum_{j=1}^{N} H_{ij}, \tag{16}$$

where $H_{ij} = 1$ if the paths $\langle i, \rho(i)\rangle$ and $\langle j, \rho(j)\rangle$ share an edge in Phase (I) and 0 otherwise (recall that $n = \log_2 N$). It is also shown in [35] that if each $\rho(i)$ is uniformly distributed in $\{1, 2, \ldots, N\}$, then for all $i, E[\sum_{j=1}^{N} H_{ij}] \leq n$. Here is the theorem that matches the explicit construction of [32].

THEOREM 10. *There are explicit constructions of oblivious routing algorithms on the hypercube that, for any $T, c \log N \leq T \leq \sqrt{N}$ ($c > 4$ is a constant),*

(I) *use $O(\log N \log(N/Q)/\log(T/\log N))$ random bits and terminate in $T$ steps with probability at least $1 - Q$ for any $0 < Q \leq 1$, and*

(II) *use $O(\log^2 N/\log(T/\log N))$ random bits and terminate in expected time at most $T$.*

*Proof* (sketch.) Consider any packet $\nu_i$; the probability that it takes more than $\frac{T}{2}$ steps in Phase (I) is at most $\Pr(\sum_{j=1}^{N} H_{ij} \leq \frac{T}{2} - \log N)$. If the $\rho(i)$s are picked uniformly and in $k$-wise independent fashion, then the $H_{ij}$ are $(k-1)$-wise independent, while $E[\sum_{j=1}^{N} H_{ij}] \leq \log N$ as before. It follows from our discussion of §2.1 and Lemma 3 that, if $T > E[\sum_{j=1}^{N} H_{ij}]$, (i.e., if $T > 4 \log N$), then

$$\Pr\left(\sum_{j=1}^{N} H_{ij} \geq T/2 - \log N\right) \leq \frac{\binom{N}{k-1}\left(\frac{\log N}{N}\right)^{k-1}}{\binom{T/2 - \log N}{k-1}} \leq \frac{(\log N)^{k-1}}{\prod_{j=0}^{k-2}(T/2 - \log N - j)}.$$

By picking $k = \Theta(\log(N/Q)/\log(T/\log N))$, we can ensure that $\Pr(\sum_{j=1}^{N} H_{ij} \geq T/2 - \log N) \leq \frac{Q}{(2N)}$ holds. Arguing similarly for Phase (II) and summing up over all $i$, we get (I) above.

For (II) above, we set $Q = \frac{1}{(2N)}$ and replace $T$ by $T - 1$ in (I). Let $T_{\max}$ be an r.v. denoting the time taken by the protocol, i.e., the maximum over all packets $i$ of the number of steps taken by packet $\nu_i$ to reach $\sigma(i)$. Note that $T_{\max} \leq \log N + N$ from upper bound (16). Also,

$$\Pr(T_{\max} > (T-1)) \leq Q$$

from (I) above. Hence,

$$\begin{aligned} E[T_{\max}] &\leq (T-1) \cdot \Pr(T_{\max} \leq (T-1)) + (\log N + N) \cdot \Pr(T_{\max} > (T-1)) \\ &\leq (T-1)\left(1 - \frac{1}{2N}\right) + (\log N + N)\frac{1}{2N} \\ &\leq T. \end{aligned}$$

Note that for any $k$, $k$-wise independent $\rho(i)$'s can be generated from $k \log N$ random bits using hash functions [9], since the $\rho(i)$'s can be thought of as belonging to the field $GF(2^n)$. Hence, we get bounds that match those of [32]. $\square$

The above example typifies the type of application we expect our methods to find, i.e., as direct "plug-ins" in analyses where the CH bounds are normally used.

**3.2. The new formulation and the method of conditional probabilities.** The *method of conditional probabilities* [34], [44] is an important technique for the derandomization of algorithms; the reader is referred to [35] for details. We now show how this method can be combined with the formulation of §2.6. This will enable us to derive simple and efficient deterministic polynomial-time algorithms from randomized algorithms that can be analyzed using our formulation *in a unified way.*

Given $n$ random bits $X_1, X_2, \ldots, X_n$, can the conditional expectation

$$E[S_k(X_1, X_2, \ldots, X_n) | X_1 = b_1, X_2 = b_2, \ldots, X_i = b_i]$$

be evaluated, or at least be given a "reasonable" upper bound, for any $k$, any $i, i = 0, 1, 2, \ldots, n-1$, and any $b_1 b_2 \cdots b_i \in \{0,1\}^i$? If the $X_i$'s are identically distributed, then it is reasonable to assume that an upper bound $U_\ell$ on $E[\prod_{r=1}^{\ell} X_{j_r} | X_1 = b_1, X_2 = b_2, \ldots, X_i = b_i]$ is known for all $\ell, \ell = 1, 2, \ldots, n-i$, and for all distinct indices $X_{j_1}, X_{j_2}, \ldots, X_{j_\ell} \in \{i+1, i+2, \ldots, n\}$; this is sufficient for the two applications shown below. Then, if

$$|\{j | (1 \le j \le i) \wedge (b_j = 1)\}| = i_1,$$

we can see that

$$\begin{aligned} &E[S_k(X_1, X_2, \ldots, X_n) | X_1 = b_1, X_2 = b_2, \ldots X_i = b_i] \\ (17) \qquad &\le \sum_{r=0}^{\min(i_1,k)} \binom{i_1}{r} (\, n - ik - r\,) U_{k-r}. \end{aligned}$$

We now present two applications in which the combination of our formulation and the method of conditional probabilities leads to fast polynomial-time algorithms, via upper bound (17). The first application, to jobshop scheduling, is a "natural" derandomization of the randomized algorithm of [41], faster than the derandomization techniques of [41] and [33]; this is shown in §3.2.1. The second application is to discrepancy theory and is discussed in §3.2.2. The "usual" method of conditional probabilities for these problems frequently calls for independence among the r.v.'s corresponding to the bits $X_1, X_2, \ldots, X_n$ seen above; this is not the case for these two problems and in general for many other problems.

**3.2.1. Improved algorithms for packet routing and jobshop scheduling.** We now present simpler approximation algorithms for *packet routing* (Leighton, Maggs, and Rao [23]) and *jobshop scheduling* (Shmoys, Stein, and Wein [41]) that provide improved approximation guarantees by using ideas from above. The *nonpreemptive jobshop scheduling* problem is as follows. Given $n$ jobs, $m$ machines, and a sequence of *operations* for each job, where each operation is assigned to a specific machine, construct a schedule to run the jobs so that the time taken to process all the jobs is minimized, subject to the following conditions: (a) the operations of each job must be done in sequence; (b) no operation of any job running on any machine can be preempted until it is completed; and (c) a machine can process at most one operation at a time. One of the results of [23] tackles a special case of this problem; the general case is handled in [41]. Both of these papers give polynomial-time algorithms to produce good approximations to an optimal schedule.

Let $P_i$ be the total time needed for job $J_i$, $P_{\max} = \max_{i\in[1,n]} P_i$, $\Pi_j$ be the total time for which machine $M_j$ is needed, and $\Pi_{\max} = \max_{i\in[1,m]} \Pi_i$. Before an actual schedule is constructed in [41], a pseudoschedule $\mathcal{S}$ is constructed that temporarily assumes that each machine can work on up to $D$ operations simultaneously, where $D > 1$ depends on the input instance. The pseudoschedule is later used to construct an actual schedule. The only step in which randomization is used in [41] is during the construction of the pseudoschedule and is the following.

An initial random delay $d_i \in \{1, 2, \ldots, \Pi_{\max}\}$ is assigned for each job $J_i$. Suppose that the sequence of operations of job $J_i$ is $O_{i,1}, \cdots O_{i,r_i}$ and operation $O_{i,r}$ takes time $t_{i,r}$; then, in the pseudoschedule $\mathcal{S}$, job $J_i$ is scheduled to start at time $d_i$ and runs to completion without interruption, i.e., operation $O_{i,r}$ starts at time $d_i + \sum_{\ell=1}^{r-1} t_{i,\ell}$. We denote the offset $\sum_{\ell=1}^{r-1} t_{i,\ell}$ by $\tau(O_{i,r})$. As shown in [23], [41], if the $d_i$'s are generated uniformly and independently, then with high probability, every machine at every unit of time will have (a *congestion* of) at most $D(n, m_{\max}) \doteq c \cdot \log(n \cdot m_{\max})/\log\log(n \cdot m_{\max})$ jobs scheduled on it for some constant $c$, where $m_{\max}$ is the maximum number of operations in any job. This step is then derandomized to deterministically compute initial delays leading to a congestion bound of $O(\log(n \cdot m_{\max}))$. Linear programming is used for the derandomization, making this step the bottleneck. This step is sped up in [33], [45]. Here, we get a better congestion bound of $D(n, m_{\max})$, as opposed to the previously known $O(\log(n \cdot m_{\max}))$ bound, with an algorithm that is more direct than those of [33], [45] while having time complexities comparable to those.

We assign random initial delays $\{d_i \in \{1, 2, \ldots, \Pi_{\max}\}\}$ uniformly and independently to the jobs. Suppose that the operations scheduled on machine $M_i$ are $O_1, \ldots, O_{m_i}$, which belong to jobs $J_{i_1}, J_{i_2}, \ldots, J_{i_{m_i}}$, respectively, and take $t_1, \ldots, t_{m_i}$ units of time. For any machine $M_i$ and time instance $t \in \{1, 2, \ldots, \Pi_{\max} + P_{\max}\}$, we define $\sum_{\ell\in[1,m]} t_\ell = \Pi_i$ many indicator r.v.'s $X_j^i(t), j = 1, 2, \ldots, \Pi_i$, to analyze the congestion on machine $M_i$ at time $t$ in $\mathcal{S}$. Each of these r.v.'s is an indicator for the event that a particular unit of time of some operation gets scheduled on $M_i$ at time $t$ in $\mathcal{S}$ as follows. The index $j$ encodes the time unit and operation. Let $j_r = \sum_{\ell=1}^{r-1} t_\ell$ for $r = 1, 2, \ldots, m_i$; then if $j_r < j \le j_{r+1}, j$ represents the $(j - j_r)$th time unit of $O_r$ as follows.

$$X_j^i(t) = \begin{cases} 1 & \text{if } j = j_r + p, 1 \le p \le t_r \text{ and the } p\text{th time unit of } O_r \\ & \text{is scheduled for time } t\text{, i.e., if } d_{i_r} + \tau(O_r) + p - 1 = t; \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that $E[X_j^i(t)] \le 1/\Pi_{\max}$ and, for any positive integer $k$, the probability that machine $M_i$ has congestion at least $k$ at time unit $t$ is

$$\Pr\left(\sum_{r=1}^{\Pi_i} X_r^i(t) \ge k\right) \le E[S_k(X_1^i(t), \ldots, X_{\Pi_i}^i(t))] \le \binom{\Pi_i}{k}\left(\frac{1}{\Pi_{\max}}\right)^k.$$

In addition, if $\sum_{r=1}^{\Pi_i} X_r^i(t) \ge k$ holds for some time $t$, then $\sum_{r=1}^{\Pi_i} X_r^i(t') \ge k$ also holds for some time $t'$, where $t'$ is one of the starting times of the operations scheduled on $M_i$. Furthermore, the starting time of each operation $O_r$ is uniformly distributed in $[\tau(O_r), \Pi_{\max} - 1 + \tau(O_r)]$. Hence, for any $k$, Pr (some machine has congestion at least $k$ at some time instance) is at most

$$\text{(18)} \qquad \sum_{i=1}^{m} \sum_{r=1}^{m_i} \sum_{t=\tau(O_r)}^{\Pi_{max}-1+\tau(O_r)} \frac{1}{\Pi_{\max}} E\left[S_{k-1}\left(X_1^i(t), \ldots, X_{\Pi_i}^i(t)\right)\right],$$

which is at most

$$\sum_{i=1}^{m} m_i \frac{\Pi_{\max}}{\Pi_{\max}} \binom{\Pi_i}{k-1} \left(\frac{1}{\Pi_{\max}}\right)^{k-1} \leq \sum_{i=1}^{m} \frac{m_i}{(k-1)!}.$$

Clearly $\sum_{i=1}^{m} m_i \leq n \cdot m_{\max}$, and hence for $k - 1 > k^* = c_1 \log(n \cdot m_{\max}) / \log\log(n \cdot m_{\max})$ for some suitable constant $c_1$, the above probability estimate is less than 1. We may now use the above form as a *pessimistic estimator* [34] to deterministically set the delays $d_i$ for the jobs one-by-one by the method of conditional probabilities [34], [44] to achieve the congestion bound of $D(n, m_{\max})$.

Assume inductively that initial delays $d_1 = d_1^*, d_2 = d_2^*, \ldots, d_s = d_s^*$ have been set deterministically for the jobs $J_1, J_2, \ldots, J_s$; the aim is to compute $d_{s+1}^*$. Consider any machine $M_i$ on which $J_{s+1}$ has at least one operation; let these operations be $A_{s+1,1}, A_{s+1,2}, \ldots, A_{s+1,a_i}$. Let $B_{s,1}, B_{s,2}, \ldots, B_{s,b_i}$ be the operations that belong to some job in $\{J_1, J_2, \ldots, J_s\}$ and are scheduled on $M_i$ and let $t_{s,1}, t_{s,2}, \ldots, t_{s,b_i}$ be the times at which they are scheduled to start on $M_i$; these times are known, since we know the values of $d_1, d_2, \ldots, d_s$. Let $O_1, O_2, \ldots, O_{c_i}$ be the operations on machine $M_i$ that belong to jobs from the set $\{J_{s+2}, J_{s+3}, \ldots, J_n\}$. For any $t \in \{1, 2, \ldots, P_{\max} + \Pi_{\max}\}$ and $r \in \{1, 2, \ldots, \Pi_{\max}\}$, we define

$$\begin{aligned} g(s+1, i, t, r) \doteq E[S_{k^*}(X_1^i(t), X_2^i(t), \ldots, X_{\Pi_i}^i(t)) | d_1 \\ = d_1^*, d_2 = d_2^*, \ldots, d_s = d_s^*, d_{s+1} = r] \end{aligned}$$

and $\text{num}(i, t, \langle x_1, x_2, \ldots, x_j \rangle)$ to be the number of operations from jobs $J_1, J_2, \ldots, J_j$ that are scheduled on machine $M_i$ at time $t$ given that $d_1 = x_1, \ldots, d_j = x_j$.

When conditioned on the event $d_1 = d_1^*, d_2 = d_2^*, \ldots, d_s = d_s^*, d_{s+1} = \Delta$ for any $\Delta \in \{1, 2, \ldots, \Pi_{\max}\}$, upper bound (18) becomes

$$\begin{aligned} f(s+1, \Delta) \doteq \sum_{i=1}^{m} \Bigg( & \sum_{j=1}^{a_i} g(s+1, i, \tau(A_{s+1,j}) + \Delta, \Delta) + \sum_{j=1}^{b_i} g(s+1, i, t_{s,j}, \Delta) \\ & + \sum_{j=1}^{c_i} \sum_{t=\tau(O_j)}^{\tau(O_j)+\Pi_{\max}-1} \frac{1}{\Pi_{\max}} g(s+1, i, t, \Delta) \Bigg). \end{aligned}$$

Recall that the method of conditional probabilities will set $d_{s+1} = d_{s+1}^*$, where $d_{s+1}^*$ is the index at which $f(s+1, \cdot)$ is minimized. Note that $f(s+1, \Delta)$ can be readily computed for any $\Delta$ and, hence, so can $d_{s+1}^*$. To make the computation more efficient, we use the following observations.

(a) Suppose we need to compute $f(s+1, \Delta)$ for some $\Delta$ using upper bound (17). Then, for any machine $M_j$ that has some operation from job $J_{s+1}$, the term $n - i$ in (17) corresponds to the number of operations of jobs $J_1, J_2, \ldots, J_{s+1}$ on machine $M_j$, i.e., $a_j + b_j$. Hence, for any $t \in \{1, 2, \ldots, P_{\max} + \Pi_{\max}\}, g(s+1, j, t, \Delta)$ can be computed in $O(k^*)$ time if $\mathrm{num}(j, t, \langle d_1^*, d_2^*, \ldots, d_s^*, \Delta\rangle)$ is known, since upper bound (17) involves a sum over at most $k^*$ terms. (Recall that $k^*$ is an upper bound on the number of operations scheduled at the same time.)

(b) Suppose inductively that $\mathrm{num}(i, t, \langle d_1^*, d_2^*, \ldots, d_s^*\rangle)$ is known for all machines $M_i$ and all times $t$. Let $\omega_{s+1}$ be the number of operations of job $J_{s+1}$. We will consider only those machines that have some operation of $J_{s+1}$; the number of such machines is clearly at most $\omega_{s+1}$. Hence, given the $\mathrm{num}(i, t, \langle d_1^*, d_2^*, \ldots, d_s^*\rangle)$ values, the $\mathrm{num}(\cdot, \cdot, \langle d_1^*, d_2^*, \ldots, d_s^*, 1\rangle)$ values need to be updated for at most $\omega_{s+1}$ machines and $P_{\max} + \Pi_{\max}$ time units, which can be done in $O(\omega_{s+1}(P_{\max} + \Pi_{\max}))$ time. Given the $\mathrm{num}(\cdot, \cdot, \langle d_1^*, d_2^*, \ldots, d_s^*, 1\rangle)$ values, the computation of $f(s + 1, 1)$ takes $O(\omega_{s+1}k^*(P_{\max} + \Pi_{\max}))$ time, since $g(\cdot, \cdot, 1)$ can be computed in $O(k^*)$ time, given these values.

(c) Suppose we have computed the $\mathrm{num}(\cdot, \cdot, \langle d_1^*, d_2^*, \ldots, d_s^*, \Delta\rangle)$ values and $f(s + 1, \Delta)$ for some value $\Delta$ and that we need to compute $f(s + 1, \Delta + 1)$. We can first compute the $\mathrm{num}(\cdot, \cdot, \langle d_1^*, d_2^*, \ldots, d_s^*, \Delta + 1\rangle)$ values and then $f(s+1, \Delta+1)$ as follows. Suppose some operation $\alpha$ of $J_{s+1}$ is scheduled to run on some machine $M_i$ from time $t_1$ to time $t_2$ when we set $d_{s+1} = \Delta$. Then

$$\mathrm{num}(i, t', \langle d_1^*, d_2^*, \ldots, d_s^*, \Delta + 1\rangle) = \mathrm{num}(i, t', \langle d_1^*, d_2^*, \ldots, d_s^*, \Delta\rangle), \quad \forall t' \in [t_1 + 1, t_2].$$

Hence, this operation $\alpha$ leaves at most two of the $\mathrm{num}(\cdot, \cdot, \langle d_1^*, d_2^*, \ldots, d_2^*, \Delta + 1\rangle)$ values different from the corresponding $\mathrm{num}(\cdot, \cdot, \langle d_1^*, d_2^*, \ldots, d_s^*, \Delta\rangle)$ values. Hence, $\mathrm{num}(\cdot, \cdot, \langle d_1^*, d_2^*, \ldots, d_s^*, \Delta + 1\rangle)$ can be updated in $O(\omega_{s+1})$ time. It now follows from arguments similar to those used above that $f(s + 1, \Delta + 1)$ can be computed in $O(k^*\omega_{s+1})$ time.

The above observations imply an efficient algorithm to compute $d_{s+1}$. Inductively maintain the $\mathrm{num}(\cdot, \cdot, \langle d_1^*, d_2^*, \ldots, d_s^*, r\rangle)$ values as $r$ goes from $1, 2, \ldots,$ to $\Pi_{\max}$ and compute $f(s + 1, 1), f(s + 1, 2), \ldots, f(s + 1, \Pi_{\max})$ in that order by sequentially updating the corresponding num values. Since computing $d_{s+1}^*$ takes $O(\omega_{s+1}k^*(P_{\max} + \Pi_{\max}))$ time, the total time complexity is $O(\omega k^*(P_{\max} + \Pi_{\max}))$, where $\omega$ is the total number of operations. Hence, we have the following theorem.

THEOREM 11. *Initial delays* $\{d_i : 1 \leq i \leq n\}$ *in the range* $\{1, 2, \ldots, \Pi_{\max}\}$ *for each job* $J_i$ *can be set in* $O((P_{\max} + \Pi_{\max})\omega \log(n \cdot m_{\max})/\log\log(n \cdot m_{\max}))$ *time, where* $\omega$ *is the total number of operations, such that in the (infeasible) schedule in which every job* $J_i$ *starts at time* $d_i$ *and runs without interruption, every machine has at most* $O(\log(n \cdot m_{\max})/\log\log(n \cdot m_{\max}))$ *jobs scheduled on it at any time.*

We feel that the above is a natural derandomization of the randomized algorithm, since it sets the delays one-by-one, as opposed to the more complex methods used before.

**3.2.2. Exact partitions in set discrepancy.** Set discrepancy problems [3] are combinatorially important, and special cases of these can model divide-and-conquer situations; see, e.g., the RNC edge-coloring algorithm of Karloff and Shmoys [21]. Given a finite set $X$ and a family of subsets $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ of $X$, the goal is to come up with a 2-coloring $\chi : X \to \{0, 1\}$ such that the *discrepancy* $\mathrm{disc}(\chi) \doteq \max_i \mathrm{disc}_i(\chi)$, where $\mathrm{disc}_i(\chi) \doteq \{|(\sum_{j\in S_i} \chi(j)) - |S_i|/2|\}$, is minimized. It is known that a 2-coloring $\chi$ with $\mathrm{disc}(\chi) = O(\sqrt{\Delta \log n})$ exists and can be computed in polynomial (in $|X|$ and $n$) time [3] and that a 2-coloring $\chi$ with $\mathrm{disc}(\chi) = O(\Delta^{0.5+\varepsilon}\sqrt{\log n})$ for any fixed $\varepsilon > 0$ can be computed in NC [8], [27], [29], where $\Delta \doteq \max_i |S_i|$. Using the ideas of §2.1, we can prove the following theorem.

THEOREM 12. *Given a finite set $X$ with $|X|$ even and a family of subsets $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ of $X$ such that $\Delta = \max_i |S_i|$, there exists a 2-coloring $\chi^* : X \to \{0,1\}$ computable in polynomial (in $|X|$ and $n$) time such that* (I) $\mathrm{disc}(\chi^*) = O(\sqrt{\Delta \log n})$ *and* (II) $|\{y \in X : \chi^*(y) = 0\}| = |\{y \in X : \chi^*(y) = 1\}|$.

*Proof.* For the existence proof, we can show that if we pick a random subset $Z \subseteq X$ with $|Z| = \frac{|X|}{2}$ uniformly from the set of all size $\frac{|X|}{2}$ subsets of $X$ and set $\chi_Z(y) = 1$ iff $y \in Z$, then $\Pr_Z(\mathrm{disc}(\chi_Z) = O(\sqrt{\Delta \log n})) > 0$ as follows. It is well known [3] and easily checked via the CH bounds that there is a constant $c > 0$ such that if $\chi(y)$ is picked uniformly and independently from $\{0,1\}$, then for any $S_i, \Pr(\mathrm{disc}_i(\chi) > c\sqrt{\Delta \log n}) < \frac{1}{n}$; hence,

$$\Pr(\mathrm{disc}(\chi) > c\sqrt{\Delta \log n}) = \Pr(\exists i : \mathrm{disc}_i(\chi) > c\sqrt{\Delta \log n}) < n \cdot \frac{1}{n} = 1.$$

Note that $\{\chi_Z(y) : y \in X\}$ is a set of self-weakening random bits with parameter 1, i.e., if $Z$ is picked uniformly at random from the set of $\frac{|X|}{2}$ sized subsets of $X$, then for any distinct $y_1, y_2, \ldots, y_i \in X$,

$$E\left[\prod_{j=1}^{i} \chi_Z(y_j)\right] = \Pr(\chi_Z(y_1) = \chi_Z(y_2) = \cdots = \chi_Z(y_i) = 1) \leq \prod_{j=1}^{i} \Pr(\chi_Z(y_j) = 1)$$
$$= \prod_{j=1}^{i} E[\chi_Z(y_j)].$$

Hence, it follows from §2.6 that for any $S_i, \Pr(\mathrm{disc}_i(\chi_Z) > c\sqrt{\Delta \log n}) < \frac{1}{n}$ still holds, concluding the existence proof.

Assume that $|X| = m$ and that $X = \{1, 2, \ldots, m\}$. To use the method of conditional probabilities to derandomize the above construction, note that for $\{i_1, i_2, \ldots, i_j\} \subseteq \{i+1, i+2, \ldots, m\}$,

$$\begin{aligned} E\left[\prod_{\ell=1}^{j} \chi_Z(i_\ell) | \chi_Z(1) = b_1, \ldots, \chi_Z(r) = b_r\right] &= \text{undefined} \quad \text{if } r_1 > \frac{m}{2} \text{or } r_2 > \frac{m}{2}, \\ &= 0 \quad \text{if } r_1 + j > \frac{m}{2}, \\ &= \frac{\binom{m-r-j}{m/2-r_1-j}}{\binom{m-r}{m/2-r_1}} \quad \text{otherwise,} \end{aligned}$$

where

$$|\{\ell | (1 \leq \ell \leq r) \wedge (b_\ell = 1)\}| = r_1$$

and $r_2 = r - r_1$. This can now be derandomized as shown in our initial discussion in §3.2. □

**Acknowledgments.** Aravind Srinivasan thanks David Shmoys for his constant encouragement, guidance, and perceptive suggestions. This author also thanks Suresh Chari, Alessandro Panconesi, and Pankaj Rohatgi for valuable discussions. All three authors also thank Mihir Bellare for pointing out references [6] and [7].

## REFERENCES

[1] N. Alon, L. Babai, and A. Itai, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, J. Algorithms, 7 (1986), pp. 567–583.

[2] N. Alon, O. Goldreich, J. Håstad, and R. Peralta, *Simple constructions of almost k-wise independent random variables*, Random Structures Algorithms, 3 (1992), pp. 289–303.

[3] N. Alon, J. Spencer, and P. Erdős, *The Probabilistic Method*, Wiley–Interscience Series, John Wiley and Sons, Inc., New York, 1992.

[4] D. Angluin and L. G. Valiant, *Fast probabilistic algorithms for Hamiltonian circuits and matchings*, J. Comput. System Sci., 18 (1979), pp. 155–193.

[5] Y. Azar, R. Motwani, and J. Naor, *Approximating arbitrary probability distributions using small sample spaces*, manuscript, 1990.

[6] M. Bellare, O. Goldreich, and S. Goldwasser, *Randomness in interactive proofs*, in Proc. IEEE Symposium on Foundations of Computer Science, 1990, pp. 563–573.

[7] M. Bellare and J. Rompel, *Randomness efficient sampling of arbitrary functions*, Technical Report MIT/LCS/TM-433.b, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, July, 1990.

[8] B. Berger and J. Rompel, *Simulating* $(\log^c n)$*-wise independence in NC*, J. Assoc. Comput. Mach., 38 (1991), pp. 1026–1046.

[9] J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, J. Comput. System Sci., 18 (1979), pp. 143–154.

[10] S. Chari, P. Rohatgi, and A. Srinivasan, *Improved algorithms via approximations of probability distributions*, in Proc. ACM Symposium on Theory of Computing, 1994, pp. 584–592.

[11] H. Chernoff, *A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations*, Annals of Mathematical Statistics, 23 (1952), pp. 493–509.

[12] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. Smolensky, *The bit extraction problem or t-resilient functions*, in Proc. IEEE Symposium on Foundations of Computer Science, 1985, pp. 396–407.

[13] V. Chvátal, *The tail of the hypergeometric distribution*, Discrete Math., 25 (1979), pp. 285–287.

[14] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković, *Approximations of general independent distributions*, in Proc. ACM Symposium on Theory of Computing, 1992, pp. 10–16.

[15] B. V. Gladkov, *Sums of random variables, any r of which are independent*, Mat. Zametki, 32 (1982), pp. 385–399.

[16] ———, *A central limit theorem for sums of random variables, any r of which are independent*, Diskret. Mat., 1 (1989), pp. 22–28; Discrete Math. Appl., 1 (1991), pp. 73–79 (English translation).

[17] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Amer. Statist. 58 (1963), pp. 13–30.

[18] M. Hofri, *Probabilistic Analysis of Algorithms*, Springer-Verlag, New York, Berlin, 1987.

[19] A. Joffe, *On a set of almost deterministic k-independent random variables*, Ann. Probab., 2 (1974), pp. 161–162.

[20] H. J. Karloff and P. Raghavan, *Randomized algorithms and pseudorandom numbers*, in Proc. ACM Symposium on Theory of Computing, 1988, pp. 310–321.

[21] H. J. Karloff and D. B. Shmoys, *Efficient parallel algorithms for edge coloring problems*, J. Algorithms, 8 (1987), pp. 39–52.

[22] C. Kruskal, L. Rudolph, and M. Snir, *A complexity theory of efficient parallel algorithms*, Theoret. Comput. Sci., 71 (1990), pp. 95–132.

[23] F. T. Leighton, B. Maggs, and S. Rao, *Universal packet routing algorithms*, in Proc. IEEE Symposium on Foundations of Computer Science, 1988, pp. 256–269.

[24] N. Linial and N. Nisan, *Approximate inclusion–exclusion*, Combinatorica, 10 (1990), pp. 349–365.

[25] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes.* North–Holland, Amsterdam, 1977.

[26] K. Mehlhorn and U. Vishkin, *Randomized and deterministic simulations of PRAMs by parallel machines with restricted granularity of parallel memories*, Acta Inform., 21 (1984), pp. 339–374.

[27] R. Motwani, J. Naor, and M. Naor, *The probabilistic method yields deterministic parallel algorithms*, in Proc. IEEE Symposium on Foundations of Computer Science, 1989, pp. 8–13.

[28] S. V. NAGAEV AND I. F. PINELIS, *Some inequalities for the distribution of the sums of independent random variables*, Theory Probab. Appl., 22 (1977), pp. 248–256.

[29] J. NAOR AND M. NAOR, *Small-bias probability spaces: efficient constructions and applications*, in Proc. ACM Symposium on Theory of Computing, 1990, pp. 213–223.

[30] N. NISAN AND D. ZUCKERMAN, *More deterministic simulation in Logspace*, in Proc. ACM Symposium on Theory of Computing, 1993, pp. 235–244.

[31] A. PANCONESI AND A. SRINIVASAN, *Fast randomized algorithms for distributed edge coloring*, in Proc. ACM Symposium on Principles of Distributed Computing, 1992, pp. 251–262.

[32] D. PELEG AND E. UPFAL, *A time-randomness tradeoff for oblivious routing*, SIAM J. Comput., 19 (1990), pp. 256–266.

[33] S. A. PLOTKIN, D. B. SHMOYS, AND É. TARDOS, *Fast approximation algorithms for fractional packing and covering problems*, in Proc. IEEE Symposium on Foundations of Computer Science, 1991, pp. 495–504.

[34] P. RAGHAVAN, *Probabilistic construction of deterministic algorithms: approximating packing integer programs*, J. Comput. System Sci., 37 (1988), pp. 130–143.

[35] ———, *Lecture notes on randomized algorithms*, Technical Report RC 15340 (#68237), IBM T. J. Watson Research Center, Yorktown Heights, NY, January, 1990; CS661 Lecture Notes, Technical Report YALE/DCS/RR-757, Department of Computer Science, Yale University, New Haven, CT, January, 1990.

[36] A RANADE, *How to emulate shared memory*, J. Comput. System Sci., 41 (1991), pp. 307–326.

[37] H. ROBBINS, *A remark on Stirling's formula*, Amer. Math. Monthly, 62 (1955), pp. 26–29.

[38] M. SANTHA AND U. V. VAZIRANI, *Generating quasirandom sequences from semirandom sources*, J. Comput. System Sci., 33 (1986), pp. 75–87.

[39] J. P. SCHMIDT AND A. SIEGEL, *On aspects of universality and performance for closed hashing*, in Proc. ACM Symposium on Theory of Computing, 1989, pp. 355–366.

[40] ———, *The analysis of closed hashing under limited randomness*, in Proc. ACM Symposium on Theory of Computing, 1990, pp. 224–234.

[41] D. B. SHMOYS, C. STEIN, AND J. WEIN, *Improved approximation algorithms for shop scheduling problems*, in Proc. ACM/SIAM Symposium on Discrete Algorithms, 1991, pp. 131–140.

[42] A. SIEGEL, *Toward a usable theory of Chernoff bounds for heterogeneous and partially dependent random variables*, manuscript, September, 1992.

[43] ———, *On universal classes of fast hash functions, their time–space tradeoff, and their applications*, in Proc. IEEE Symposium on Foundations of Computer Science, 1989, pp. 20–25.

[44] J. SPENCER, *Ten Lectures on the Probabilistic Method*, Society for Industrial and Applied Mathematics, Philadelphia, 1987.

[45] C. STEIN, *Approximation algorithms for multicommodity flow and shop scheduling problems*, Ph.D. thesis and Technical Report MIT/LCS/TR-550, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1992.

[46] L. G. VALIANT, *A scheme for fast parallel communication*, SIAM J. Comput., 11 (1982), pp. 350–361.

[47] ———, *A theory of the learnable*, Communications of the ACM, 27 (1984), pp. 1134–142.

[48] L. G. VALIANT AND G. J. BREBNER, *Universal schemes for parallel communication*, in Proc. ACM Symposium on Theory of Computing, 1981, pp. 263–277.

[49] U. V. VAZIRANI, *Randomness, Adversaries, and Computation*, Ph.D. thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, 1986.

[50] U. V. VAZIRANI AND V. V. VAZIRANI, *Random polynomial time is equal to slightly random polynomial time*, in Proc. IEEE Symposium on Foundations of Computer Science, 1985, pp. 417–428.

[51] ———, *Random polynomial time is equal to semirandom polynomial time*, Technical Report 88-959, Department of Computer Science, Cornell University, Ithaca, NY, 1988.

[52] M. N. WEGMAN AND J. L. CARTER, *New hash functions and their use in authentication and set equality*, J. Comput. System Sci., 22 (1981), pp. 265–279.

[53] A. WIGDERSON AND D. ZUCKERMAN, *Expanders that beat the eigenvalue bound: Explicit construction and applications*, in Proc. ACM Symposium on Theory of Computing, 1993, pp. 245–251.

[54] D. ZUCKERMAN, *Simulating BPP using a general weak random source*, in Proc. IEEE Symposium on Foundations of Computer Science, 1991, pp. 79–89.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 251–257, May 1995

# NONREDUNDANT 1'S IN Γ-FREE MATRICES *

JEREMY P. SPINRAD†

**Abstract.** This paper studies a new method for representing Γ-free matrices, which occur in characterizations of chordal bipartite and strongly chordal graphs. We show that the number of Γ-free matrices with $n$ rows and columns (and thus the number of chordal bipartite and strongly chordal graphs with $n$ vertices) is proportional to $2^{\Theta(n \log^2 n)}$, and give an asymptotically space optimal method for storing these matrices.

**Key words.** chordal bipartite graphs, totally balanced matrices, strongly chordal graphs, $\beta$-acyclic hypergraphs

**AMS subject classifications.** 05B20, 05C30, 05C50

## 1. Introduction.

A matrix is Γ-free if it has no pair of rows and columns that induce a Γ, as described in the next section. Γ-free matrices were introduced by Lovász [10]. Hoffman, Kolen, and Sakarovitch [8] used Γ-free matrices to characterize when a particular linear programming problem could be solved using a greedy algorithm. They also showed that a graph is chordal bipartite, as defined below, if and only if the bipartite adjacency matrix has a Γ-free ordering. Anstee and Farber [13] showed that a graph is strongly chordal if and only if the neighborhood matrix has a Γ-free ordering. Lubiw [11] developed an efficient algorithm to determine whether a matrix has a Γ-free ordering, which led to efficient recognition algorithms for chordal bipartite and strongly chordal graphs. The fastest known recognition algorithms for these cases of graphs, both refinements of Lubiw's algorithm, are [13] for sparse graphs and [14] for dense graphs, with time complexities of $O(m \log n)$ and $O(n^2)$, respectively.

This paper studies the number of Γ-free matrices and therefore the number of chordal bipartite and strongly chordal graphs. We show that there are $2^{\Theta(n \log^2 n)}$Γ-free $n$ by $n$ matrices.

We derive this bound by using a new method for representing Γ-free matrices. We show that this representation is optimal in the sense that the number of bits used in the representation of an $n$ by $n$ Γ-free matrix is at most a constant times the logarithm of the number of Γ-free matrices.

## 2. Definitions.

A Γ in a matrix is a pair of rows and columns that induce the submatrix

$$\begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}.$$

A matrix that has a permutation of the rows and columns that are Γ-free is called a *totally balanced* matrix. Totally balanced matrices are useful in the study of linear programming [1], [8].

A bipartite graph is *chordal bipartite* if every cycle of length of at least 6 has a chord. Hoffman, Kolen, and Sakarovitch [8] showed that a graph is chordal bipartite if and only if the bipartite adjacency matrix has a Γ-free ordering.

* Received by the editors March 25, 1991; accepted for Publication (in revised form) May 31, 1994. This research was supported by National Security Agency grant R592-9632.

† Department of Computer Science, Vanderbilt University, Nashville, Tennessee 37235.

Γ-free matrices can also be used to characterize strongly chordal graphs, which are studied in [3], [7]. Farber [6] showed that a graph is strongly chordal if and only if its neighborhood matrix has a Γ-free ordering.

Totally balanced matrices also arise, under the name $\beta$-acyclic hypergraphs, in database theory [4], [5].

For a summary of work on totally balanced and Γ-free matrices, see Lubiw [11]. Lubiw also gives the key algorithms for fast recognition of Γ-free matrices and Γ-free ordering of totally balanced matrices.

**3. Nonredundant 1's.** This section introduces a new representation method for Γ-free matrices. We represent the matrix by giving the positions of a subset of the 1 entries. The subset must have the property such that if you know the subset and you know the resulting matrix is Γ-free, the matrix can be completely reconstructed.

A 1 entry in position $i, j$ of the matrix is called *redundant* if the replacement of the entry by a 0 would result in a Γ. For example, in the matrix

$$\begin{matrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{matrix},$$

the 1's in positions $(2,3), (3,2), (3,3)$ are redundant. As 1 entry that is not redundant is called nonredundant. The term redundant is used because these 1 entries can be deduced from the 1's in the other positions. We propose storing a Γ-free matrix by storing only the positions of the nonredundant 1's.

We note that it is possible to find all nonredundant 1's in the matrix in $O(n+m)$ time, where $m$ is the number of 1 entries in the matrix, if we are given the 1 entries in "adjacency list" form. The idea is similar to the algorithm developed by Lubiw [11] for verifying that a matrix is Γ-free in $O(n+m)$ time.

THEOREM 1. *There is an algorithm that can identify all nonredundant* 1*'s in a* Γ*-free matrix* $A$ *in* $O(n+m)$ *time, where* $m$ *is the number of nonzero entries in* $A$.

*Proof.* For each 1 entry $A[i,j]$, let $j'$ be the next 1 in row $i$ that comes after column $j$, and let $i'$ be the next 1 in column $j$ that comes after row $i$. We mark the 1 in position $A[i',j']$ as redundant. It should be clear that any 1 we mark as redundant is redundant.

Suppose some element $A[x,y]$ is redundant. Consider the "smallest" Γ that would be formed if $A[x,y] = 0$; by smallest, we mean that the area contained within this Γ is minimal. Let $A[i,j]$ be the upper left position of this Γ. Either column $y$ is not the next 1 entry in row $i$, or row $x$ is not the next 1 entry in column $j$, or we would mark $A[x,y]$ redundant. Without loss of generality, assume $j' \neq y$ is the next 1 entry after $j$ in row $i$. Then $A[x,j'] = 1$, or the rows $i, x$ and columns $j, j'$ induce a Γ. Then rows $i$ and $x$, columns $j'$ and $y$ induce a Γ if $A[x,y]$ is replaced by 0, contradicting the assumption that $A[i,j]$ is the upper left position of the smallest Γ formed. Therefore, every redundant 1 is marked by this algorithm. □

A similar procedure allows us to take the set of nonredundant 1's and reconstruct the entire matrix.

THEOREM 2. *Given the set of nonredundant* 1*'s in a* Γ*-free matrix* $A$, *we can reconstruct* $A$ *in* $O(n^2)$ *time.*

*Proof.* We search the matrix from row 1 to row $n$ and each row from column 1 to column $n$. For each 1 entry (whether this is originally redundant or nonredundant), we find the next 1 in the row and column and add a 1 to avoid a Γ if it is not there already.

Let $A[x,y]$ be the first matrix entry scanned that should be 1 which is left at 0 by our algorithm. All entries above and to the left are filled in correctly. Consider the smallest $\Gamma$ formed if $A[x,y] = 0$; let $u$ and $v$ be the row and column number of the upper left corner of this $\Gamma$.

There cannot be any 1 entries of the matrix between columns $v$ and $y$ in row $u$; otherwise assume that $w < y$ is the next column number after column $v$ that contains a 1 in row $u$. $A[x,w] = 1$, or the rows $u,x$ and columns $v,w$ form a $\Gamma$. Therefore, the rows $n,x$ and columns $w,y$ form a $\Gamma$, contradicting the assumption that the $\Gamma$ under consideration was smallest. For similar reasons, there cannot be any 1 entries between rows $u$ and $x$ in column $v$.

Since we did not change $A[x,y]$ to 1 when we examined position $A[u,v]$, and $A[u,y]$ and $A[x,v]$ are the next 1 entries of the matrix in row $u$ and column $v$, respectively, at least one of the entries $A[u,y]$ or $A[x,v]$ must have been changed from 0 to 1 after position $A[u,v]$ was examined. There are no 1 entries examined between $A[u,v]$ and $A[u,y]$, so $A[u,y]$ will not be changed from 0 to 1 after $A[u,v]$ is examined. For $A[x,v]$ to change from 0 to 1, there must be an entry examined between rows $u$ and $x$ that has a 1 in column $v$. There are no such entries, since row $x$ has the next 1 entry in column $v$ after row $u$. Therefore, when $A[u,v]$ is examined, the value of $A[x,y]$ is changed to 1. $\square$

We note that it is relatively easy to modify the algorithm presented in Theorem 2 to get an $O(m \log n)$ bound for reconstructing the matrix from the set of nonredundant 1's, which will be better for sparse matrices.

Therefore, a $\Gamma$-free matrix can be stored by giving the position of all nonredundant 1's. The space used to store the matrix in this fashion is $O(\log n) *$ the number of nonredundant 1's in the matrix. To determine whether this is a good form of storage, we would like to know the maximum number of nonredundant ones in a $\Gamma$-free matrix and the number of $\Gamma$-free matrices. If the size of the representation is at most a constant times the logarithm of the number of $\Gamma$-free matrices, we say the representation is optimal. We show that any $\Gamma$ -free matrix has $O(n \log n)$ nonredundant 1's and that the number of $\Gamma$-free matrices is $\Omega(2^{\Omega(n \log^2 n)})$, which means that storing the nonredundant 1's is space optimal.

## 4. Lower bounds.

In this section, we give lower bounds on the number of nonredundant 1's and the number of $\Gamma$-free matrices. We show that there are $\Gamma$-free matrices with $\Omega(n \log n)$ nonredundant 1's and that there are $\Omega(2^{\Omega(n \log^2 n)})\Gamma$-free matrices.

This shows, for example, that there is no constant bound on the boxicity of a chordal bipartite graph. Any boxicity $k$ graph can be stored using $O(k \log n)$ bits per vertex by storing the coordinates of the box corresponding to each vertex. Thus, the number of boxicity $k$ graphs is $O(2^{O(nk \log n)})$. Similar arguments show that chordal bipartite graphs must have arbitrarily high interval number. More sophisticated arguments can be used to show that graph classes such as circle intersection graphs and visibility graphs contain $O(2^{O(n \log n)})$ graphs with $n$ vertices and thus cannot contain all chordal bipartite graphs [2].

We construct a matrix with $\Omega(n \log n)$ nonredundant 1's recursively. Divide a matrix of size $n$ into four quadrants. Place the identity matrix in the upper left quadrant and a matrix with all values $= 1$ in the lower right quadrant. Repeat this construction recursively within the other two quadrants.

We first show that this procedure will never produce a $\Gamma$. The $\Gamma$ cannot contain any entry from the bottom right quadrant since these values are all 1. Therefore, the

$\Gamma$ cannot contain any entry from the upper left quadrant, since the next 1 in the row would be in the upper right, the next 1 in the column would be in the lower left, and the bottom right of such a $\Gamma$ is in the quadrant filled with 1's. No $\Gamma$ can span both the upper right and bottom left quadrant, unless some other quadrant is involved. Therefore, assuming the submatrices of size $n/2$ were constructed to be $\Gamma$-free, the matrix of size $n$ is also $\Gamma$-free.

We now show that the matrix constructed above has $\Omega(n \log n)$ nonredundant ones. Clearly, every 1 in the upper left quadrant is nonredundant. Any nonredundant 1 in the upper right and lower left quadrant taken individually is also nonredundant in the entire matrix. Let $NR(n)$ be the number of nonredundant 1's in an $n$ by $n$ matrix constructed as above. We can construct a recurrence relation $NR(n) \geq 2NR(n/2) + n/2, NR(1) = 1$, which solves to $\Theta(n \log n)$. Therefore, the number of nonredundant 1's in the matrix constructed above is $\Omega(n \log n)$.

We can get a lower bound on the number of $\Gamma$-free matrices by using a slight modification of the construction above. Suppose that we are given two arbitrary $\Gamma$-free matrices $M_1, M_2$ of size $n/2$. Place an arbitrary perfect matching in the upper left-hand quadrant of an $n$ by $n$ matrix, place $M_1$ and $M_2$ in the upper right and lower left quadrants, and fill the bottom right quadrant with 1's. For the same reasons given above, the matrix we construct must be $\Gamma$-free.

THEOREM 3. *The number of $\Gamma$-free matrices is* $\Omega(2^{\Omega(n \log^2 n)})$.

*Proof.* Let $NG(n)$ be the number of $\Gamma$-free $2n$ by $2n$ matrices. Since there are $n!$ perfect matchings in the upper left quadrant, $NG(n) \geq n! * NG(n/2) * NG(n/2)$. By expanding this recurrence relation, we get $NG(n) \geq n! * (n/2!)^2 * (n/4!)^4 * \cdots * (n/n)!^n$. We know that $n! > (n/2)^{n/2} = 2^{(n/2)\log n/2}$. Therefore, $NG(n) > 2^{(n/2)\log /2} * 2^{(n/2)\log n/4} * 2^{(n/2)\log n/8} * \cdots NG(n) > 2^{n/2(\log n/2 + \log n/4 + \log n/8 + \cdots)} = 2^{n/2((\log n-1) + \log n-2) + \cdots + 1))} = 2^{(n/2)(\log n)(\log n-1)/2}$, which is $\Omega(2^{\Omega(n \log^2 n)})$. □

We note that when this result is translated back to the number of chordal bipartite or strongly chordal graphs, it is not necessary to worry about different matrices producing isomorphic graphs. Even if we divide the number of $\Gamma$-free matrices by $n!$, (the number of possible graphs with $n$ vertices isomorphic to a graph $G$), the result is $\Omega(2^{\Omega(n \log^2 n)})$.

## 5. Upper bounds.

In this section, we show that the number of nonredundant 1's in a $\Gamma$-free matrix is $O(n \log n)$. This shows immediately that the number of $\Gamma$-free matrices (and thus strongly chordal and chordal bipartite graphs) is $O(2^{O(n \log^2 n)})$, since any such matrix can be stored by giving the positions of all nonredundant 1s using $O(n \log^2 n)$ bits.

We will read the matrix from row 1 to row $n$. Create a graph $G$ with vertices corresponding to columns of the matrix. We will read the matrix row by row, updating the graph $G$ as we go. Add an edge from column $i$ to column $j$ if $i < j$, and we see a row in which $i$ and $j$ both have value 1. Note that if there is an edge from $i$ to $j$, any future row that has a 1 in column $i$ must have a 1 in column $j$ as well, or we would get a $\Gamma$. When we read a new row, we cannot have nonredundant ones in two columns that are already connected by an edge in $G$. We also show that there cannot be a pair of nonredundant 1's in a row if there is already an edge between these rows in the transitive closure of $G$.

LEMMA 4. *After any row has been read, the transitive reduction of the transitive closure of $G$ has maximum outdegree* 1.

*Proof.* We note that $G$ is a directed acyclic graph, since every edge goes from a lower-numbered vertex to a higher-numbered vertex. Thus, the transitive reduction of $G$ is uniquely defined. Suppose $a$ has two outedges in the transitive reduction to $b$ and $c$, where $b < c$. Since these are edges of the transitive reduction, there must be a row for which $a$ and $b$ have a 1 and a row for which $a$ and $c$ have a 1. There cannot be a previous row for which $b$ and $c$ share a 1, or we would have $a \to b \to c$ in $G$. There are two possibilities, depending on which shared row comes first. These are shown below; both have Γ's, so they cannot occur if $G$ is Γ-free.

$$\begin{matrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{matrix} \qquad \begin{matrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{matrix}. \qquad \square$$

LEMMA 5. *If there is already an edge from $i$ to $j$ in the transitive closure of $G$, there cannot be nonredundant 1's in both column $i$ and column $j$ of a new row.*

*Proof.* Let $r$ be a row with nonredundant 1's in positions $i$ and $j$, and consider the path $i \to c_1 \to c_2 \to \cdots \to c_k \to j$ in the transitive reduction of the transitive closure of $G$. We claim that row $r$ has redundant 1's in columns $c_1, \dots, c_k, j$. Consider the first of these columns that does not have a redundant 1; call this $z_m$. Let $z_{m-1}$ be the predecessor of $z_m$ in the path from $i$ to $j$ in $G$. Since $z_{m-1}$ and $z_m$ share a 1 in some previous row, and $z_{m-1}$ is a 1 in row $r$ (either it is earlier than $z_m$ and thus a redundant 1 or it is $i$ and a nonredundant 1), $z_m$ must be a redundant 1. Therefore, the 1 in column $j$ is redundant, contradicting our original assumption. $\square$

Let $A_1$ be the first nonredundant 1 in each row and let $B_1$ be the first two nonredundant 1's in each column that is not in $A_1$. Let $A_i$ be the first nonredundant 1 for each row that is not in $A_1 \dots A_{i-1}$ or $B_1 \dots B_{i-1}$ and let $B_i$ be the first two nonredundant 1's for each column that are not in $A_1 \dots A_i$ or $B_1 \dots B_{i-1}$. We show that the set $A_{\log n+2}$ is empty. Since each $A_i$ and $B_i$ contain at most $2n$ nonredundant 1's, the number of nonredundant 1's in the matrix is therefore $O(n \log n)$.

We use the following matrix $M$ to give concrete examples of the sets defined here. Certain entries will be marked "$n$," meaning that these are positions of nonredundant ones. Other entries will be marked "$x$," meaning that these are either 0 entries or redundant 1's. Generating an actual example to illustrate the concepts would involve creating a very large matrix, so the matrix below does not correspond to a valid pattern of "$n$"s and "$x$"s; specifically, the entries $(3,4), (5,5)$, and $(6,5)$ would be redundant in any actual example containing the other values as nonredundant 1's.

$$\begin{matrix} n & x & n & n & x & x \\ x & n & x & x & n & x \\ n & x & x & n & x & n \\ x & n & x & n & x & x \\ x & x & x & n & n & x \\ x & n & x & x & n & x \end{matrix}.$$

In the matrix $M$ above, $A_1$ is the set of entries at positions $(1,1), (2,2), (3,1), (4,2), (5,4), (6,2)$. $B_1$ corresponds to $(1,3), (1,4), (3,4), (2,5), (5,5), (3,6)$, and $A_2$ contains the entries at positions $(4,4)$ and $(6,5)$.

THEOREM 6. *There are $O(n \log n)$ nonredundant 1's in a Γ-free matrix.*

*Proof.* Consider a nonredundant 1 that is in $A_{\log n+2}$. Call this nonredundant $1\, Y_{\log n+2}$. For each nonredundant 1 in $Y_i$, there are 2 nonredundant 1's that share this column but have lower row number in $B_{i-1}$, and for each of these members of $B_{i-1}$ there is a member of $A_{i-1}$ that has the same row number but an earlier column number. Let $X_{i-1}$ be the two members of $B_{i-1}$ in the same column as each member of

$Y_i$ and let $Y_{i-1}$ be the nonredundant 1's from $A_{i-1}$ in the same rows as these members of $X_{i-1}$.

As an example, consider the matrix $M$ described immediately before the statement of this theorem; let $Y_2$ be the entry at position $(6,5)$. Then $X_1$ corresponds to $(2,5)$ and $(5,5)$, and $Y_1$ is the set of entries at positions $(2,2)$ and $(5,4)$.

We show that no members of any $X_i$ can share the same row number and no members of $Y_i$ can share the same column number. This is certainly true for $i = \log n + 2$; consider the largest value of $i$ such that some pair of nonredundant 1's in $Y_i$ share the same column number or some pair of members of $X_i$ share the same row number.

For each element of $Y_i$ or $X_i$, we consider the parent of the element to be the nonredundant 1 from $Y_{i+1}$ that caused the element to be added to $Y_i$ or $X_i$. Any pair of elements in $Y_i$ or $X_i$ must have a least common ancestor, since the data structure formed is a tree. If we look at the first pair that share a row or column, the two elements are descendants of the two different elements of $B$ that correspond to children of their least common ancestor. We show that one of these (the 1 value with larger row number) is redundant, contradicting the assumption that it is in $B$.

Each time we add an element to $Y_i$, it is because there is a row that has a nonredundant 1 in both this column and the column of its parent. Let $p$ be a parent of $c$. Before the row corresponding to $p$ is read, the graph $G$ already has an edge from the column of $c$ to the column of $p$.

If two elements of $X_i$ share a common row, let $c_1$ be the column number of the earlier column that has a 1 in the shared row. If two elements of $Y_i$ share a common column, let $c_1$ be the column number that is shared by two different members of $Y_i$. Let $c_l$ be the column number of the least common ancestor of the pair that shares a row or column. Tracing back from the parents of each nonredundant 1 in this column to the least common ancestor, we know that $G$ has edges from $c_1$ to $d_1$ to $d_2$ to $\dots$ to $d_j$ to $c_l$, where the $d$ values are the column number of the parent, grandparent, and so forth of one of the nonredundant 1's that shares a common row/column and also edges from $c_1$ to $e_1$ to $e_2$ to $\dots$ to $e_j$ to $c_l$, where the $e$ values are the column numbers of the parent, grandparent, and so on, of the other nonredundant 1. Without loss of generality, assume that the row sharing $e_j$ and $c_l$ comes after the row sharing $d_j$ and $c_l$.

Let us look at the transitive reduction of the transitive closure of $G$ immediately before we read the row that shares nonredundant 1's in columns $e_j$ and $c_l$. Since the outdegree in the transitive reduction of $G$ is 1 and we already have a path from $c_1$ to $e_j$ and a path from $c_1$ to $c_l$, there is a path in $G$ from $e_j$ to $c_l$. This implies that there cannot be nonredundant 1's in both position $e_j$ and $c_l$ of this row, contradicting our assumption.

Since each nonredundant 1 in $Y_j$ generates two nonredundant 1's in $Y_{j+1}$ and no members of any $Y_i$ can share the same column number, $A_{\log n+2}$ must be empty (each member would generate $2^{\log n+1}$ members of $Y_{\log n+1}$, none of which can share a column). Therefore, the number of nonredundant 1's is $O(n \log n)$. $\square$

## 6. Open problems.

This paper gives asymptotic estimates of the number of $\Gamma$-free matrices and thus gives estimates on the number of strongly chordal and chordal bipartite graphs. We present a natural storage scheme for $\Gamma$-free matrices that is asymptotically space optimal.

There are other desirable aspects of a representation scheme for graphs besides space optimality. The issues we discuss here are closely related to local structure [12] or implicit representation of graph classes [9], when generalized to graph classes with more than $2^{O(n \log n)}$ graphs on $n$ vertices.

There are a number of drawbacks to the new storage scheme. Let us contrast this scheme with an interval representation for an interval graph. In an interval graph, we can store for each vertex the positions of the two endpoints in an interval representation, and this is asymptotically space optimal. In the case of interval graphs, we also have the property that the minimum possible amount of information is stored at each vertex, and we can test whether there is an edge between two vertices in constant time using only the information stored at that vertex.

We would like to achieve similar results for graph classes that are defined by Γ-free matrices. First, is there a storage method using $O(n \log^2 n)$ space that allows us to determine whether two vertices are adjacent efficiently? Second, can we make this representation local, in the sense that the adjacency of two vertices can be tested using only $O(\log^2 n)$ information that is stored at each of the two vertices? Chordal bipartite graphs form a hereditary class of graphs; there is an open question [9] as to whether every hereditary graph class with $2^{O(n \log n)}$ graphs on $n$ vertices has a space optimal representation that allows this form of adjacency testing. As far as the author knows, the more general question of whether every hereditary class of graphs with $2^{nf(n)}$ members has a representation with $O(f(n))$ bits per vertex that allows local adjacency testing is also open.

## REFERENCES

[1] I. Adler, A. J. Hoffman, and R. Shamir, *Monge and feasibility sequences in general glow problems*, preprint.

[2] N. Alon and E. R. Scheinerman, *Degrees of freedom versus dimension for containment orders*, Order, 5 (1988), pp. 11–16.

[3] R. P. Anstee and M. Farber, *Characterizations of totally balanced matrices*, J. Algorithms, 5 (1984), pp. 215–230.

[4] A. D'Atri and M. Moscarini, *On hypergraph acyclicity and graph chordality*, Inform. Process. Lett. 29 (1988), pp. 271–274.

[5] R. Fagin, *Degree of acyclicity for hypergraphs and relational database schemes,* J. Assoc. Comput. Mach. 30 (1983), pp. 514–550.

[6] M. Farber, *Characterizations of strongly chordal graphs*, Discrete Math., 43 (1983), pp. 173–189.

[7] ———, *Domination, independent domination, and duality in strongly chordal graphs*, Discrete Appl. Math., 7 (1984), pp. 115–130.

[8] A. J. Hoffman, A. W. J. Kolen, and M. Sakarovitch, *Totally-balanced and greedy matrices*, SIAM J. Algebr. Discrete Meth., 6 (1985), pp. 721–730.

[9] S. Kannan, M. Naor, and S. Rudich, *Implicit representation of graphs*, SIAM J. Discrete Math., 5 (1992), pp. 596–603.

[10] L. Lovász, *Combinatorial problems and exercises*, North-Holland, Amsterdam, 1979.

[11] A. Lubiw, *Doubly Lexical Orderings of Matrices*, SIAM J. Comput., 16 (1987), pp. 854–879.

[12] J. H. Muller, *Local structure in graphs*, Ph.D. thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA, 1988.

[13] R. Paige and R. E. Tarjan, *Three partition refinement algorithms*, SIAM J. Comput., 16 (1987), pp. 973–989.

[14] J. P. Spinrad, *Doubly lexical ordering of dense* $0-1$ *matrices*, Inform. Process. Lett., 45 (1993), pp. 229–235.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 258–280, May 1995

# SLICEABLE FLOORPLANNING BY GRAPH DUALIZATION *

GARY K. H. YEAP† AND MAJID SARRAFZADEH‡

**Abstract.** Previous algorithms on rectangular dual graph floorplanning generate general floorplans which include the class of nonsliceable floorplans. We examine the framework of generating sliceable floorplans using the rectangular dual graph approach and present an algorithm that generates a sliceable floorplan if the input graph satisfies certain sufficient conditions. For general input, the algorithm is still able to generate sliceable floorplans by introducing pseudomodules where the areas occupied by the pseudomodules are used for wiring. For an $n$-vertex adjacency graph, the algorithm generates a sliceable floorplan in $O(n \log n + hn)$ time where $h$ is the height of the sliceable floorplan tree.

**Key words.** very large scale integration (VLSI) floorplanning, sliceable floorplans, planar graphs, graph dualization

**AMS subject classifications.** 05C85, 68Q20, 68Q35, 68Q25, 68R10

## 1. Introduction.

The rectangular dual graph approach to very large scale integration (VLSI) floorplanning, introduced in [1], is based on the idea of preserving the planar adjacency of input modules. The adjacency requirements of a floorplan are specified by an adjacency graph. Each vertex represents a rectangular partition (module) of the chip and each edge represents an adjacency requirement between two modules. A floorplan and its adjacency graph exhibit a certain duality relation. To ensure the existence of a floorplan, the adjacency graph must be restricted to a special class of planar triangulated graphs. We call it a *rectangular admissible graph*. The characterization of the graph was introduced by Kozminski and Kinnen [1], [2]. They also introduced algorithms to generate floorplans based on this dual graph approach. Bhasker and Sahni [4] improved the time complexity by providing an algorithm which finds a floorplan in linear time if one exists. Sun and Sarrafzadeh [5] and Yeap and Sarrafzadeh [6] generalized the approach to incorporate modules with shapes more complicated than rectangles.

Previous approaches yielded general floorplans which include the class of nonsliceable floorplans. We focus our interest on generating the class of sliceable floorplans. A sliceable floorplan has several attractive features. Since it is inherently a tree structure (as opposed to a graph structure), it facilitates later phases of layout processing [9]. For example, a version of the floorplan sizing algorithm has been shown to be NP-complete for general floorplans but is optimally solvable for sliceable floorplans [10].

For some classes of rectangular admissible graphs, only nonsliceable floorplans exist. For example, the graph in Fig. 1 has a rectangular floorplan as shown. By inspection, one can verify that no sliceable floorplans exist for the graph. The exact characterization of this class of graphs is still unknown. Finding a sliceable floorplan for a rectangular admissible graph, if one exists, is an open problem. One major condition on the rectangular admissible graph is that it contains no complex cycle

* Received by the editors November 13, 1991; accepted for publication (in revised form) April 19, 1994. This work was supported in part by National Science Foundation grant MIP-8921540.

† Motorola, 2100 East Elliot Road, MD EL510, Tempe, Arizona 85284.

‡ Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois 60208.

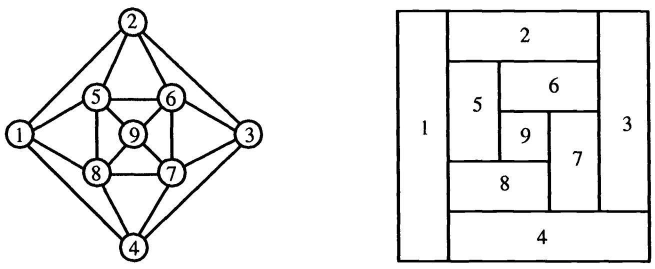


FIG. 1. *A nonsliceable rectangular admissible graph.*

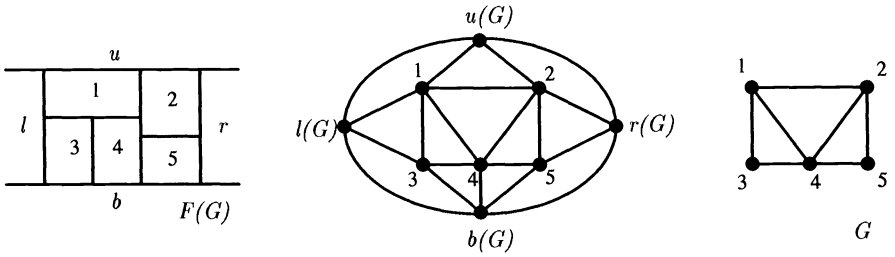


FIG. 2. *A floorplan $F(G)$, extended dual $EPTG(G)$, and dual $G$.*

of length 3. We will prove that with the additional constraint that the input graph contains no complex cycle of length 4, sliceable floorplans always exist.

When an adjacency graph does not admit a sliceable floorplan, one could introduce pseudovertices to the input graph. We add a pseudovertex on an edge when adjacency cannot be satisfied. Pseudovertices represent rectangular areas created purely to satisfy the adjacency requirements. The areas consist of only routing wires and contain no devices. The areas consumed by the pseudovertices depend on the density of the wires carried by the edges. If the wiring density of an edge is zero, adding pseudovertices will not waste any chip space. Zero edge density occurs when the original adjacency of modules is not a fully triangulated graph. In our algorithm, heuristic measures can be imposed to minimize the areas consumed by pseudovertices.

Section 2 of this paper defines the formal terminologies. Section 3 discusses a formal framework for sliceable rectangular floorplans. The sliceability requirements on the primal floorplan are translated to the dual graph, and the corresponding properties in the dual are examined. Section 4 presents an algorithm for generating sliceable floorplans based on the dual graph approach. Section 5 presents some application aspects of the floorplanning algorithm.

## 2. Preliminaries.

A *rectangular floorplan* (RFP) $F$ is a plane graph where

(a) each edge is either a vertical or a horizontal line segment,

(b) each face is a rectangle,

(c) each vertex has degree 3,

(d) the boundary of $F$ is a rectangle.

The restriction of a floorplan to degree 3 vertices does not lose generality. A floorplan with degree 4 vertices can be transformed into one with only degree 3 vertices by a minor modification [1]. We assume that $F$ is bounded by four infinite faces $r, u, l, b$ as shown in Fig. 2. Given $F$, the *extended dual* $EPTG(G)$ of $F$ can be constructed as

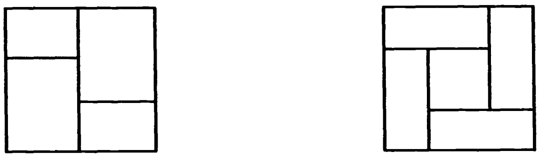


FIG. 3. *A sliceable floorplan (left) and a nonsliceable floorplan (right).*

follows:

(a) Each face of $F$ corresponds to a vertex of $G$.

(b) There is an edge between two vertices of $G$ if the two corresponding faces in $F$ are adjacent.

If the vertices $r(G), u(G), l(G), b(G)$ and their incident edges are deleted from $EPTG(G)$, the resulting graph $G$ is called the *dual* of $F$. The rectangular dual approach to floorplanning consists of obtaining a floorplan $F$ from a given dual graph $G$.

Two different $F$'s may have an identical dual $G$. A vertex in $G$ which is adjacent to more than one vertex of $\{r(G), u(G), l(G), b(G)\}$ is called a *corner vertex*. The four corner vertices in $G$ correspond to the four corner faces of $F$. The corner vertices in $G$ need not be distinct if the corner faces of $F$ are not distinct. An $F$ with dual $G$ is denoted by $F(G)$. If $G$ is the dual of some floorplan $F$, we say that $G$ *admits* a floorplan $F$.

Given $G$, unless otherwise stated we assume that the four corner vertices of $G$ have been fixed, thus $EPTG(G)$ is known. We denote the corner vertices as $\mathrm{NW}(G), \mathrm{NE}(G), \mathrm{SE}(G)$, and $\mathrm{SW}(G)$ (northwest, northeast, southeast, and southwest). The selection of corner vertices is not a trivial process. If the corners are not selected carefully, the extended dual $EPTG(G)$ may not be rectangular admissible. Basically, the corners should be selected so that the addition of vertices $r(G), u(G), l(G), b(G)$ does not result in complex triangles (to be defined). The external edges (edges incident to the infinite face) of $G$ are divided into four mutually disjoint subsets (possibly empty), denoted by $\mathrm{TOP_e}(G), \mathrm{LEFT_e}(G), \mathrm{BOTTOM_e}(G)$, and $\mathrm{RIGHT_e}(G)$. The corresponding vertices (incident to the infinite face) are denoted by $\mathrm{TOP_v}(G), \mathrm{LEFT_v}(G)$, $\mathrm{BOTTOM_v}(G)$, and $\mathrm{RIGHT_v}(G)$. If $G$ is a dual of $F$, all internal faces of $EPTG(G)$ are triangles because vertices of $F$ have degree 3. This is also true for $G$ since it is a subgraph of $EPTG(G)$. For example, in Fig. 2,

$$\begin{aligned}
&\mathrm{NW}(G) = 1, \mathrm{NE}(G) = 2, \mathrm{SE}(G) = 5, \mathrm{SW}(G) = 3;\\
&\mathrm{TOP_e}(G) = \{(1,2)\}, \mathrm{LEFT_e}(G) = \{(1,3)\},\\
&\mathrm{BOTTOM_e}(G) = \{(3,4),(4,5)\}, \mathrm{RIGHT_e}(G) = \{(2,5)\},\\
&\mathrm{TOP_v}(G) = \{1,2\}, \mathrm{LEFT_v}(G) = \{1,3\},\\
&\mathrm{BOTTOM_v}(G) = \{3,4,5\}, \mathrm{RIGHT_v}(G) = \{2,5\}.
\end{aligned}$$

An RFP $F$ is called *sliceable* if it is a rectangle or can be decomposed into two nonempty RFPs, $F_1, F_2$ by a vertical or horizontal line such that $F_1$ and $F_2$ are both sliceable. Examples of sliceable and nonsliceable floorplans are shown in Fig. 3. A sliceable RFP can be represented by a tree.

A *path* is an ordered set of adjacent vertices. A path can also be equivalently denoted by the set of edges in the path. If the two end vertices of a path are identical, it is called a *cycle*. A *complex cycle* (of length $n$) is a cycle $C_n$ of $n$ edges (of a plane graph) where there is at least one vertex in the finite region bounded by $C_n$. A complex cycle with vertices $\{v_1, \ldots, v_n\}$ is denoted by $C_n(v_1, \ldots, v_n)$. In particular, we are interested in $C_3$ and $C_4$.

A *chord free path* (CFP) in $G$ is a path $P = (v_1, \ldots, v_n)$, where for all $i \neq j$,

(a) $v_i \neq v_j$, and

(b) if $(v_i, v_j) \in G$ then $|i - j| = 1$.

If a path $Q = (q_1, \ldots, q_n)$ is not a CFP in $G$, then either $q_i = q_j$ for some $i \neq j$ or there exist some edges $(q_i, q_j) \in G$, where $|i - j| \geq 2$. In the latter case, the edges $(q_i, q_j)$ are called *chords* of path $Q$.

A *vertical slice* is an ordered set of edges $E_s = (e_1, \ldots, e_n)$ in $G$, where

(a) $e_1 \in \mathrm{TOP_e}(G), e_n \in \mathrm{BOTTOM_e}(G)$,

(b) for $1 < i < n, e_i \notin \mathrm{TOP_e}(G) \cup \mathrm{BOTTOM_e}(G) \cup \mathrm{LEFT_e}(G) \cup \mathrm{RIGHT_e}(G)$;

(c) $G$ is decomposed into exactly two nonempty components $G_l$ (left subgraph) and $G_r$ (right subgraph) when $E_s$ is removed;

(d) for any $e \in E_s$, adding $e$ causes $G_l$ and $G_r$ to be connected.

A *horizontal slice* is defined similarly where $e_1 \in \mathrm{LEFT_e}(G)$ and $e_n \in \mathrm{RIGHT_e}(G)$, and $G$ is decomposed into $G_u$ (upper subgraph) and $G_b$ (lower subgraph). A *slice* is either a vertical or horizontal slice. Let $e_1 = (v_1, w_1) \in \mathrm{TOP_e}(G)$ and $e_n = (v_n, w_n) \in \mathrm{BOTTOM_e}(G)$ be edges of a vertical slice with $v_1, v_n \in G_l$ and $w_1, w_n \in G_r$. When $G$ is decomposed into $G_l$ and $G_r$ by the vertical slice, the four corner vertices of $G_l$ and $G_r$ are well defined:

$$\mathrm{NW}(G_l) = \mathrm{NW}(G), \quad \mathrm{NE}(G_l) = v_1, \quad \mathrm{SE}(G_l) = v_n, \quad \mathrm{SW}(G_l) = \mathrm{SW}(G);$$
$$\mathrm{NW}(G_r) = w_1, \quad \mathrm{NE}(G_r) = \mathrm{NE}(G), \quad \mathrm{SE}(G_r) = \mathrm{SE}(G), \quad \mathrm{SW}(G_r) = w_n.$$

*EPTG*$(G_l)$ can be constructed by adding $r(G_l)$ and edges $\{(r(G_l), v_1), \ldots, (r(G_l), v_n)\}$. *EPTG*$(G_r)$ can be constructed similarly by adding $l(G_r)$. Note that we may have $v_1 = v_n$ and/or $w_1 = w_n$. Figure 4 shows an example of decomposition by a vertical slice.

A vertical slice $E_s$ on *EPTG*$(G)$ defines a *left boundary path* $P_l(E_s) = (u(G), v_1, \ldots, v_n, b(G)), v_i \in \mathrm{RIGHT_v}(G_l)$, and a *right boundary path* $P_r(E_s) = (u(G), w_1, \ldots, w_m, b(G)), w_i \in \mathrm{LEFT_v}(G_r)$ (see Fig. 4). Note that $(v_1, \ldots, v_n) \cap (w_1, \ldots, w_m) = \phi$ and $(v_1, \ldots, v_n) \cup (w_1, \ldots, w_m)$ is the set of vertices of $E_s$. Boundary paths of a horizontal slice are defined similarly. When a vertical slice is specified, the left and right boundary paths are well defined; conversely, when the left or right boundary paths are specified, vertical slice can be defined. Given a right boundary path $P = (v_1, \ldots, v_n)$, where $v_1 \in \mathrm{TOP_v}(G)$ and $v_n \in \mathrm{BOTTOM_v}(G)$, we can construct a slice $E_s$ by taking all edges incident to vertices of $P$ and on the left (but excluding edges) of $P$. If such $E_s$ satisfies the condition of a slice, we call $E_s$ the *left-induced slice* of path $P$. Similarly, we can define the *right-induced slice* of $P$. If an $E_s$ so constructed is not a slice, the left- (right-) induced slice of $P$ is undefined.

A *vertical slice* in $F$ (not in $G$) is a set of edges $S = (e_1, \ldots, e_n)$ of a nonboundary path in $F$, where

(a) all $e_i$'s are on a common vertical cut-line;

(b) $e_1$ is incident to the top boundary of $F(G)$; and

(c) $e_n$ is incident to the bottom boundary of $F(G)$.

*Horizontal slice* and *slice* on $F$ are defined similarly. In a sliceable RFP, at least one slice exists.

**3. Properties of sliceable floorplans and their duals.** Many properties of a floorplan are reflected on its corresponding dual and vice versa. Some of the properties have been investigated in earlier literature [1]–[6]. Lemma 1 below was proved in [1].

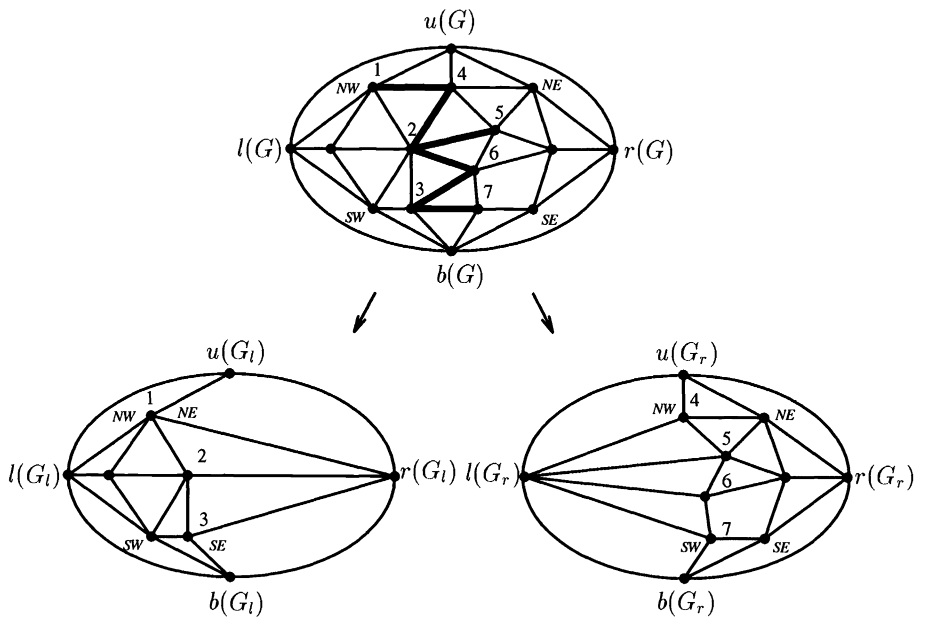


FIG. 4. *Decomposition by a vertical slice.* $E_s = \{(1,4),(2,4),(2,5),(2,6),(3,6),(3,7)\}, P_l(E_s) = (u(G),1,2,3,b(G)), P_r(E_s) = (u(G),4,5,6,7,b(G))$.

LEMMA 1. *Let $G$ be a planar triangulated graph with corner vertices fixed. $G$ admits an RFP if and only if $EPTG(G)$ contains no $C_3$ (complex cycle of length 3).*

LEMMA 2. *Let $G$ be a planar triangulated graph which admits an RFP. Let $E_s$ be a vertical slice on $G$, and $G_l, G_r$ be two subgraphs decomposed by $E_s$. If both boundary paths $P_l(E_s), P_r(E_s)$ are CFPs, then both $G_l$ and $G_r$ admit RFPs.*

*Proof.* Let $P_l(E_s) = (p_1, \ldots, p_n)$. Since $G$ admits an RFP, by Lemma 1 $EPTG(G)$ contains no $C_3$. Consider the $EPTG(G_l)$. Suppose for the sake of contradiction that $G_l$ does not admit an RFP. By Lemma 1, there is a $C_3(a,b,c)$ in $EPTG(G_l)$. Since $EPTG(G)$ contains no $C_3$, at least one of the edges of $C_3(a,b,c)$ is not in $EPTG(G)$. The only edges that exist in $EPTG(G_l)$ but not in $EPTG(G)$ are incident to $r(G_l)$ (refer to Fig. 4). Since $P_l(E_s)$ is a CFP, any edge incident to $r(G_l)$ cannot form a $C_3$. Thus, $EPTG(G_l)$ contains no $C_3$, a contradiction. By Lemma 1, $EPTG(C_l)$ admits an RFP. Similarly, $EPTG(G_r)$ also admits an RFP. A similar lemma applies to horizontal slices. □

A slice $E_s$ that satisfies the conditions that both boundary paths $P_1(E_s)$ and $P_2(E_s)$ are CFPs is called a *proper slice.* When a left floorplan $F_l$ and a right floorplan $F_r$ are placed side by side to form a floorplan, we denote it as $F_l|E_s|F_r$, where $E_s$ is the corresponding slice of the floorplan.

LEMMA 3. *Let $E_s$ be a proper vertical slice on $G$ which admits an RFP. Let $G_l$ and $G_r$ be the left and right subgraphs decomposed by $E_s$. $F(G_l)|E_s|F(G_r)$ has dual $G$.*

*Proof.* The dual of $F(G_l)|E_s|F(G_r)$ can be constructed by embedding $G_l$ on the left and $G_r$ on the right with $E_s$ connecting them. The resulting graph is exactly $G$. □

LEMMA 4. *Let $F(G)$ be a sliceable RFP. Let $S = (e_1, \ldots, e_n)$ be a slice of $F(G)$. The corresponding set of dual edges $E_s = (e'_1, \ldots, e'_n)$ in $G$ is a proper slice.*

*Proof.* Since $S$ consists of only vertical (horizontal) line segments, it is a union of

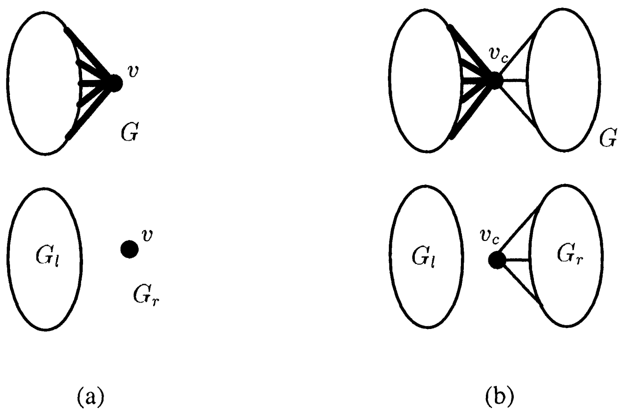


FIG. 5. *Slicing G for special cases.* (a) *nondistinct corner vertex v.* (b) *G contains a cut vertex $v_c$.*

two floorplans $F_l$ and $F_r$ with $S$ as the common boundary. Let $G_l$ and $G_r$ be the dual of $F_l$ and $F_r$. By Lemma 1, $EPTG(G_l)$ and $EPTG(G_r)$ contain no $C_3$. Therefore, there are no chords on the left and right boundary paths of $E_s$. □

THEOREM 1. *Let $G$ be a planar triangulated graph which admits an RFP. If $G$ contains no $C_4$ (complex cycle of length 4), then it admits a sliceable RFP.*

*Proof.* We prove the theorem by showing that, given an arbitrary planar triangulated graph $G$, where $EPTG(G)$ contains no $C_3$ and $G$ contains no $C_4$, there exists at least one proper slice $E_s$.

If $G$ contains only one vertex, the proof is trivial. Since $EPTG(G)$ is given, the corner vertices are fixed. If there is a vertex $v \in G$ which occupies exactly two corners (no vertex can occupy exactly 3 corners), simply let $E_s$ be the set of edges incident to $v$. Since $G$ contains no $C_3, E_s$ is a proper slice. If $G$ contains a cut vertex $v_c$, let $E_s$ be the set of edges incident to the left (or right) of $v_c$. The graphs in these cases are shown in Fig. 5. Thick lines show proper slices $E_s$.

We are only left with the cases where the four corners of $G$ are distinct and $G$ contains no cut vertices. We label the vertices of $EPTG(G)$ using the following procedure:

(a) Delete $u(G), l(G), b(G)$ and their incident edges.

(b) Perform a breath first search (BFS) traversal on the remaining graph with $r(G)$ as the root. Label a vertex $v$ at level $i$ in a BFS tree with a superscript $i$, i.e., $v^i$.

We "redraw" the graph such that vertices on the same BFS level are aligned vertically. Figure 6 shows such a BFS drawing. Solid lines indicate edges that must exist in $EPTG(G)$. Dashed lines indicate that there may be zero or more edges/vertices. For the purpose of discussion, $u(G)$ and $b(G)$ are special vertices which can be at levels 1 and 2. In fact, we need not redraw the whole graph because we are only interested in the vertices and edges up to level 2. We will show that we can construct a proper vertical slice by considering the BFS drawing of the $EPTG(G)$.

Let $E(i,j)$ be the set of edges between levels $i$ and $j$ and let the vertices at level $i$ be labeled $v_1^i, \ldots, v_{n_i}^i$ from the top (see Fig. 6). Consider the vertical slice

$$S_0 = E(1,2) - \{(u(G),x) | x \in V\} - \{(b(G),y) | y \in V\} \tag{1}$$

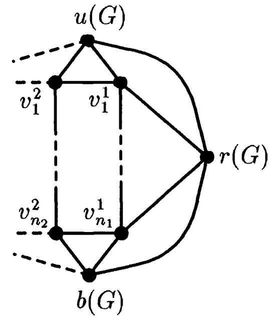


FIG. 6. *BFS drawing of an extended graph $EPTG(G)$.*

and its corresponding boundary paths

$$P_r(S_0) = (v_0^1 = u(G), v_1^1, \dots, v_{n_1}^1, v_{n_1+1}^1 = b(G)),$$
$$P_l(S_0) = (v_0^2 = u(G), v_1^2, \dots, v_{n_2}^2, v_{n_2+1}^2 = b(G)).$$

Since the corner vertices are distinct, $n_1, n_2 \geq 2$. An example of $EPTG(G)$ and $S_0$ is given in Appendix I.

If $S_0$ is a proper slice, we are done. If $S_0$ is not a proper slice, the right boundary path $P_r(S_0)$ is still a CFP because any edge $e = (v_i^1, v_j^1) \in E(1,1)$ is not a chord of $P_r(S_0)$; otherwise there exists a complex triangle $C_3(v_i^1, v_j^1, r(G))$. Thus the chords must be in $P_l(S_0)$. Because of the way $S_0$ is selected, the chords must appear on the left of $P_l(S_0)$.

A chord $(v_i^2, v_j^2), i < j$, where there are no other chords $(v_k^2, v_l^2)$, where $k \leq i < j \leq l$, is called a *maximal chord.* Let $E_c = \{e_1, \dots, e_c\}$ be the set of maximal chords of $P_l(S_0)$. For any chord $(v_i^2, v_j^2) \in E_c$, there is no chord $(v_k^2, v_l^2)$, where $k \in [i+1, j-1]$ and $l \notin [i,j]$ (i.e., chords are pairwise "noncrossing"). This observation allows us to perform local modifications in the neighborhood of a maximal chord $(v_i^2, v_j^2)$ to construct a proper slice.

The neighborhood of a maximal chord $e_p = (v_i^2, v_j^2) \in E_c, i < j$ can be depicted by the general configuration in Fig. 7(a). $v_m$ is the vertex to the right of $e_p$ which contributes to the triangular face $(v_i^2, v_j^2, v_m)$. $v_m$ must be located at level 2 or above. Figures 7(b) and (c) show some special cases of the general configuration when $v_m$ is at level 2. Again, dashed lines indicate zero or more edges/vertices.

For each maximal chord $e_p = (v_i^2, v_j^2) \in E_c, i < j$, let

$E_{tz}(a,b,c,d) \subseteq E(1,2)$ be the set of edges bounded by the trapezoid region $(v_a^2, v_b^2, v_c^1, v_d^1)$, including the edges $(v_a^2, v_d^1), (v_b^2, v_c^1)$;

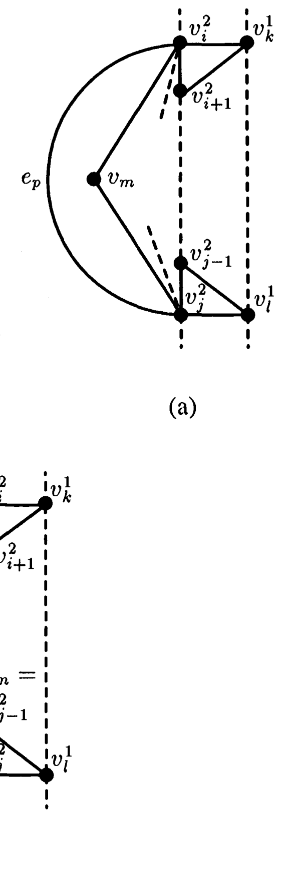
$E_p(i) \subset E(2,2)$ be the set of edges from (including) $(v_i^2, v_{i+1}^2)$ to $e_p = (v_i^2, v_j^2)$ (not including) if we scan the edges incident to vertex $v_i^2$ clockwise; if $i = 0$, i.e., edge $e_p$ is incident to $u(G), E_p(i=0) = \phi$;

$E_p(j) \subset E(2,2)$ be the set of edges from (not including) $e_p = (v_j^2, v_i^2)$ to $(v_j^2, v_{j-1}^2)$ (including) if we scan the edges incident to vertex $v_j^2$ clockwise; if $j = n_2 + 1$, i.e., edge $e_p$ is incident to $b(G), E_p(j = n_2 + 1) = \phi$.

The regions of the edge sets $E_{tz}(i,j,l,k), E_p(i), E_p(j)$ of a maximal chord $e_p = (v_i^2, v_j^2)$ are depicted by the shaded regions in Fig. 8.

For each maximal chord $e_p = (v_i^2, v_j^2) \in E_c, p = 1, 2, \dots, c$, we identify the two vertices $v_k^1$ and $v_l^1$ at level 1 (see Fig. 7) and obtain a new set $S_p$ by

$$S_p = S_{p-1} - E_{tz}(i+1, j-1, l, k) + E_p(i) + E_p(j), \qquad p = 1, 2, \dots, c,$$



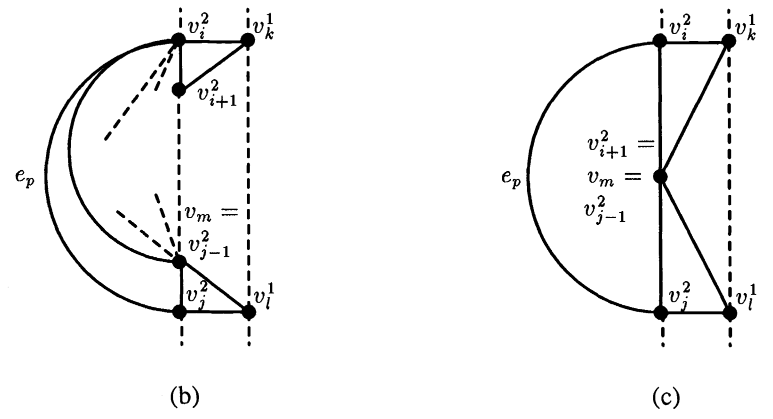


FIG. 7. *Configurations of a maximal chord:* (a) *general configuration;* (b) *configuration* $v_m = v_{j-1}^2$; (c) *configuration when* $v_{i+1}^2 = v_m = v_{j-1}^2$.

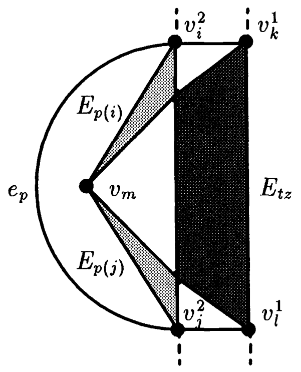


FIG. 8. *The regions of the sets* $E_{tz}(i, j, l, k), E_p(i)$, *and* $E_p(j)$ *of a maximal chord* $e_p$.

where $S_0$ is defined by equation (1). The above operation is called *bypassing a maximal chord* corresponding to $e_p$. An example of bypassing operation is shown in Fig. 9. The original slice (in thick lines) is shown on the left and the bypassed slice is shown on the right with maximal chord $e_p = (5, 8)$. After the bypassing operation is performed

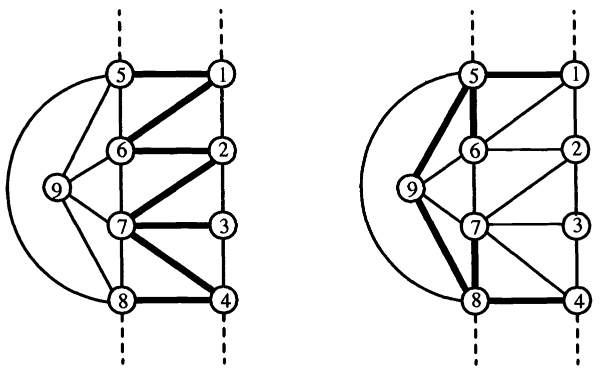


FIG. 9. *Bypassing a maximal chord $c_p$: original slice (left) and bypassed slice (right).*

on all maximal chords of $E_c$, we claim that the set $E_s = S_c$ is a proper slice ($c = |E_c|$). This implies that there exists at least one proper slice in $G$. Since the subgraphs $G_l$ and $G_r$ also contain no $C_3$ or $C_4$, the theorem applies recursively and a sliceable RFP exits.

To prove the claim, we observe that because of the bypassing operation for constructing $E_s$, the left boundary path $P_l(E_s)$ of $EPTG(G)$ is a CFP. It remains to show that $P_r(E_s)$ is also a CFP. There are two types of vertices in $P_r(E_s)$. Type 1 vertices are vertices at level 1 which are also found in the original $P_r(S_0)$ before bypassing. The other vertices are called type 2 vertices. They are added when a chord is bypassed, and by definition all type 2 vertices are not located at level 1 (see Fig. 8). If a type 2 vertex is added when a maximal chord $e_p = (v_i^2, v_j^2)$ is bypassed, we call it a type $2.p$ vertex. Each type $2.p$ vertex is adjacent to either $v_i^2$ or $v_j^2$. Suppose for the sake of contradiction that $P_r(E_s)$ contains a chord $(x, y)$. The chord must be located on the right of $P_r(E_s)$. Consider the following cases.

*Case* 1 ($x, y$ are both type 1 vertices). Since $x$ and $y$ are level 1 vertices, $x, y \in P_r(S_0)$. However, $(x, y)$ cannot be a chord of $P_r(S_0)$ since $P_r(S_0)$ is a CFP. Therefore, $(x, y)$ must be an edge of $P_r(S_0)$. It became a chord when we bypassed $e_p = (v_i^2, v_j^2)$ for some $p$. Referring to Fig. 7, we can see that this condition implies the existence of $C_4(x, y, v_i^2, v_j^2)$.

*Case* 2 ($x, y$ are both type 2 vertices). From the construction of $E_s$, both vertices must belong to identical subtype $2.p$. Suppose both vertices $x, y$ are adjacent to $v_i^2$. This implies the existence of $C_3(x, y, v_i^2)$ (the chord $(x, y)$ is on the right of $P_r(E_s)$). The same argument applies if both vertices are adjacent to $v_j^2$. If $x$ is adjacent to $v_i^2$ and $y$ is adjacent to $v_j^2$, it implies the existence of $C_4(x, y, v_j^2, v_i^2)$.

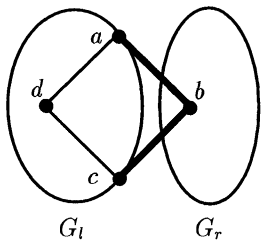
*Case* 3 ($x$ is type 1 and $y$ is type $2.p$). Without loss of generality, let $v_i^2$ be the vertex adjacent to $y$. From Fig. 7, a chord $(x, y)$ exists only when $x$ is $v_k^1$ or $v_l^1$. If $x = v_k^1$, there exists $C_3(x = v_k^1, y, v_i^2)$. If $x = v_l^1$, there exists $C_4(x = v_l^1, y, v_i^2, v_j^2)$.

All possible cases above lead to contradictions by implying the existence of $C_3$ or $C_4$. Thus $P_r(E_s)$ is a CFP. □

An example illustrating an application of Theorem 1 is given in Appendix I. The converse of Theorem 1 is certainly not true, that is, there exist sliceable floorplans whose duals contain some $C_4$, for example, a $C_4$ with one vertex inside.

To help the development of a slicing algorithm, we observe the following lemma.

LEMMA 5. *The intersection of a slice $E_s$ and any $C_4$ is either empty or contains exactly two edges.*



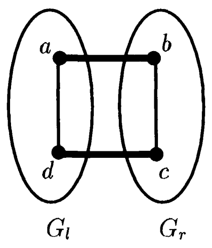


FIG. 10. *The two cases of slicing a $C_4$: corner-sliced (left) and center-sliced (right).*

*Proof.* The proof of the lemma follows immediately from the definition of $E_s$. Let $I$ be the set of edges from the intersection of $E_s$ and $C_4$. $|I|$ must be no greater than 4 since $|C_4| = 4$. Let $G_l$ and $G_r$ be the left and right subgraphs sliced by $E_s$. Suppose $|I| = 1$. Since the vertices of $C_4$ are still connected, they must belong to the same connected component. Thus we can add the edge in $I$ without increasing the number of connected components in $G - E_s$. This contradicts the fact that $E_s$ is a slice. If $|I| = 3$ and $G - E_s$ contain exactly two components, there must be at least one edge in $I$ that can be added to $G - E_s$ without reducing the number of connected components, a contradiction. If $|I| = 4$, there are at least three connected components in $G - E_s$, i.e., $G_l, G_r$, and the vertices in the area bounded by the $C_4$; this is another contradiction. □

## 4. An algorithm for generating sliceable floorplans.

In this section we discuss some issues of slicing a planar triangulated graph (PTG) to generate a sliceable floorplan. The slicing operation is complicated by the presence of $C_4$ in a PTG. We first examine the properties of slicing a $C_4$. We then present an algorithm that generates a sliceable floorplan from a PTG. If the PTG contains no $C_4$, the algorithm always generates a sliceable floorplan. Otherwise, pseudovertices may be added to maintain sliceability in the floorplan. Special considerations are made when slicing a PTG containing $C_4$.

### 4.1. Slicing a $C_4$.

Consider a $C_4(a, b, c, d)$ and a slice $E_s$. Let $E_i$ be the set of edges in the intersection of $C_4(a, b, c, d)$ and $E_s$. If $E_i$ is not empty, then by Lemma 5, $E_i$ contains exactly two edges. Without loss of generality, we can classify the two edges of $E_i$ as follows:

(a) *corner-slice*: $E_i = \{(a, b), (b, c)\}$;

(b) *center-slice*: $E_i = \{(a, b), (c, d)\}$.

The two cases are shown in Fig. 10. When a $C_4$ is corner-sliced, $b \in G_r$ (without loss of generality) and $a, c, d \in G_l$, where $a, c$ are vertices of the slice $E_s$. Consider the vertices of $E_s$ which are adjacent to vertex $b$. Let $P_s = \{a = v_1, \ldots, v_n = c\}$ be the vertices. The existence of a chord $(v_i, v_j)$ in $P_s$ would imply a complex triangle $C_3(v_i, v_j, b)$. Thus no chords exist in $P_s$. Furthermore, edges $(c, d)$ and $(d, a)$ can never be chords of $P_l(E_s)$ because $d \notin P_l(E_s)$.

For a center-sliced $C_4, a, d \in G_l, b, c \in G_r$, and $a, b, c, d \in E_s$. Since the $C_4$ contains some vertices, either $(a, d)$ is a chord of $P_l(E_s)$ or $(b, c)$ is a chord of $P_r(E_s)$ (or both). Bypassing operation on the chords does not help because the condition holds regardless of the slice $E_s$. Thus, when a $C_4$ is center-sliced, we have to modify the input graph $G$ to produce a sliceable floorplan.

From the above observations, we conclude that we need to corner-slice as many $C_4$'s as possible to avoid chords on boundary paths of a slice. The number of $C_4$'s in a PTG may exceed $O(n)$ where $n$ is the number of vertices of the PTG. If we have to identify all $C_4$'s, the time complexity will be dominated by the search for all $C_4$'s. However, we could process $C_4$ hierarchically: a complex cycle $C$ may contain other complex cycles in the region bounded by the edges of $C$. We define a *maximal* $C_4$ (denoted by $MC_4$) to be a $C_4$ which is not contained in any other $C_4$ of the PTG. The number of $MC_4$'s in a PTG is $O(n)$. By definition, an $MC_4$ cannot contain another $MC_4$. When a PTG contains no $C_3$, two $MC_4$'s do not intersect. We only consider an $MC_4$ of a PTG when searching for a proper slice. We try to find a corner-slice for each $MC_4$. Due to rotation symmetry, there are four different corner-slicings of an $MC_4$.

**4.2. Slicing a planar triangulated graph.** The proposed algorithm for generating a sliceable floorplan from an $EPTG(G)$ is a divide-and-conquer algorithm. The fundamental task of the algorithm is to construct a proper slice $E_s$ from the $EPTG(G)$. From now on we will assume that the corners of $EPTG(G)$ are distinct and $G$ contains no cut vertices. If $G$ has nondistinct corners and/or contains cut vertices, generating a proper slice is trivial (see Fig. 5).

In general, an $EPTG(G)$ may have an exponential number of proper slices, while for others no proper slice may exist. The proof of Theorem 1 provides a method to construct a proper slice of an $EPTG(G)$. When $G$ contains no $C_4$ a proper slice always exists. In the proof, the initial slice $S_0$ is chosen to be the edge incident to the vertices on the right boundary of $G$. The choice of such $S_0$ ensures that the right boundary path $P_r(S_0)$ is a CFP since $EPTG(G)$ contains no $C_3$. We could have chosen any slice $S$ as long as $P_r(S)$ is a CFP. Different choices of $S$ will yield different floorplans. Choosing an $S$ where $P_r(S)$ is a CFP is not a difficult task. We start with an arbitrary path $P$, where the end vertices of $P$ are vertices $u(G)$ and $b(G)$. If $P$ is a CFP we construct $S$ from the edges incident to $P$ and on the left of $P$ (the left-induced slice of $P$). If $P$ is not a CFP we choose a maximal subset $P'$ of vertices of $P$ by traversing the chords so that $P'$ is a CFP. We then choose the edges on the left of $P'$ as $S$.

To search for a vertical CFP $P$, we define a search graph $G_s$, which is a directed subgraph of $EPTG(G)$, using the following procedure:

**P1.** For each $MC_4 = C_4(a,b,c,d)$ in $G$:
- **P1.1.** Delete all vertices (and incident edges) contained in $MC_4(a,b,c,d)$. Do not delete vertices $a,b,c,d$.
- **P1.2.** Add four directed edges $(a,b),(b,c),(c,d),(d,a)$ where the vertices $a,b,c,d$ are in counterclockwise order. If adding such edges results in two directed edges $(u,v)$ and $(v,u)$, delete both edges.
- **P1.3.** Add two undirected edges $(a,c)$ and $(b,d)$.

**P2.** Delete all edges incident to $\mathrm{LEFT_v}(G)$.
**P3.** Delete all edges incident to $r(G)$.
**P4.** Delete all edges in $\mathrm{TOP_e}(G) \cup \mathrm{BOTTOM_e}(G)$.
**P5.** Change all undirected edges incident to $u$ into directed edges away from $u$.
**P6.** Change all undirected edges incident to $b$ into directed edges toward $b$.
**P7.** Change all undirected edges on $\mathrm{RIGHT_e}(G)$ into directed edges pointing downwards.

An example of $EPTG(G)$ and its search graph $G_s$ is shown in Fig. 11. The undirected edges in $G_s$ can be traversed in both directions. We search for a directed CFP $P_s$ from $u(G)$ to $b(G)$ in $G_s$. If a diagonal edge $(a,c)$ in an $MC_4(a,b,c,d)$ (in counterclockwise order) is traversed in $G_s$, the corresponding path in $G$ should traverse

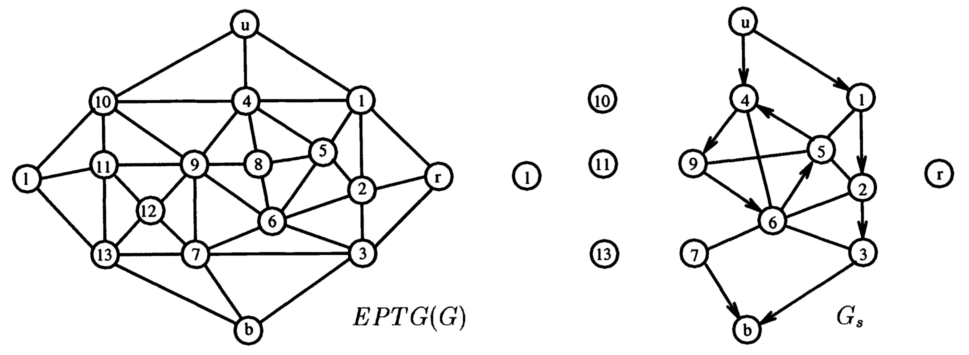


FIG. 11. *An EPTG(G) (left) and its search graph $G_s$ (right).*

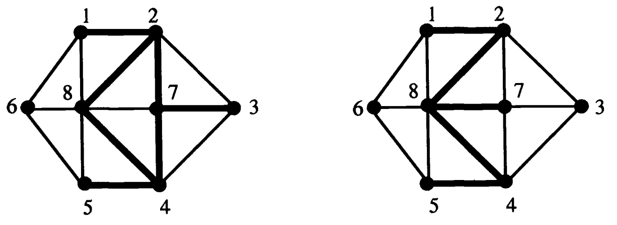


FIG. 12. *Modifying initial slice to satisfy the first condition of CFP.*

all vertices adjacent to vertex $b$ in $MC_4(a, b, c, d)$. For example, if we traverse the edge (4, 6) of $G_s$ in Fig. 11, we should traverse all vertices adjacent to vertex 5, i.e., (4, 8, 6) in $G$. From $P_s$ in $G_s$, we find the corresponding path $P_r(S_0)$ in $G$. The left-induced slice $S_0$ of $P_r(S_0)$ serves as an initial slice. For example, in Fig. 11,

$$
\begin{aligned}
&P_s = (u, 4, 6, 7, b) \in G_s, \\
&P_r(S_0) = (u, 4, 8, 6, 7, b) \in EPTG(G), \\
&S_0 = \{(4, 10), (4, 9), (8, 9), (6, 9), (7, 9), (7, 12), (7, 13)\}.
\end{aligned}
$$

Procedure **P1** guarantees that an $MC_4$ is corner-sliced and not center-sliced. **P1.2** is needed to avoid center-slicing an $MC_4$ on the right of path $P_s$. **P1.3** allows an $MC_4$ to be corner-sliced. **P2** is needed since $G_l$ must contain all vertices of $\text{LEFT}_\text{v}(G)$. (We have assumed that $G$ contains no cut vertices.) **P3** eliminates paths through $r(G)$. The rest of the procedure simplifies the analysis: if $P_r(S_0)$ traverses $\text{TOP}_\text{e}(G)$ or $\text{BOTTOM}_\text{e}(G)$, then it must contain a chord; **P4** excludes this possibility. Since the end vertices of $P_r(S_0)$ are $u(G)$ and $b(G)$, **P5** and **P6** are included. **P7** is included since $\text{RIGHT}_\text{v}(G)$ must be on $G_r$.

In general, there is more than one path in $G_s$. Heuristic measures can be employed to choose a good path. The search heuristics should try to corner-slice as many $MC_4$'s as possible. Corner-slicing is equivalent to traversing a diagonal edge of an $MC_4$ or traversing exactly one vertex of an $MC_4$ when searching for $P_s$ in $G_s$. If corner-slicing an $MC_4$ is not possible, we should try to avoid traversing any vertices of the $MC_4$. The unsliced $MC_4$ will appear in either $G_l$ or $G_r$ (exclusively). The $MC_4$ will be sliced when its subgraph is decomposed.

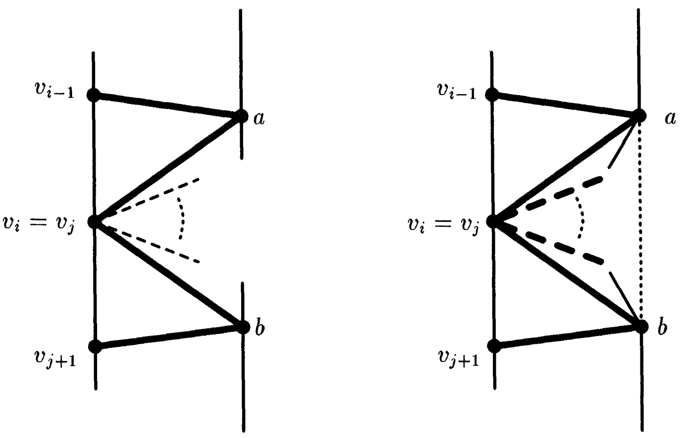


FIG. 13. *Initial slice (left) and modified initial slice (right).*

If no path exists in $G_s$, then there is no proper slice. In this case we have to accept an improper slice by center-slicing some $MC_4$. Again, we could impose heuristics to select an improper slice. When the presence of a chord is inevitable, we have to *fix the chord.* This will be discussed later in this section.

When the path $P_s$ on $G_s$ has been chosen, $P_r(S_0)$ in $EPTG(G)$ is determined. We obtain the left-induced slice $S_0$ of $P_r(S_0)$. By definition, $P_r(S_0)$ is also the right boundary path of the slice $S_0$. $P_r(S_0)$ is a CFP since we only accept CFPs when searching for $P_s$. There are two cases where the left boundary path $P_l(S_0) = (v_1, \dots, v_n)$ fails to be a CFP:

(a) For some $i \neq j, v_i = v_j$.

(b) There are some chords in $P_l(S_0)$.

The two cases are derived from the definition of a CFP. The first case is easy to solve. We construct a path $P_l'$ by excluding vertices $v_k, i < k < j$ from $P_l(S_0)$. The process is repeated until we obtain a path $P_l^*$ where the first condition is not violated. The right-induced slice of $P_l^*$ must exist and serves as the modified initial slice $S_0'$. An example of the procedure is shown in Fig. 12. The original slice is shown on the left with $P_r(S_0) = (2,3,4)$ and $P_l(S_0) = (1,8,7,8,5)$. On the right of the figure, $P_r(S_0) = (2,7,4)$ and $P_l(S_0) = (1,8,5)$.

We claim that the right boundary path $P_r(S_0')$ of modified slice $S_0'$ is still a CFP. Let $v_i = v_j, i < j$ in the original slice $S_0$. The neighborhood of the graph for the modified slice $S_0'$ is shown in Fig. 13. Thick edges represent edges in the slices $S_0$ and $S_0'$. Dotted edges may or may not exist. Solid edges and vertices must exist as shown in the configuration. From the figure, we can see that all vertices between vertices $a$ and $b$ in $P_r(S_0')$ are adjacent to $v_i = v_j$. Thus there are no chords $(x,y)$ among the vertices or we would have $C_3(x,y,v_i = v_j)$. Also, the vertices cannot form any chords with any vertices in the original $P_r(S_0)$ due to the subpath of $P_r(S_0)$ between vertices $a$ and $b$ (shown on the right with a dotted vertical line).

From the above construction, we can thus assume that the first condition of an initial slice $S_0$ is always satisfied. The only conditions that need to be examined are the chords of $P_l(S_0)$. If $G$ does not contain any $C_4$, we can apply the bypassing operation described in Theorem 1 to obtain a proper slice. In the presence of $C_4$, the operation is still successful if the maximal chord is not part of any $C_4$.

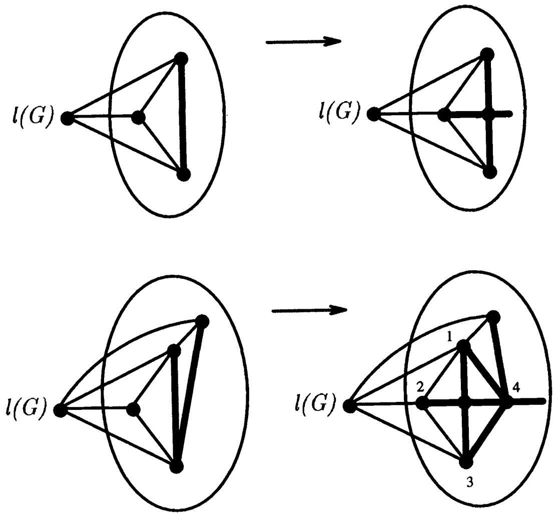


FIG. 14. *Fixing the chords of an improper slice.*

When all of the above methods fail and a chord cannot be avoided, the algorithm has to accept an improper slice. When this occurs, each chord will contribute a $C_3$ in one of the subgraphs, say $EPTG(G_r)$. We delete the chord, add a vertex in the middle of the chord and four new edges. The operation is called *fixing a chord*, as shown in Fig. 14. Fixing a chord eliminates the $C_3$ caused by the chord and maintains triangulation of graph $EPTG(G_r)$. Note that when we fix a chord, a new $C_4$ is created (e.g., $C_4(1,2,3,4)$ of Fig. 14). However, three vertices of the $C_4$ fall in the boundary of $G$. This ensures that the $C_4$ will be corner-sliced in the future. $C_3$'s of $EPTG(G_l)$ are handled similarly.

A summary of the algorithm is described as follows.

```
ALGORITHM SLICE($EPTG(G)$)

INPUT: An extended dual $EPTG(G)$.
OUTPUT: A planar triangulated graph $G'$ and a sliceable floorplan $F(G')$. $G'$ is $G$
        with some pseudovertices added. If $EPTG(G)$ contains no $C_3$ and $G$ contains
        no $C_4$, then $G' = G$.
BEGIN
    1. Check if:
        1.1. the corners of $EPTG(G)$ are not distinct.
        1.2. $G$ contains a cut vertex.
        if so, a slice is trivially generated.
    2. Find all $MC_4$'s of $G$.
    3. Determine the orientation of the slice: Horizontal or Vertical.
    4. Construct the search graph $G_s$.
    5. Search for a path $P_s$ in $G_s$ which satisfies the following criteria:
        Priority 1: Maximize the number of $MC_4$'s which are corner-sliced by $P$.
        Priority 2: Minimize the number of $MC_4$'s which are center-sliced by $P$.
    6. Find the left-induced slice $S_0$ in $G$ from $P_s$.
```

```
 7. Modify the slice $S_0$ to eliminate vertices of $P_l(S_0)$ so that the first condition
    of CFP is not violated.
 8. Find the maximal chords of $P_l(S_0)$.
 9. For each maximal chord $e_p$, perform a bypass operation on $e_p$ and obtain a
    final slice $E_s$. /* see Theorem 1 */
10. Decompose $G$ into $G_l$ and $G_r$ and obtain $EPTG(G_l)$ and $EPTG(G_r)$.
11. If $EPTG(G_r)$ or $EPTG(G_l)$ contains a $C_3$ due to chords, fix the chords by
    adding a pseudovertex and corresponding edges. Update $G'$ if pseudovertices
    are added. /* see Fig. 14 */
12. Recursively call SLICE($EPTG(G_l)$) and SLICE($EPTG(G_r)$).
13. Construct floorplan $F(G')$ by merging the floorplans of $F(G'_l)$ and $F(G'_r)$.
END.
```

The $MC_4$ of a graph can be found in $O(n \log n)$ time, where $n$ is the number of vertices of $G$. The operation need not be repeated during the recursive steps since the $MC_4$'s of $G_l$ and $G_r$ are also those of $G$. An $O(n \log n)$ algorithm for finding all $MC_4$'s is described in Appendix II.

The search graph $G_s$ can be constructed in linear time. The path $P$ can also be found in linear time using simple heuristics, e.g., using a depth-first search which incorporates some cost measure to achieve balanced slice. The bypassing operation also has linear time complexity since the number of edges in a planar graph is $O(n)$. Standard planar graph data structures like a doubly connected edge list [11] are sufficient for the algorithm. All other steps in the algorithm are standard operations on planar graphs which can be achieved in $O(n)$ time.

The time complexity of the recursive part of the algorithm is

$$T(n) = T(n_l) + T(n_r) + O(n),$$

where $n_l$ and $n_r$ are the number of vertices of $G_l$ and $G_r$, respectively, with $n_l + n_r = n$. The operation of fixing a chord adds at most $c$ vertices where $c$ is the number of chords. Since $c < n$, the time complexity remains unchanged. At each level of the recursion, we spend $O(n)$ time. If we consider the output floorplan tree, we spend $O(n)$ time in total to construct the tree nodes at each level. Thus we have

$$T(n) = O(hn),$$

where $h$ is the height of the floorplan tree.

In general, $n_l$ and $n_r$ are not balanced. In fact, there exist instances of input $EPTG(G)$ where the trees are extremely unbalanced. The graph shown in Fig. 15 is an example. The graph belongs to a class of $EPTG(G)$ called *4-contractable.* The floorplan of a 4-contractable graph is unique [8] as given by the figure; thus $h = O(n)$. $h$ is $\theta(\log n)$ when a balanced partition can be achieved for the nonterminal nodes of the floorplan tree. Since finding $MC_4$ requires $O(n \log n)$, the time complexity of the algorithm is $O(n \log n + hn)$.

## 5. Algorithm improvements and extensions.

**5.1. Minimizing wasted area.** When a proper slice does not exist, we perform a chord fixing operation. For each chord, a pseudovertex is added to the original $G$. These pseudovertices contain no circuit elements in VLSI layout. They are added merely to satisfy the adjacency requirements specified by $G$. In the final chip layout, the rectangles corresponding to the pseudovertices contain only routing wires with

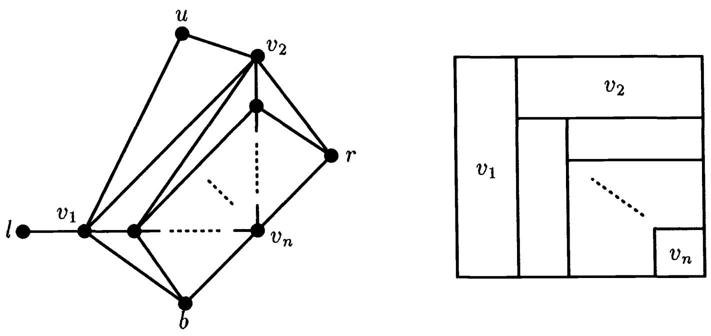


FIG. 15. *An inherently unbalanced floorplan.*

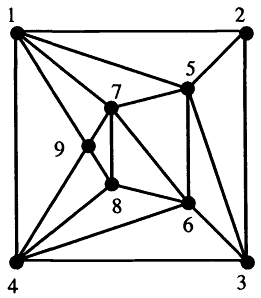


FIG. 16. *Choosing a slice to minimize wasted area.*

no devices. Minimizing the area consumed by these pseudovertices becomes a major concern. The routing area depends on the number of routing wires of the edge being fixed. Therefore, the slicing algorithm should prefer chords with less routing wires.

Proper choice of a slice also helps in reducing the wasted area. Consider $C_4(1,2,3,4)$ in Fig. 16. Suppose corner-slicing of the $C_4$ is not possible. We have to slice the $C_4$ with edges (1, 2), (3, 4). If we choose the slice $\{(1,2),(1,5),(1,7),(1,9),(4,9),(4,8),(4,6),(4,3)\}$, we have three chords (2, 3), (5, 6), and (7, 8). However, if we choose $\{(1,2),(2,5),(3,5),(3,6),(3,4)\}$, we have only one chord (1, 4). The proper choice of slicing depends on the wiring density of the edges and the configuration of the $C_4$. The objective is to find a slice with less wasted area after pseudovertices are added to fix the chords.

**5.2. Generating families of sliceable floorplans and sizing.** In general, there is a family of floorplans which have identical dual $G$. An algorithm for generating all floorplans in the family is an immediate extension of the basic algorithm. In [2], algorithms for generating all floorplans for an input planar triangulated graph were discussed. In [7], an algorithm that generates all floorplans with linear time complexity per floorplan was reported. It should be noted that the number of floorplans in a family may be exponential in terms of the number of vertices.

Our algorithm is immediately extended to generate all sliceable floorplans of the input graph. Two sliceable floorplans are distinct if and only if their tree representations are distinct. The problem of enumerating all floorplans is reduced to the task of enumerating all proper slices on the search graph $G_s$. Any algorithm for enumerating paths can be employed.

**5.3. Pseudovertices.** Figure 17 demonstrates the outputs of the algorithm

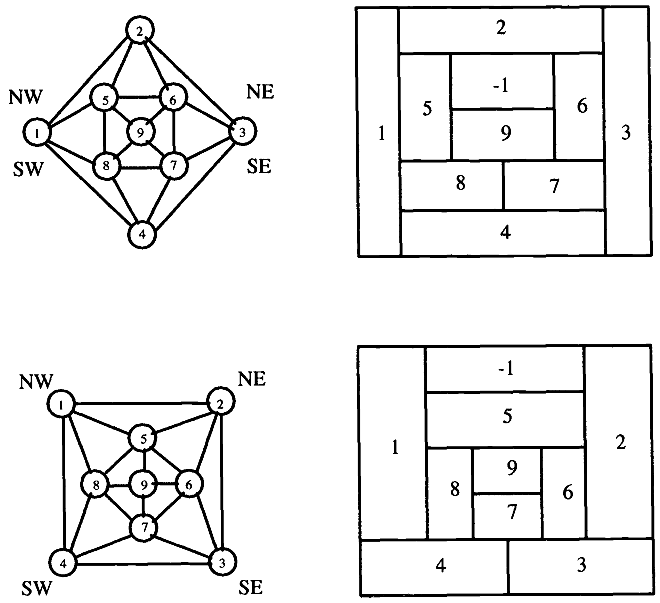


FIG. 17. *Adding pseudovertices to maintain sliceability.*

where pseudovertices are added to satisfy the sliceability requirement. The inputs are taken from graph of Fig. 1, which does not admit any sliceable floorplan. The two extended graphs show identical adjacency graphs with different corner assignments. The outputs of the program are also shown in the figure. The negative numbered rectangles in the floorplans represent pseudomodules.

Experiments using randomly generated planar triangulated graphs have shown that the number of pseudovertices added to the input graph is approximately 22% of the number of input vertices. On the pseudovertices added, approximately 70% (16% of the number of input vertices) are generated by $C_3$ of the original input. This represents the inherent price we have to pay for using the rectangular dual graph approach. But with only an additional 6% more vertices, we are able to generate sliceable floorplans. Since sliceable floorplans have many desirable characteristics, it is justifiable to pay some extra cost by applying our algorithm.

**6. Conclusions and future work.** An algorithm for generating sliceable floorplans based on a rectangular approach has been presented. If the input graph contains no $C_4$ (complex cycle of length 4), then the algorithm always generates a sliceable floorplan. In the presence of $C_4$, a sliceable floorplan is also possible though not guaranteed. With minor modifications to the adjacency requirements of an input graph, the algorithm always generates a sliceable floorplan even when the input graph is known to be inherently nonsliceable. Typically, the sliceability requirement only introduces a small number of pseudovertices to the original input graph, which makes the algorithm very suitable for practical applications.

Many problems remain unsolved in the rectangular dualization approach to floorplanning. An important issue is the planarization and triangulation of the given adjacency graph to obtain a "good" planar triangulated graph. For example, the qualitative effect of planarization to the final floorplan is still unknown. Practical issues such as incorporating weights to vertices and edges of the graph are not well formulated. In particular, the effects of weight measures on the choice of slices are not

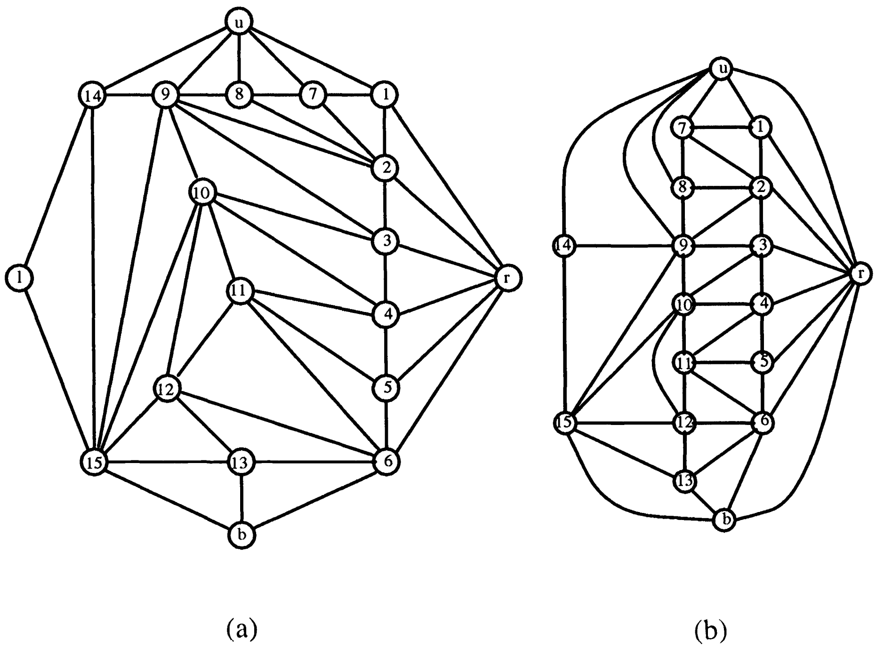


FIG. 18. *Construction of a proper slice when four corner vertices are distinct.* (a) *an* $EPTG(G)$ *with no* $C_3$ *and* $C_4$. (b) *breadth first search drawing of* $EPTG(G)$.

well understood. The problem of incorporating sizing in the floorplanning system is currently under investigation. Another interesting problem is the enumeration of all sliceable floorplans, currently being investigated by the authors.

## Appendix I. An example for the proof of Theorem 1.

Figure 18(a) shows an $EPTG(G)$ with no $C_3$ and $G$ with no $C_4$. The BFS drawing of the graph is depicted in Fig. 18(b). The construction of a proper slice $E_s$ is as follows:

$$\begin{aligned}
&S_0 = \{(1,7),(2,7),(2,8),(2,9),(3,9),(3,10),(4,10),\\
&\qquad (4,11),(5,11),(6,11),(6,12),(6,13)\},\\
&P_l(S_0) = \{u,1,2,3,4,5,6,b\},\\
&P_r(S_0) = \{u,7,8,9,10,11,12,13,b\},\\
&\text{chords of } P_l(S_0) = \{(u,8),(u,9),(10,12)\},\\
&\text{maximal chords } E_c(S_0) = \{(u,9),(10,12)\},\\
&v_0^1 = u, v_1^1 = 1, v_2^1 = 2, v_3^1 = 3, v_4^1 = 4, v_5^1 = 5, v_6^1 = 6, v_7^1 = b,\\
&v_0^2 = u, v_1^2 = 7, v_2^2 = 8, v_3^2 = 9, v_4^2 = 10, v_5^2 = 11, v_6^2 = 12, v_7^2 = 13, v_8^2 = b.
\end{aligned}$$

For maximal chord $e_1 = (v_i^2, v_j^2) = (v_0^2, v_3^2) = (u, 9)$,

$$\begin{aligned}
&i = 0, j = 3, k = 1, l = 2, v_m = 8,\\
&E_{tz}(i+1, j-1, l, k) = E_{tz}(1,2,2,1) = \{(1,7),(2,7),(2,8)\},\\
&E_1(i) = E_1(0) = \phi, \qquad E_1(j) = E_1(3) = \{(8,9)\}.
\end{aligned}$$

For maximal chord $e_2 = (v_i^2, v_j^2) = (v_4^2, v_6^2) = (10, 12)$,

$$\begin{aligned}
&i = 4, j = 6, k = 4, l = 6, v_m = 11,\\
&E_{tz}(i+1, j-1, l, k) = E_{tz}(5,5,6,4) = \{(4,11),(5,11),(6,11)\},\\
&E_2(i) = E_2(4) = \{(10,11)\}, \qquad E_2(j) = E_2(6) = \{(11,12)\}.
\end{aligned}$$

To construct $E_s$,

$$\begin{aligned}
S_1 &= S_0 - E_{tz}(1,2,2,1) + E_1(i=0) + E_1(j=3)\\
&= \{(8,9),(2,9),(3,9),(3,10),(4,10),(4,11),(5,11),(6,11),(6,12),(6,13)\},\\
E_s &= S_2 = S_1 - E_{tz}(5,5,6,4) + E_2(i=4) + E_2(j=6)\\
&= \{(8,9),(2,9),(3,9),(3,10),(4,10),(10,11),(11,12),(6,12),(6,13)\},\\
P_l(E_s) &= (u, 9, 10, 12, 13, b),\\
P_r(E_s) &= (u, 8, 2, 3, 4, 11, 6, b).
\end{aligned}$$

It can be easily verified that $E_s$ is a slice and $P_r(E_s), P_l(E_s)$ are CFPs.

**Appendix II. Finding maximal $C_4$ of a planar triangulated graph.** The algorithm requires a subalgorithm to detect $MC_4$. We present a divide-and-conquer algorithm to find all $MC_4$ of a PTG.

We partition the input graph $G$ into vertex-balanced left and right subgraphs $G_l, G_r$ with a slice $C = (e_1, \ldots, e_c)$. The slice need not strictly satisfy the condition that $e_1 \in \mathrm{TOP_e}(G)$ and $e_c \in \mathrm{BOTTOM_e}(G)$. We only require that $e_1$ and $e_c$ be external edges while $e_i, 1 < i < c$, are not external edges. We call $C$ a *cutset.* A vertex-balanced cutset can be easily generated by some tree search techniques (e.g., breadth first search) until half of the vertices are visited.

Let $P_l(C) = (v_0^l, \ldots, v_L^l)$ and $P_r(C) = (v_0^r, \ldots, v_R^r)$ be the left and right boundary paths of $C$. By Lemma 5, the intersection of an $MC_4(a,b,c,d)$ consists of either zero or two edges. At each recursion step, the algorithm finds all $MC_4$'s that intersect the slice $C$. Referring to the discussions of §4.1, an $MC_4(a,b,c,d)$ is either corner-sliced or center-sliced. We consider both cases separately.

*Case* 1 (center-sliced: $(a,b),(c,d) \in C$). Without loss of generality, we assume that $a, d \in G_l$ and $b, c \in G_r$ (see Fig. 10). Since vertices $a, b, c, d \in P_l(C) \cup P_r(C)$, we can delete all vertices not in $P_l(C) \cup P_r(C)$. Now, the edges of $G_l(G_r)$ are only incident vertices of $P_l(C)(P_r(C))$. Let the remaining graph be $G_d$.

For the purpose of discussion, we assume that the vertices of $P_l(C)$ and $P_r(C)$ are distinct. The modifications needed when their vertices are not distinct will be discussed later. For each vertex $v$ of $P_l(C)$ and $P_r(C)$, we find the minimum and maximum edge indices of $C$ whose edges are incident to $v$. Let $v_{\min}$ and $v_{\max}$ be the indices. For example, if $v$ is incident to edges $e_3, e_4, e_5, v_{\min} = 3$ and $v_{\max} = 5$. Let $u, v$ be two vertices of $P_l(C)$ where $u$ precedes $v$ when $P_l(C)$ is traversed. There is an important monotonic property:

$$u_{\min} \le u_{\max} < v_{\min} \le v_{\max}.$$

The property also holds for $P_r(C)$.

Let the first edge of $C, e_1 = (a,b)$ with $a \in G_l$ and $b \in G_r$. Let $(a,d) \in G_l$ and $(b,c) \in G_r$ be the two external edges of $G_d$ incident to vertices $a$ and $d$, respectively. The situation is depicted in Fig. 19(a). We search for the existence of $MC_4(a,b,c,d)$.

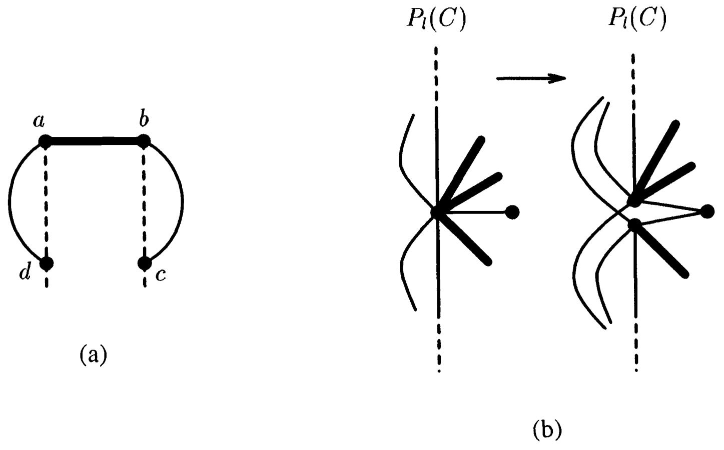


FIG. 19. *Finding $MC_4(a,b,c,d)$, where $(a,b),(c,d) \in C$.* (a) *finding $MC_4$ in Case* 1.1; (b) *duplicating vertices to maintain monotonic property.*

Let $[i,j]$ denote the set of integers between $i$ and $j$ (inclusive). Consider the min and max indices carried on vertices $c$ and $d$. There are two possibilities:

(1) $[c_{\min}, c_{\max}] \cap [d_{\min}, d_{\max}] = \phi$;

(2) $[c_{\min}, c_{\max}] \cap [d_{\min}, d_{\max}] = K \neq \phi$.

In the first case, edge $(c,d)$ does not exist. If $c_{\min} > d_{\max}$, we delete the external edge $(b,c)$ in $G_d$. By the monotonic property, any subsequent vertex visited by the algorithm $d' \in P_l(C)$ incident to vertex $a$ will have

$$d'_{\max} < d_{\min} \leq d_{\max} < c_{\min},$$

because $d'$ precedes $d$ in $P_l(C)$. Thus $d'$ is not adjacent to $c$. Conversely, if $d_{\min} > c_{\max}$, we delete edge $(a,d)$ for the same reason.

For the second case, $MC_4(a,b,c,d)$ exists with edge $(c,d)$ in $C$. However, $K$ must contain exactly one integer. Otherwise, we would have at least two distinct edges $e_i, e_j$ with $i,j \in K, i \neq j$, simultaneously incident to vertices $c$ and $d$. This is a contradiction since there are no duplicate edges in $G$.

If an $MC_4$ is found, all vertices and incident edges in the area bounded by the $MC_4$ are deleted. After all external edges incident to vertices $a$ and $b$ are exhausted, we delete $(a,b)$ and select the first remaining edge in $C$ (which is an external edge).

When $P_l(C)$ contains nondistinct vertices, the monotonic property fails. However, by duplicating the nondistinct vertices, we are able to maintain the monotonic property. Each time we encounter a vertex $v$ which has been traversed in $P_l(C)$, we make duplicate vertices $v'$ at a very small distance from $v$ and redistribute the incident edges. The resulting graph may not be planar but it does not affect the solution. The operation is shown in Fig. 19(b). The thick edges represent edges in $C$. A similar procedure is applied to $P_r(C)$.

*Case* 2 (corner-sliced: $(a,b),(b,c) \in C$). Assume $a,c,d \in G_l$ and $b \in G_r$ (see Fig. 10). First, we delete the following edges and vertices since they will not be part of any $MC_4(a,b,c,d)$ we are searching for:

(a) all edges not incident to vertices of $P_l(C) \cup P_r(C)$;

(b) all remaining edges of $G_r$;

(c) all remaining vertices of degree less than 2;

(d) all remaining vertices not incident to the infinite face.

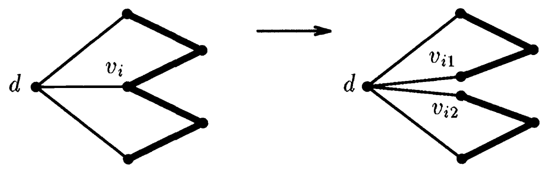


FIG. 20. *Duplicating vertices and edges to find $MC_4(a,b,c,d)$, where $(a,b),(b,c) \in C$.*

Let $G_d$ be the resulting graph. Consider a vertex $d \in G_l - P_l(C)$. Let $\{(d,v_1),\ldots,(d,v_n)\}$ be the incident edges in $G_d$. Since $d \notin P_l(C), d$ cannot be adjacent to vertices of $G_r$. If $v_i \notin P_l(C)$ for some $i$, then the edge $(d,v_i)$ would have been deleted in step (a). Thus $v_i \in P_l(C)$ for all $i$.

If an edge $(d,v_i)$ is incident to the infinite face of $G_d$, we do nothing. If $(d,v_i)$ is not incident to the infinite face, we duplicate the vertex $v_i$ and the edge $(d,v_i)$ as shown in Fig. 20. (The thick edges represent edges in $C$.) The duplicate vertices $v_{i1}$ and $v_{i2}$ are spaced a sufficiently small distance apart. Let $(d,v_{i1})$ and $(d,v_{i2})$ be the duplicate edges. Since the original $v_i$ is incident to the infinite face (by step (d)), edges $(d,v_{i1})$ and $(d,v_{i2})$ are now incident to the infinite face. We perform such modification for all vertices $v_i$ and all vertices $d \in G_l - P_l(C)$.

Consider an $MC_4(a,b,c,d)$ in $G_d$, where $a,c,d \in G_l$ and $b$ in $G_r$. Because of the above modification, the edges $(a,d)$ and $(c,d)$ of the $MC_4$ are now incident to the infinite face. $MC_4(a,b,c,d)$ can be easily identified by considering all vertices $b$ in $P_r(C)$. There is at most one $MC_4$ corresponding to each vertex $b \in P_r(C)$.

The algorithm is repeated with left and right subgraphs interchanged for $MC_4(a,b,c,d)$, where $b \in G_l$ and $a,c,d \in G_r$.

Combining the two cases above, we find all $MC_4$ with edges in the cutset $C$. The algorithm can be recursively applied to $G_l$ and $G_r$ to find other $MC_4$'s. The algorithm is shown in pseudoinstructions below.

```
ALGORITHM FIND__MC4($G$)

INPUT: A planar triangulated graph $G$.
OUTPUT: All $MC_4$ of $G$.
BEGIN
    1. Find a balanced cutset $C$ where the left subgraph $G_l$ and right subgraph $G_r$
       are connected. For example, use breadth first search until half of the vertices
       in $G$ are visited.
    2. FIND__MC4__CASE__1($G_l, G_r, C$).
    3. FIND__MC4__CASE__2($G_l, G_r, C$).
    4. FIND__MC4__CASE__2($G_r, G_l, C$).
    5. FIND__MC4($G_r$).
    6. FIND__MC4($G_l$).
END.

PROCEDURE FIND__MC4__CASE__1($G_l, G_r, C$)

INPUT: A cutset $C$ and left and right subgraphs $G_l, G_r$.
OUTPUT: All $MC_4(a,b,c,d)$ where $a,d \in G_l$ and $b,c \in G_r$.
BEGIN
    1. Delete all vertices $v \notin P_l \cup P_r$.
    2. If $P_l(C)$ or $P_r(C)$ is nondistinct, make duplicate vertices and redistribute the
       edges. See Fig. 19(b).
    3. Let $G_d$ be the resulting graph. The vertices of $P_l(C)$ and $P_r(C)$ now have
       monotonic property.
    4. FOR $v \in P_l(c) \cup P_r(C)$ DO
        4.1. $v_{\min}$ = min index of incident edges in $C$.
        4.2. $v_{\max}$ = max index of incident edges in $C$.
        END FOR.
    5. WHILE exist $(a,b) \in C$ which is an external edge DO
        5.1. Let $a \in G_l, b \in G_r$.
        5.2. Let $(a,d) \in G_l$ and $(b,c) \in G_r$ where $(a,d)$ and $(b,c)$ are external edges.
        5.3. IF $[c_{\min}, c_{\max}] \cap [d_{\min}, d_{\max}] = \phi$ THEN
            5.3.1 IF $c_{\min} > d_{\max}$ THEN delete the external edge $(b,c)$.
            5.3.2 IF $d_{\min} > c_{\max}$ THEN delete the external edge $(a,d)$.
        5.4. ELSE
            5.4.1. IF there are vertices in the cycle $(a,b,c,d)$ THEN
                5.4.1.1. Report $MC_4(a,b,c,d)$.
                5.4.1.2. Delete edges and vertices inside $MC_4(a,b,c,d)$.
        END WHILE.
END.
```

PROCEDURE **FIND__MC4__CASE__2**$(G_l, G_r, C)$

```
INPUT: A cutset $C$ and left and right subgraphs $G_l, G_r$.
OUTPUT: All $MC_4(a,b,c,d)$ where $a,b,d \in G_l$ and $c \in G_r$.
BEGIN
    1. Delete all edges $(u,v)$ where both $u,v \notin P_l(C) \cup P_l(C)$.
    2. Delete all edges in $G_r$.
    3. Delete all vertices with degree less than 2.
    4. Delete all vertices not incident to infinite face.
    5. Let $G_d$ be the resulting graph.
    6. FOR $d \in (G_d \cap G_l) - P_l(C)$ DO
        6.1. FOR $v_i$ incident to $d$ DO
            6.1.2. IF $(d, v_i)$ is not incident to the infinite face THEN
                   Duplicate vertices $v_{i1}, v_{i2}$, edges $(d, v_{i1}), (d, v_{i2})$ as shown in
                   Fig. 20.
            END FOR.
        END FOR.
    7. FOR $b \in G_d \cap G_r \cap P_r(C)$ DO
        7.1. If external edges $(a,b)$ and $(b,c)$ exist THEN
            7.1.1. Let $(b,d_1)$ and $(c,d_2)$ be the external edges where $d_1, d_2 \in (G_d \cap$
                   $G_l) - P_l(C)$.
            7.1.2. IF $d_1 = d_2$ and there are vertices in the cycle $(a,b,c,d)$ THEN
                   Report $MC_4(a,b,c,d_1)$.
        END FOR.
END.
```

At each recursive step, the algorithm **FIND__MC4()** visits the edges of $G$ at most a constant number of times. Although step 6 of **FIND__MC4__CASE__2()** contains two nested loops, the time complexity remains linear because each edge $(d, v_i)$ is visited a constant number of times in **6.1.2.** Therefore, the time complexity of the algorithm is $O(n \log n)$.

## REFERENCES

[1] K. Kozminski and E. Kinnen, *Rectangular dual of planar graphs*, Networks, 15 (1985), pp. 145–157.

[2] ———, *Rectangular dualization and rectangular dissection*, IEEE Trans. Circuits Systems, 35 (1988), pp. 1401–1416.

[3] Y. T. Lai and S. M. Leinwand, *Algorithms for floorplan design via rectangular dualization*, IEEE Trans. Comput. Aided Design, 7 (1988), pp. 1278–1289.

[4] J. Bhasker and S. Sahni, *A linear algorithm to find a rectangular dual of a planar triangulated graph*, Algorithmica, 3 (1988), pp. 247–278.

[5] Y. Sun and M. Sarrafzadeh, *Floorplanning by graph dualization: L-shaped models*, in Proc. IEEE Internat. Symposium on Circuits and Systems, New Orleans, LA, 1990, pp. 2845–2848; Algorithmica, 10 (1993), pp. 429–456.

[6] K. H. Yeap and M. Sarrafzadeh, *2-concave rectilinear polygons—necessary and sufficient for graph dualization*, in Proc. 29th Annual Allerton Conference on Communication, Control, and Computing, Urbana, IL, 1991, University of Illinois at Urbana-Champaign.

[7] S. Tsukiyama, K. Tani, and T. Maruyama, *A condition for a maximal planar graph to have a unique rectangular dual and its application to VLSI floorplan*, in Proc. IEEE Internat. Symposium on Circuits and Systems, Portland, OR, 1989, pp. 931–934.

[8] S. Tsukiyama, K. Koike, and I. Shirakawa, *An algorithm to eliminate all complex triangles in a maximal planar graph for use in VLSI floorplan*, in Proc. IEEE Internat. Symposium on Circuits and Systems, San Jose, CA, 1986, pp. 321–324.

[9] R. Otten, *Efficient floorplan optimization*, in Proc. Internat. Conference on Computer-Aided Design, Santa Clara, CA, 1984, pp. 499–502.

[10] L. Stockmeyer, *Optimal orientation of cells in slicing floorplan designs*, Inform. Control, 57 (1983), pp. 91–101.

[11] F. Preparata and M. Shamos, *Computational Geometry—An Introduction*, Springer-Verlag, 1985.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 281–290, May 1995

# TWO-WAY ROUNDING*

DONALD E. KNUTH†

**Abstract.** Given $n$ real numbers $0 \le x_1, \ldots, x_n < 1$ and a permutation $\sigma$ of $\{1, \ldots, n\}$, we can always find $\bar{x}_1, \ldots, \bar{x}_n \in \{0,1\}$ so that the partial sums $\bar{x}_1 + \cdots + \bar{x}_k$ and $\bar{x}_{\sigma 1} + \cdots + \bar{x}_{\sigma k}$ differ from the unrounded values $x_1 + \cdots + x_k$ and $x_{\sigma 1} + \cdots + x_{\sigma k}$ by at most $n/(n+1)$, for $1 \le k \le n$. The latter bound is best possible. The proof uses an elementary argument about flows in a certain network, and leads to a simple algorithm that finds an optimum way to round.

**Key words.** rounding, partial sums, network flows, discrepancy

**AMS subject classifications.** 90C27, 90B10, 05A05

**0. Introduction.** Many combinatorial optimization problems in integers can be solved or approximately solved by first obtaining a real-valued solution and then rounding to integer values. Spencer [11] proved that it is always possible to do the rounding so that partial sums in two independent orderings are properly rounded. His proof was indirect—a corollary of more general results [7] about discrepancies of set systems—and it guaranteed only that the rounded partial sums would differ by at most $1 - 2^{-2^n}$ from the unrounded values. The purpose of this note is to give a more direct proof, which leads to a sharper result.

Let $x_1, \ldots, x_n$ be real numbers and let $\sigma$ be a permutation of $\{1, \ldots, n\}$. We will write

$$S_k = x_1 + \cdots + x_k\,, \qquad \Sigma_k = x_{\sigma 1} + \cdots + x_{\sigma k}\,, \qquad 0 \le k \le n\,,$$

for the partial sums in two independent orderings. Our goal is to find integers $\bar{x}_1, \ldots, \bar{x}_n$ such that

$$\lfloor x_k \rfloor \le \bar{x}_k \le \lceil x_k \rceil\,,$$

and such that the rounded partial sums

$$\overline{S}_k = \bar{x}_1 + \cdots + \bar{x}_k\,, \qquad \overline{\Sigma}_k = \bar{x}_{\sigma 1} + \cdots + \bar{x}_{\sigma k}$$

also satisfy

$$(*) \qquad \lfloor S_k \rfloor \le \overline{S}_k \le \lceil S_k \rceil\,, \qquad \lfloor \Sigma_k \rfloor \le \overline{\Sigma}_k \le \lceil \Sigma_k \rceil\,,$$

for $0 \le k \le n$. Such $\bar{x}_1, \ldots, \bar{x}_n$ will be called a *two-way rounding* of $x_1, \ldots, x_n$ with respect to $\sigma$.

LEMMA. *Two-way rounding is always possible.*

*Proof.* We can assume without loss of generality that $S_n = m$ is an integer, by adding an additional term and increasing $n$ if necessary. We can also assume that $0 < x_k < 1$ for all $k$. Construct a network with nodes $\{s, a_1, \ldots, a_m, u_1, \ldots, u_n, v_1, \ldots, v_n, b_1, \ldots, b_m, t\}$ and the following arcs:[1]

$$s \longrightarrow a_j \qquad \text{and} \qquad b_j \longrightarrow t \qquad \text{for} \qquad 1 \le j \le m\,;$$

*Received by the editors December 20, 1993; accepted for publication (in revised form) April 6, 1994.

†Computer Science Department, Stanford University, Stanford, California 94305-2140.

[1]Here and in what follows $[a \mathinner{..} b)$ denotes the half-open interval $\{\, x \mid a \le x < b \,\}$. This notation, due independently to Hoare and Ramshaw, is recommended in [5].

$$\begin{aligned} u_k &\to v_k && \text{for} && 1 \le k \le n; \\ a_j &\to u_k && \text{if} && [j-1 \mathinner{.\,.} j) \cap [S_{k-1} \mathinner{.\,.} S_k) \ne \emptyset; \\ v_{\sigma k} &\to b_j && \text{if} && [j-1 \mathinner{.\,.} j) \cap [\Sigma_{k-1} \mathinner{.\,.} \Sigma_k) \ne \emptyset. \end{aligned}$$

Each arc has capacity 1. This network supports a natural flow of $m$ units, if we send 1 unit through each arc $s \to a_j$ and $b_j \to t$, and $x_k$ units through $u_k \to v_k$; the flow in $a_j \to u_k$ is the measure of the interval $[j-1 \mathinner{.\,.} j) \cap [S_{k-1} \mathinner{.\,.} S_k)$, and the flow in $v_{\sigma k} \to b_j$ is similar. Deleting the arcs $s \to a_j$ defines a cut of capacity $m$, so this must be a minimum cut.

Since the arc capacities are integers, the max-flow/min-cut theorem implies that this network supports an integer flow of $m$ units. Let $\bar{x}_k$ be the amount that flows through $u_k \to v_k$, for $1 \le k \le n$, in one such flow. Then $\bar{x}_k \in \{0,1\}$. If $j = \lceil S_k \rceil$ we have $\overline{S}_k = \bar{x}_1 + \cdots + \bar{x}_k =$ flow into $\{u_1, \ldots, u_k\} \le$ flow out of $\{a_1, \ldots, a_j\} = j$, because all arcs $a_i \to u_l$ for $l \le k$ have $i \le j$. If $j = \lfloor S_k \rfloor$ then $\overline{S}_k =$ flow into $\{u_1, \ldots, u_k\} \ge$ flow out of $\{a_1, \ldots, a_j\} = j$, because all arcs $a_i \to u_l$ for $i \le j$ have $l \le k$. A similar argument proves that $\lfloor \Sigma_k \rfloor \le \overline{\Sigma}_k \le \lceil \Sigma_k \rceil$, hence $(*)$ holds. □

COROLLARY. *Given any fixed $k$, two-way rounding is possible with $\bar{x}_k = \lceil x_k \rceil$, as well as with $\bar{x}_k = \lfloor x_k \rfloor$.*

*Proof.* We may assume as before that $0 < x_k < 1$. The construction in the lemma establishes a feasible flow of $x_k$ units in the arc $u_k \to v_k$. It is well known that the polytope of all feasible flows has vertices whose coordinates are integers (see, for example, Application 19.2 in Schrijver [10]). Therefore the arc $u_k \to v_k$ is saturated in at least one maximum flow, and it carries no flow at all in at least one other. □

Incidentally, it is important to impose a capacity of 1 on the arcs $u_k \to v_k$ in the construction of this proof. Otherwise we might get solutions in which $\bar{x}_k = 2$. Condition $(*)$ does not by itself imply that $\bar{x}_k \le \lceil x_k \rceil$ or that $\bar{x}_k \ge \lfloor x_k \rfloor$.

## 1. An application.

Sometimes it is desirable to round "spreadsheet" data to larger units while preserving row and column sums and the grand total.

THEOREM 1. *Given $mn$ real numbers $x_{ij}$ for $1 \le i \le m$, $1 \le j \le n$, we can round them to integers $\bar{x}_{ij}$ in such a way that*

$$\lfloor x_{ij} \rfloor \le \bar{x}_{ij} \le \lceil x_{ij} \rceil, \qquad \textit{for all } i \textit{ and } j;$$

$$\left\lfloor \sum_{j=1}^{n} x_{ij} \right\rfloor \le \sum_{j=1}^{n} \bar{x}_{ij} \le \left\lceil \sum_{j=1}^{n} x_{ij} \right\rceil, \qquad \textit{for all } i;$$

$$\left\lfloor \sum_{i=1}^{m} x_{ij} \right\rfloor \le \sum_{i=1}^{m} \bar{x}_{ij} \le \left\lceil \sum_{i=1}^{m} x_{ij} \right\rceil, \qquad \textit{for all } j;$$

*and*

$$\left\lfloor \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \right\rfloor \le \sum_{i=1}^{m} \sum_{j=1}^{n} \bar{x}_{ij} \le \left\lceil \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \right\rceil.$$

*Proof.* Let $a_i = \sum_{j=1}^{n} x_{ij}$, $b_j = \sum_{i=1}^{m} x_{ij}$, and $s = \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$, and consider the $(m+1) \times (n+1)$ array

$$\begin{array}{ccccc} x_{11} & x_{12} & \ldots & x_{1n} & \alpha_1 \\ x_{21} & x_{22} & \ldots & x_{2n} & \alpha_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \ldots & x_{mn} & \alpha_m \\ \beta_1 & \beta_2 & \ldots & \beta_n & s \end{array}$$

where $\alpha_i = -a_i$ and $\beta_j = -b_j$. Apply two-way rounding to these numbers, when ordered by rows and by columns. The resulting integers $\bar{x}_{ij}$ and $\bar{\alpha}_i$ satisfy the condition $\sum_{i=1}^{k}(\sum_{j=1}^{n}\bar{x}_{ij} + \bar{\alpha}_i) = 0$ for all $k$, by $(*)$, hence $\sum_{j=1}^{n}\bar{x}_{ij} + \bar{\alpha}_i = 0$ for all $i$; it follows that $\lfloor a_i \rfloor = -\lceil -a_i \rceil \le -\bar{\alpha}_i = \sum_{j=1}^{n}\bar{x}_{ij} \le -\lfloor -a_i \rfloor = \lceil a_i \rceil$. Similarly $\lfloor b_j \rfloor \le \sum_{i=1}^{m}\bar{x}_{ij} \le \lceil b_j \rceil$ for all $j$. We also have $\sum_{i=1}^{m}\alpha_i + s = 0$; hence $\sum_{i=1}^{m}\bar{\alpha}_i + \bar{s} = 0$. The sum $\sum_{i=1}^{m}\sum_{j=1}^{n}\bar{x}_{ij}$ therefore equals $\bar{s}$, which is either $\lfloor s \rfloor$ or $\lceil s \rceil$. $\square$

**2. A sharper bound.** Notice that $(*)$ is equivalent to the conditions

$$|S_k - \overline{S}_k| < 1 \qquad \text{and} \qquad |\Sigma_k - \overline{\Sigma}_k| < 1\,, \qquad \text{for} \quad 0 \le k \le n\,,$$

since $\overline{S}_k$ and $\overline{\Sigma}_k$ are integers. Let us say that two-way rounding has discrepancy bounded by $\delta$ if $|S_k - \overline{S}_k| \le \delta$ and $|\Sigma_k - \overline{\Sigma}_k| \le \delta$ for all $k$. A slight extension of the construction in the lemma makes it possible to prove a stronger result.

THEOREM 2. *If $S_n = m$ is an integer, the sequence $(x_1, \ldots, x_n)$ can be two-way rounded with discrepancy bounded by $(2m+1)/(2m+2)$.*

*Proof.* We will prove that two-way rounding bounded by $\delta$ is possible for all $\delta > (2m+1)/(2m+2)$. Only finitely many roundings exist, so the stated result follows by taking the limit as $\delta$ decreases to $(2m+1)/(2m+2)$.

The proof uses a network like that of the lemma, but we omit certain arcs that would lead to discrepancies near 1. More precisely, if $\epsilon$ is any fixed positive number $< 1/(2m+2)$, we have

$$\begin{aligned} a_j &\to u_k \qquad \text{if} \quad [j-1+\epsilon \,..\, j-\epsilon) \cap [S_{k-1} \,..\, S_k) \ne \emptyset\,; \\ v_{\sigma k} &\to b_j \qquad \text{if} \quad [j-1+\epsilon \,..\, j-\epsilon) \cap [\Sigma_{k-1} \,..\, \Sigma_k) \ne \emptyset\,. \end{aligned}$$

We also allow these arcs to have infinite capacity. But the capacity of the "source" arcs $s \to a_j$, the "middle" arcs $u_k \to v_k$, and the "sink" arcs $b_j \to t$ remains 1.

The minimum cut in this reduced network has size $m$. For if any $m-1$ of the unit-capacity arcs are cut, we will prove that we can still connect $s$ to $t$. Suppose we remove $p$ source arcs, $q$ middle arcs, and $r$ sink arcs, where $p+q+r = m-1$. We send $1-2\epsilon$ units of flow from $s$ through each of the $m-p$ remaining source arcs. From every $a_j$ reached in this way, we send as many units of flow from $a_j \to u_k$ as the size of the interval $[j-1+\epsilon \,..\, j-\epsilon) \cap [S_{k-1} \,..\, S_k)$. Some of the flow now gets stuck, if $u_k$ is one of the $q$ vertices for which the arc $u_k \to v_k$ was removed. But at most $1-2\epsilon$ units flow into each $u_k$, so we still have at least $(m-p-q)(1-2\epsilon) = (r+1)(1-2\epsilon)$ units of flow arriving at $\{v_1, \ldots, v_n\}$. Now consider an "antiflow" of $1-2\epsilon$ units from $t$ back through each of the $m-r$ remaining sink arcs $b_j \to t$. From every such $b_j$ we send the antiflow back through $v_{\sigma k} \to b_j$ according to the size of $[j-1+\epsilon \,..\, j-\epsilon) \cap [\Sigma_{k-1} \,..\, \Sigma_k)$. In this way $(m-r)(1-2\epsilon)$ units of antiflow come from $t$ to $\{v_1, \ldots, v_n\}$. Each vertex $v_k$ contains at most $x_k$ units of flow and at most $x_k$ units of antiflow. We know that the total flow plus antiflow at $\{v_1, \ldots, v_n\}$ is at least $(r+1)(1-2\epsilon) + (m-r)(1-2\epsilon) = m+1-(2m+2)\epsilon > m = x_1 + \cdots + x_n$. Therefore some vertex $v_k$ must contain both flow and antiflow. And this establishes the desired link between $s$ and $t$.

Since $m$ is the size of a minimum cut and all capacities are integers, the network supports an integer flow of value $m$. Let $\bar{x}_k$ be the flow from $u_k$ to $v_k$; we will prove that $(\bar{x}_1, \ldots, \bar{x}_n)$ is a two-way rounding with discrepancy $< \delta = 1-\epsilon$. Note that

$$|\overline{S}_k - S_k| < 1-\epsilon \Leftrightarrow \lfloor S_k + \epsilon \rfloor \le \overline{S}_k \le \lceil S_k - \epsilon \rceil\,.$$

If $j = \lceil S_k - \epsilon \rceil$ we have $\overline{S}_k = \bar{x}_1 + \cdots + \bar{x}_k =$ flow into $\{u_1, \ldots, u_k\} \leq$ flow out of $\{a_1, \ldots, a_j\} = j$, because all arcs $a_i \to u_l$ for $l \leq k$ have $[i-1+\epsilon \mathinner{\ldotp\ldotp} i-\epsilon) \cap [S_{l-1}, S_l) \neq \emptyset$, hence $i - 1 + \epsilon < S_l$ and $i \leq \lceil S_l - \epsilon \rceil \leq j$. Similarly, if $j = \lfloor S_k + \epsilon \rfloor$ we have $\overline{S}_k \geq$ flow out of $\{a_1, \ldots, a_j\} = j$, because all arcs $a_i \to u_l$ for $i \leq j$ have $l \leq k$. (If $l > k$ we would have $S_{l-1} \geq S_k \geq j - \epsilon \geq i - \epsilon$, contradicting $S_{l-1} < i - \epsilon$.) A similar proof shows that $\lfloor \Sigma_k + \epsilon \rfloor \leq \overline{\Sigma}_k \leq \lceil \Sigma_k - \epsilon \rceil$. □

**3. A lower bound.** The bound of Theorem 2 is, in fact, best possible, in the sense that no better bound can be guaranteed as a function of $m$.

THEOREM 3. *For all positive integers $m$ there exists a sequence of real numbers $(x_1, \ldots, x_n)$ with sum $m$ and a permutation $\sigma$ of $\{1, \ldots, n\}$ that cannot be two-way rounded with discrepancy $< (2m+1)/(2m+2)$.*

*Proof.* Let $n = 2m + 2$ and $\epsilon = 1/n$. Define

$$x_1 = x_2 = x_3 = \epsilon\,; \qquad x_{m+3} = (2m-1)\epsilon\,;$$

$$x_{k+3} = 2\epsilon\,,\ x_{k+m+3} = 2m\epsilon\,, \quad \text{for} \quad 1 \leq k < m\,;$$

$$\sigma 1 = 2\,,\ \sigma 2 = 1\,,\ \sigma 3 = m+3\,,\ \sigma(2m+2) = 3\,;$$

$$\sigma(2k+2) = k+3\,,\ \sigma(2k+3) = k+m+3\,, \quad \text{for} \quad 1 \leq k < m\,.$$

For example, when $m = 4$ we have $(x_1, \ldots, x_{10}) = (.1, .1, .1, .2, .2, .2, .7, .8, .8, .8)$ and $(\sigma 1, \ldots, \sigma 10) = (2, 1, 7, 4, 8, 5, 9, 6, 10, 3)$. Hence

$$(S_1, \ldots, S_{10}) = (.1,\ .2,\ .3,\ .5,\ .7,\ .9,\ 1.6,\ 2.4,\ 3.2,\ 4.0)\,,$$

$$(\Sigma_1, \ldots, \Sigma_{10}) = (.1,\ .2,\ .9,\ 1.1,\ 1.9,\ 2.1,\ 2.9,\ 3.1,\ 3.9,\ 4.0)\,.$$

We will prove that this sequence and permutation cannot be two-way rounded with discrepancy less than $(2m+1)/(2m+2) = 0.9$; the same proof technique will work for any $m \geq 1$.

The main point is that whenever $S_k$ or $\Sigma_k$ has the form $l \pm 0.1$ where $l$ is an integer, it must be rounded to $l$ in order to keep the discrepancy small. This forces $\overline{S}_1 = \overline{\Sigma}_1 = 0$, $\overline{\Sigma}_3 = \overline{\Sigma}_4 = 1$, $\overline{\Sigma}_5 = \overline{\Sigma}_6 = 2$, $\overline{\Sigma}_7 = \overline{\Sigma}_8 = 3$, $\overline{\Sigma}_9 = 4$, hence $\bar{x}_1 = \bar{x}_2 = \bar{x}_3 = \bar{x}_4 = \bar{x}_5 = \bar{x}_6 = 0$. But then $\overline{S}_6 = \bar{x}_1 + \cdots + \bar{x}_6 = 0$ differs by 0.9 from $S_6$. □

**4. A uniform bound.** Although Theorem 3 proves that Theorem 2 is "optimal," we can do still better if $m$ is greater than $\frac{1}{2}n$, because we can replace each $x_k$ by $1 - x_k$. This replaces $m$ by $n - m$, and the bound on discrepancy decreases to $(2n - 2m + 1)/(2n - 2m + 2)$. Then we can restore the original $x_k$ and change $\bar{x}_k$ to $1 - \bar{x}_k$. This computation preserves $|S_k - \overline{S}_k|$ and $|\Sigma_k - \overline{\Sigma}_k|$, so it preserves the discrepancy.

Further improvement is also possible when $m = \lfloor n/2 \rfloor$, if we look at the construction closely. The following theorem gives a uniform bound in terms of $n$, without any assumption about the value of $x_1 + \cdots + x_n$.

THEOREM 4. *Any sequence $(x_1, \ldots, x_n)$ and permutation $(\sigma 1, \ldots, \sigma n)$ can be two-way rounded with discrepancy bounded by $n/(n+1)$.*

*Proof.* We will show in fact that the discrepancy can always be bounded by $(n - 1)/n$, when $x_1 + \cdots + x_n = m$ is an integer. The general case follows from this special case if we set $x_{n+1} = \lceil S_n \rceil - S_n$ and increase $n$ by 1.

If $2m + 2 \leq n$ or $2n - 2m + 2 \leq n$, the result follows from Theorem 2 and possible complementation. Therefore we need only show that a discrepancy of at most $(n-1)/n$ is achievable when $m = \lfloor n/2 \rfloor$.

Consider first the case $n = 2m+1$. We use the network in the proof of Theorem 2, but now we allow $\epsilon$ to be any number $< 1/n$. Suppose, as in the previous proof, that we can disconnect $s$ from $t$ by deleting $p$ source arcs, $q$ middle arcs, and $r$ sink arcs, where $p + q + r = m - 1$. Let $q$ be minimum over all such ways to disconnect the network. We construct flows and antiflows as before, and we say that $x_k$ is *green* if $v_k$ contains positive flow and *red* if $v_k$ contains positive antiflow. No $x_k$ is both green and red, since there is no path from $s$ to $t$. The previous proof showed that there are at least $(r+1)(1-2\epsilon)$ units of green flow and $(m-r)(1-2\epsilon)$ units of red flow, hence there are at least $m+1-(2m+2)\epsilon$ units of flow altogether. If we can raise this lower bound by $\epsilon$, we will have a contradiction, because $m+1-(2m+1)\epsilon > m$.

Suppose $q > 0$, and let $u_k \to v_k$ be a middle arc that was deleted. At most two arcs emanate from $v_k$ in the network. Since $q$ is minimum, there must in fact be two; otherwise we could restore $u_k \to v_k$ and delete a nonmiddle arc. The two arcs from $v_k$ must be consecutive, from $v_k \to b_j$ and $v_k \to b_{j+1}$ for some $j$. Furthermore the arcs $b_j \to t$ and $b_{j+1} \to t$ have not been cut. If $k = \sigma l$ we have $\Sigma_{l-1} < j - \epsilon$ and $\Sigma_l > j + \epsilon$. Our lower bound on antiflow can now be raised by $2\epsilon$, because it was based on the weak assumption that no antiflow runs back from $[j-\epsilon \mathinner{.\,.} j+\epsilon)$. This improved lower bound leads to a contradiction; hence $q = 0$.

Divide the interval $[0 \mathinner{.\,.} m)$ into $3m$ regions, namely "tiny left" regions of the form $[j-1 \mathinner{.\,.} j-1+\epsilon)$, "inner" regions of the form $[j-1+\epsilon \mathinner{.\,.} j-\epsilon)$, and "tiny right" regions of the form $[j-\epsilon \mathinner{.\,.} j)$, for $1 \le j \le m$. If we color the points of $[S_{k-1} \mathinner{.\,.} S_k)$ with the color of $x_k$, our lower bound $(r+1)(1-2\epsilon)$ for green flow was essentially obtained by noting that $m-p = r+1$ of the inner regions are purely green. Similarly, if we color the ponts of $[\Sigma_{k-1} \mathinner{.\,.} \Sigma_k)$ with the color of $x_{\sigma k}$, our lower bound for red flow was obtained by noting that $m-r = p+1$ inner regions in this second coloring are purely red. Notice that there is complete symmetry between red and green, because we can invert the network and replace $\sigma$ by $\sigma^{-1}$.

Call an element $x_k$ *large* if it exceeds $1-\epsilon$. If any $x_k$ is large, the interval $[S_{k-1} \mathinner{.\,.} S_k)$ occupies more than $\epsilon$ units outside of an inner region; this allows us to raise the lower bound by $\epsilon$ and obtain a contradiction. Therefore no element is large. It follows that no element $x_k$ can intersect more than 2 tiny regions, when $x_k$ is placed in correspondence with $[S_{k-1} \mathinner{.\,.} S_k)$ or with $[\Sigma_{k-1} \mathinner{.\,.} \Sigma_k)$.

Let's look now at the $2m$ tiny regions. Each of them must contain at least some red in the first coloring; otherwise we would have at least $(p+1)(1-2\epsilon)$ red units packed into at most $2m-1$ tiny regions and $p$ inner regions, hence $(p+1)(1-2\epsilon) \le (2m-1)\epsilon + p(1-2\epsilon)$, contradicting $\epsilon < 1/n$. This means there must be at least $m+1$ red elements $x_k$, since no red element is large and since $m$ non-large red intervals can intersect all the tiny regions only if they also cover all the inner regions (at least one of which is green). Similarly, there must be at least $m+1$ green elements. But this is impossible, since there are only $2m+1$ elements altogether. Therefore the network has minimum cut size $m$, and the rest of the proof of Theorem 2 goes through as before.

Now suppose $n = 2m$. Then we can carry out a similar argument, but we need to raise the lower bound by $2\epsilon$. Again we can assume that $q = 0$. We can also show without difficulty that there cannot be *two* large elements. When $n = 2m$ the argument given above shows that at least $2m-1$ of the tiny regions must contain some red, in the first coloring.

Suppose there are only $m-1$ red elements. Then, in the first coloring, $m-2$ of them intersect 2 tiny intervals and the other is large and intersects 3; we have raised the red lower bound by $\epsilon$. But $(p+1)(1-2\epsilon)+\epsilon$ red units cannot be packed into $2m-1$ tiny regions and $p$ inner regions, because $(p+1)(1-2\epsilon)+\epsilon > (n-1)\epsilon + p(1-2\epsilon)$.

A symmetrical argument shows that there cannot be only $m-1$ green elements. Therefore exactly $m$ elements are red and exactly $m$ are green. Suppose no element is large. Then we have at least one purely green tiny interval in the first coloring and at least one purely red tiny interval in the second—another contradiction. Thus, we may assume that there is one large red element and that the $2m$ tiny intervals in the first coloring contain a total of less than $\epsilon$ units of green. In particular, each of them contains some red. Either the first interval $[0 \mathinner{..} \epsilon)$ or the last interval $[m-\epsilon \mathinner{..} m)$ is intersected by a nonlarge red element, which intersects at most $\epsilon$ units of space in tiny intervals. The other $m-1$ red elements intersect at most $2\epsilon$ units of tiny space each, so at most $(2m-1)\epsilon$ such units are red. This final contradiction completes the proof. $\square$

The result of Theorem 4 is best possible, because we can easily prove (as in Theorem 3) that the values

$$x_1 = \frac{1}{n+1}, \qquad x_k = \begin{cases} (n-1)/(n+1), & k \text{ even}, 2 \le k \le n, \\ 2/(n+1), & k \text{ odd}, 3 \le k \le n, \end{cases}$$

and a "shuffle" permutation that begins

$$\sigma k = \begin{cases} 2k-1 & \text{for } 1 \le 2k-1 \le n, \quad n \text{ odd}, \\ 2k & \text{for } 1 \le 2k \le n, \quad n \text{ even}, \end{cases}$$

cannot be two-way rounded with discrepancy less than $n/(n+1)$.

**5. An algorithm.** So far we have discussed only worst-case bounds. But a particular two-way rounding problem, defined by values $(x_1, \ldots, x_n)$ and a permutation $(\sigma 1, \ldots, \sigma n)$, will usually be solvable with smaller discrepancy than that guaranteed by Theorems 2 and 4. A closer look at the construction of Theorem 2 leads to an efficient algorithm that finds the best possible discrepancy in any given case.

THEOREM 5. *Let $\epsilon$ be any positive number. There exists a solution with discrepancy less than $1-\epsilon$ to a given two-way rounding problem if and only if the network constructed in the proof of Theorem 2 supports an integer flow of value $m$.*

*Proof.* The final paragraph in the proof of Theorem 2 demonstrates the "if" half. Conversely, suppose $\bar{x}_1, \ldots, \bar{x}_n$ is a solution with discrepancy $< 1-\epsilon$. If $\bar{x}_k = 1$, let $j = \bar{S}_k$. Then $j-1 = \bar{S}_{k-1}$, so the condition $|\bar{S}_{k-1} - S_{k-1}| < 1-\epsilon$ implies $S_{k-1} < j-\epsilon$. Also $|\bar{S}_k - S_k| < 1-\epsilon$ implies $S_k > j-1+\epsilon$. Therefore there is an arc $a_j \to u_k$. Similarly, there is an arc $v_{\sigma k} \to b_j$ when $\bar{x}_{\sigma k} = 1$ and $j = \bar{\Sigma}_k$. Thus the network supports an integer flow of value $m$. $\square$

In other words, the optimum discrepancy $\delta = 1-\epsilon$ is obtained when $\epsilon$ is just large enough to reduce the network to the point where no $m$-unit flow can be sustained, if $\delta \ge \frac{1}{2}$. We can in fact find an optimum rounding as follows. Let

$$f(j,k) = \min(j - S_{k-1}, S_k - j + 1)$$

be the "desirability" of the arc $a_j \to u_k$, and

$$g(j, \sigma k) = \min(j - \Sigma_{k-1}, \Sigma_k - j + 1)$$

the desirability of $v_{\sigma k} \to b_j$. (Thus the arcs $a_j \to u_k$, $v_{\sigma k} \to b_j$ are included in the network of Theorem 2 if and only if their desirability is greater than $\epsilon$.) Sort these arcs by desirability and add them one by one to the initial arcs $\{s \to a_j, u_k \to v_k, b_j \to t\}$

TABLE 1.
*Empirical optimum discrepancies.*

| | $m = 1$ | $m = 2$ | $m = \lfloor \lg n \rfloor$ | $m = \lfloor \sqrt{n} \rfloor$ | $m = \frac{1}{2}n$ |
|---|---|---|---|---|---|
| $n = 10$ | $.566 \pm .06$ | $.619 \pm .07$ | $.627 \pm .07$ | $.627 \pm .07$ | $.622 \pm .08$ |
| $n = 100$ | $.537 \pm .02$ | $.575 \pm .03$ | $.664 \pm .03$ | $.710 \pm .03$ | $.759 \pm .02$ |
| $n = 1000$ | $.513 \pm .007$ | $.527 \pm .01$ | $.582 \pm .01$ | $.662 \pm .02$ | $.794 \pm .02$ |
| $n = 10000$ | $.504 \pm .002$ | $.509 \pm .003$ | $.535 \pm .005$ | $.612 \pm .01$ | $.818 \pm .01$ |
| $n = 100000$ | $.502 \pm .001$ | $.503 \pm .001$ | $.513 \pm .002$ | $.570 \pm .005$ | $.838 \pm .007$ |

until an integer flow of $m$ units is possible. Then let $\bar{x}_k$ be the flow in $u_k \to v_k$, for all $k$; this flow has discrepancy equal to 1 minus the desirability of the last arc added, and no smaller discrepancy is possible.

Notice that the arc $a_j \to u_k$ has desirability $> \frac{1}{2}$ if and only if $S_{k-1} < j - \frac{1}{2} < S_k$, so at most $m$ such arcs are present. If all $x_k$ lie between 0 and 1, at most $m + n - 1$ arcs of the form $a_j \to u_k$ will have positive desirability, since both $a_{j-1} \to u_k$ and $a_j \to u_k$ will be desirable if and only if $S_{k-1} < j < S_k$.

The following simple algorithm turns out to be quite efficient, assuming that $m \le \frac{1}{2}n$: Begin with the network consisting of arcs $\{s \to a_j, u_k \to v_k, b_j \to t\}$ for $1 \le j \le m$ and $1 \le k \le n$, plus any additional arcs of desirability $> \frac{1}{2}$. Call an arc $a_j \to u_k$ or $v_{\sigma k} \to b_j$ "special" if its desirability lies between $1/\min(2m+2, n)$ and $\frac{1}{2}$, inclusive; fewer than $2m+2n$ arcs are special. Then, for $j = 1, \ldots, m$, send one unit of flow from $a_j$ to $t$ along an "augmenting path," using the well-known algorithm of Ford and Fulkerson [2, pp. 17–19] but specialized for unit-capacity arcs. In other words, construct a breadth-first search tree from $a_j$ until encountering $t$; then choose a path from $a_j$ to $t$ and reverse the orientation of all arcs on that path. If $t$ is not reachable from $a_j$, add special arcs to the network, in order of decreasing desirability, until $t$ is reachable.

## 6. Computational experience.

The running time of this algorithm is bounded by $O(mn)$ steps, but in practice it runs much faster on random data. For example, Tables 1 and 2 show the results of various tests when the input permutation $\sigma$ is random and when the values $(x_1, \ldots, x_n)$ are selected as follows. Let $y_1, \ldots, y_n$ be independent uniform integers in the range $1 \le y_k \le N$, where $N$ is a large integer (chosen so that arithmetic computations will not exceed 31 bits). Increase one or more of the $y$'s by 1, if necessary, until $y_1 + \cdots + y_n$ is a multiple of $m$; then set $x_k = y_k/d$, where $d = (y_1 + \cdots + y_n)/m$. Reject $(x_1, \ldots, x_n)$ and start over, if some $x_k \ge 1$. (In practice, rejection occurs about half the time when $m = \frac{1}{2}n$, but almost never when $m \ll \frac{1}{2}n$.)

Table 1 shows the optimum discrepancies found, and Table 2 shows the running time in memory references or "mems" [6, pp. 464–465] divided by $n$. All entries in these tables are given in the form $\mu \pm \sigma$, where $\mu$ is the sample mean and $\sigma$ is an estimate of the standard deviation; more precisely, $\sigma$ is the square root of an unbiased estimate of the variance. The number of test runs $t(n)$ for each experiment was $10^6/n$; thus, $10^5$ runs were made for each $m$ when $n = 10$, but only 10 runs were made for each $m$ when $n = 10^5$. The actual confidence interval for the tabulated $\mu$ values is therefore approximately $2\sigma/\sqrt{t(n)} = .002\sigma\sqrt{n}$.

Notice that when $m \ll n$, the optimum discrepancy is nearly $\frac{1}{2}$. Indeed, this is obvious on intuitive grounds: When $n$ is large, approximately $\epsilon n$ values of $k$ will have $S_k$ within $\frac{1}{2}\epsilon$ of $\{\frac{1}{2}, \frac{3}{2}, \ldots, m - \frac{1}{2}\}$, and approximately $\epsilon^2 n$ will also have equally good values $\Sigma_{\sigma^{-1}k}$. So we are essentially looking for a perfect matching in a bipartite graph

TABLE 2.
*Empirical running time, in mems/n.*

| | $m = 1$ | $m = 2$ | $m = \lfloor \lg n \rfloor$ | $m = \lfloor \sqrt{n} \rfloor$ | $m = \frac{1}{2}n$ |
|---|---|---|---|---|---|
| $n = 10$ | $10 \pm 4$ | $19 \pm 6$ | $27 \pm 8$ | $27 \pm 8$ | $37 \pm 11$ |
| $n = 100$ | $2.9 \pm 1.3$ | $6 \pm 2$ | $18 \pm 5$ | $29 \pm 7$ | $76 \pm 15$ |
| $n = 1000$ | $0.9 \pm 0.5$ | $1.9 \pm 0.7$ | $8.5 \pm 2.2$ | $25 \pm 6$ | $152 \pm 32$ |
| $n = 10000$ | $0.3 \pm 0.2$ | $0.6 \pm 0.2$ | $3.6 \pm 0.8$ | $22 \pm 7$ | $289 \pm 49$ |
| $n = 100000$ | $0.1 \pm 0.1$ | $0.2 \pm 0.1$ | $1.4 \pm 0.4$ | $17 \pm 4$ | $540 \pm 72$ |

with $m$ vertices in each part and $\epsilon^2 n$ edges. For fixed $m$ as $n \to \infty$, the matching will exist when $\epsilon^2 n$ is sufficiently large, hence the mean optimum discrepancy is $\frac{1}{2}+O(n^{-\frac{1}{2}})$.

However, the behavior of the mean optimum discrepancy when $m = \frac{1}{2}n$ is not clear. It appears to approach 1, but quite slowly, perhaps as $1 - c/\log n$.

When $n$ is fixed and $m$ varies, the mean optimum discrepancy is not maximized when $m = \frac{1}{2}n$. For example, when $n = 10$, Table 1 shows that it is .622 when $m = 5$ but .627 when $m = 3$.

The running times shown in Table 2 do not include the work of constructing the network or sorting the special arcs by desirability. Those operations are easily analyzed, and in practice they take $am + bn$ steps for some constants $a$ and $b$, because a straightforward bucket sort is satisfactory for this application. Therefore only the running time of the subsequent flow calculations is of interest.

The average running time to compute the flows appears to be $o(n)$ when $m \le \sqrt{n}$, and approximately proportional to $n^{1.3}$ when $m = \frac{1}{2}n$. So it is much less than the obvious upper bound $mn$ of the Ford–Fulkerson scheme. The author tried to obtain still faster results by using more sophisticated max-flow algorithms, but these "improved" algorithms actually turned out to run more than an order of magnitude slower.

For example, the algorithm of Dinitz, as improved by Karzanov and others, seems at first to be especially well suited to this application because the network of Theorem 2 is "simple" in the sense discussed by Papadimitriou and Steiglitz [9, pp. 212–214]. Every internal vertex has in-degree 1 or out-degree 1, hence edge-disjoint paths are vertex disjoint and the running time with unit-capacity arcs is $O(\,|V|^{1/2}\,|A|\,) = O(n^{3/2})$. Using binary search to find the optimum number of special arcs gives us a guaranteed worst-case performance of $O\bigl(\min(m, n^{1/2})n \log n\bigr)$. Unfortunately, in practice the performance of that algorithm actually matches this worst-case estimate, even on random data. For example, when $m = \frac{1}{2}n$ the observed running time in mems/$n$ was $15284 \pm 2455$ when $n = 10^4$, compared to $289 \pm 49$ by the simple algorithm. Each flow calculation consumed more than $1000n$ mems, and binary search required $\lceil \lg 2n \rceil = 14$ flow calculations to be carried out.

When modern preflow push/relabel algorithms are specialized to unit-capacity networks of the type considered here, they behave essentially like the Dinitz algorithm and are no easier to implement (see Goldberg, Plotkin, and Vaidya [4]). Such algorithms do allow networks to change dynamically by adding arcs from $s$ and/or deleting arcs to $t$ (see Gallo, Grigoriadis, and Tarjan [3]); however, our application requires adding or deleting special arcs in the *middle* of the network, so the techniques of [3] do not apply. Thus the simple Ford–Fulkerson algorithm seems to be a clear winner for this application, in spite of a lack of performance guarantees.

## 7. Comments and open problems.

How complex can the networks of Theorem 2 be? If we have any bipartite graph with $m$ vertices in each part and with $n$ edges, and if every edge can be extended to a perfect matching, then we can find real numbers $(x_1, \ldots, x_n)$ in the range $0 < x_k \le 1$ and a permutation $(\sigma 1, \ldots, \sigma n)$ such that $x_1 + \cdots + x_n = m$ and the two-way roundings are in one-to-one correspondence with the perfect matchings of the given graph. We can take $(x_1, \ldots, x_n) = t_1\alpha_1 + \cdots + t_n\alpha_n$ where $t_1 + \cdots + t_n = 1$ and $\alpha_k$ is the characteristic vector of a perfect matching that uses edge $k$. The sum of $x_k$ over all the edges touching any vertex is 1. Represent an edge from $u$ to $v$ by the ordered pair $(u, v)$ and label the edges $1, \ldots, n$ in lexicographic order of these pairs; then define the permutation $\sigma 1, \ldots, \sigma n$ by lexicographic order of the dual pairs $(v, u)$. It follows that if $k$ is the final edge for vertex $j$ in the first part, we have $S_k = j$, and if $\sigma k$ is the final edge for vertex $j$ in the second part, we have $\Sigma_k = j$. The correspondence between matchings and roundings is now evident.

This construction shows that the networks arising in Theorem 2 are general enough to mimic the networks that arise in bipartite matching problems, but only when the bipartite graphs contain no unmatchable edges; the corollary preceding Theorem 2 shows that the latter restriction cannot be removed. This restriction on network complexity might account for the excellent performance we obtain with the simple Ford–Fulkerson algorithm.

If the capacity constraint on $u_k \to v_k$ is removed, our network becomes equivalent to a network for bipartite matching, in which we want to match $\{a_1, \ldots, a_m\}$ to $\{b_1, \ldots, b_m\}$ through edges $a_j — b_{j'}$ whenever $a_j \to u_k$ and $v_k \to b_{j'}$. The problem of finding the *best* such match, when the edge $a_j — b_{j'}$ is ranked by the minimum of the desirabilities $f(j, k)$ and $g(j', k)$, is then a *bottleneck assignment problem* [1], [2]. (It is an open question whether there is a nice way to characterize all bottleneck assignment problems that arise from two-way rounding problems in this manner.)

The problem of optimum two-way rounding is, however, more general than the bottleneck assignment problem, because the unit capacity constraint on $u_k \to v_k$ is significant. Consider, for example, the case $n = 7$, $m = 3$, $(x_1, \ldots, x_7) = \frac{1}{28}(8, 8, 24, 11, 11, 11, 11)$, $(\sigma 1, \ldots, \sigma 7) = (2, 1, 3, 5, 4, 7, 6)$. Then $(S_1, \ldots, S_7) = (\Sigma_1, \ldots, \Sigma_7) = \frac{1}{28}(8, 16, 40, 51, 62, 73, 84)$ and the arcs $\{a_j \to u_k, v_k \to b_j\}$ ranked by desirability are

| | |
|---|---|
| $a_3 \to u_6,\ v_7 \to b_3$ | desirability $= \min\left(\frac{22}{28}, \frac{17}{28}\right) = \frac{17}{28}$ |
| $a_1 \to u_2,\ v_1 \to b_1,\ a_2 \to u_4,\ v_5 \to b_2$ | desirability $\frac{16}{28}$ |
| $a_1 \to u_3,\ v_3 \to b_1,\ a_2 \to u_3,\ v_3 \to b_2$ | desirability $\frac{12}{28}$ |
| $a_3 \to u_7,\ v_6 \to b_3$ | desirability $\frac{11}{28}$ |
| $a_1 \to u_1,\ v_2 \to b_1$ | desirability $\frac{8}{28}$ |
| $a_3 \to u_5,\ v_4 \to b_3$ | desirability $\frac{6}{28}$ |
| $a_2 \to u_5,\ v_4 \to b_2$ | desirability $\frac{5}{28}$ |

Thus the edges $a_j — b_{j'}$ ranked by desirability are

| | |
|---|---|
| $a_1 — b_1,\ a_1 — b_2,\ a_2 — b_1,\ a_2 — b_2$ | $\left(\frac{12}{28}\text{ via } u_3, v_3\right)$ |
| $a_3 — b_3$ | $\left(\frac{11}{28}\text{ via } u_6, v_6 \text{ or } u_7, v_7\right)$ |
| $a_1 — b_1$ | $\left(\frac{8}{28}\text{ via } u_1, v_1 \text{ or } u_2, v_2\right)$ |
| $a_2 — b_3,\ a_3 — b_2$ | $\left(\frac{6}{28}\text{ via } u_4, v_4 \text{ or } u_5, v_5\right)$ |
| $a_2 — b_2$ | $\left(\frac{5}{28}\text{ via } u_4, v_4 \text{ or } u_5, v_5\right)$ |

The bottleneck assignment problem is solved by matching $a_1 — b_1$, $a_2 — b_2$, and $a_3 — b_3$ with desirability $\min\left(\frac{12}{28}, \frac{12}{28}, \frac{11}{28}\right) = \frac{11}{28}$. But this matching does not correspond to a valid two-way rounding because it uses the intermediate arc $u_3 \to v_3$

twice; it rounds $x_3$ to 2 and $x_6$ (or $x_7$) to 1. The optimum two-way rounding uses another route from $a_1$ to $b_1$ and has desirability $\min\left(\frac{8}{28}, \frac{12}{28}, \frac{11}{28}\right) = \frac{8}{28}$, discrepancy $1 - \frac{8}{28} = \frac{20}{28}$; it rounds $x_1$ (or $x_2$), $x_3$, and $x_6$ (or $x_7$) to 1, the other $x$'s to 0.

In closing, we note that a conjecture of József Beck [7], [11] remains a fascinating open problem: Is there a constant $K$ such that three-way rounding is always possible with discrepancy at most $K$? (In three-way rounding the partial sums are supposed to be well approximated with respect to a third permutation $(\tau 1, \ldots, \tau n)$, in addition to $(1, \ldots, n)$ and $(\sigma 1, \ldots, \sigma n)$.) It suffices [7], [11] to prove this when $x_k = \frac{1}{2}$ for all $k$.

Can any of the methods of this paper be extended to find better bounds on the discrepancy of arbitrary set systems (or at least of set systems more general than those for two-way rounding), in the sense of [11]?

**Acknowledgments.** I thank Joel Spencer for proposing the problem and for showing me a simple construction that forces discrepancy $n/(n+1)$. Thanks also to Noga Alon, Svante Janson, and Serge Plotkin for several stimulating discussions as I was working out the solution described above. Shortly after I had proved Theorems 2–4, a somewhat similar construction was found independently by Jacek Ossowski, who described it in terms of common systems of distinct representatives instead of network flows; see §9.2 in [8].

## REFERENCES

[1] J. EDMONDS AND D. R. FULKERSON, *Bottleneck extrema*, J. Combin. Theory, 8 (1970), pp. 299–306.

[2] L. R. FORD, JR., AND D. R. FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[3] 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.

[4] A. V. GOLDBERG, S. A. PLOTKIN, AND P. M. VAIDYA, *Sublinear-time parallel algorithms for matching and related problems*, J. Algorithms, 14 (1993), pp. 180–213.

[5] R. L. GRAHAM, D E. KNUTH, AND O. PATASHNIK, *Concrete Mathematics*, Addison–Wesley, Reading, MA, 1989.

[6] D. E. KNUTH, *The Stanford GraphBase*, ACM Press, New York, 1994.

[7] L. LOVÁSZ, J. SPENCER, AND K. VESZTERGOMBI, *Discrepancy of set-systems and matrices*, European J. Combin., 7 (1986), pp. 151–160.

[8] L. MIRSKY, *Transversal Theory*, Academic Press, New York, 1971.

[9] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization*, Prentice–Hall, Englewood Cliffs, NJ, 1982.

[10] A. SCHRIJVER, *Theory of Linear and Integer Programming*, Wiley, London, New York, 1986.

[11] J. SPENCER, *Ten Lectures on the Probabilistic Method*, CBMS-NSF Regional Conference Series in Applied Mathematics, 52, Society for Industrial and Applied Mathematics, Philadelphia, 1987, pp. 37–44.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 291–321, May 1995

# DEGREE-CONSTRAINED NETWORK SPANNERS WITH NONCONSTANT DELAY*

ARTHUR L. LIESTMAN† AND THOMAS C. SHERMER†

**Abstract.** A spanning subgraph $S = (V, E')$ of a connected simple graph $G = (V, E)$ is a $f(x)$-*spanner* if for any pair of nodes $u$ and $v$, $d_S(u, v) \leq f(d_G(u, v))$ where $d_G$ and $d_S$ are the usual distance functions in graphs $G$ and $S$, respectively. The *delay* of the $f(x)$-spanner is $f(x) - x$. In this paper $(2.5\sqrt{(3x+6)/4} + 6 + x)$-spanners for two-dimensional grids with maximum degree 3 are found and it is proven that the delay of these spanners is within a constant factor of optimal. A $(\frac{1}{k}x + k + 8 - \frac{7}{k} + x)$-spanner of the X-tree with maximum degree 3 is described, and it is proven that the delay of this spanner is within a constant factor of optimal. In addition, a $(2 + x)$-spanner of the pyramid with maximum degree 6 and a $(\frac{1}{k}x + k + 8 - \frac{7}{k} + x)$-spanner of the pyramid with maximum degree 5 are described, and it is proven that the delay of the latter spanner is within a constant factor of optimal.

**Key words.** grid, pyramid, X-tree, spanner, parallel network

**AMS subject classifications.** 68M10, 05C99, 90B12

## 1. Introduction and definitions.

There are several popular topologies used for constructing parallel computers. Our goal is to determine substructures of such topologies with smaller maximum degree. We require also that these substructures, called *spanners*, have the property that the distance between two vertices in the substructure is not significantly larger than the corresponding distance in the original structure. In this paper we consider spanners of the two-dimensional grid that have maximum degree 3, spanners of the X-tree with maximum degree 3, and spanners of the pyramid with maximum degrees 5 and 6.

Spanners were introduced by Peleg and Ullman [7] who used these structures for efficient simulation of synchronous networks on asynchronous networks. A second motivation for studying spanners is their possible use as network topologies. A spanner can be used as a substitute for a desired network, often giving a much sparser network with a similar structure and only slightly larger communication costs. A third motivation is to provide a relatively inexpensive way of improving the quality of a network. For example, if one wished to construct a network using specialized links (perhaps links with higher bandwidth or higher reliability), an alternative would be to construct a spanner with these links and to use normal connections for the remaining links in the network.

A network is represented by a connected simple graph $G = (V, E)$. We use the notation $d_G(u, v)$ to denote the distance between vertices $u$ and $v$ in graph $G$. The subscript $G$ may be omitted if it is clear from context.

Let $f(x)$ be a function from the nonnegative integers to the nonnegative reals. A spanning subgraph $S$ of a connected simple graph $G$ is called a $f(x)$-*spanner* if for any pair of nodes $u$ and $v$ in $G$, $d_S(u, v) \leq f(d_G(u, v))$ [4]. This definition has natural generalizations for edge-weighted graphs, digraphs, and hypergraphs. We call $d_S(u, v) - d_G(u, v)$ the *delay between vertices $u$ and $v$ in $S$*, denoted $d'_S(u, v)$. For an $f(x)$-spanner $S$, we let $f'(x) = f(x) - x$ and refer to $f'(x)$ as the *delay* of the spanner.

* Received by the editors July 7, 1993; accepted for publication (in revised form) January 31, 1994. This research was supported by Natural Sciences and Engineering Research Council of Canada grants OGP0001734 and 0GP0046218.

† School of Computing Science, Simon Fraser University, Burnaby, Britsh Columbia V5A 1S6, Canada.

Note that $f'(x)$ is actually an upper bound on the maximum delay in $S$ between any pair of vertices at distance $x$ in $G$.

It may be possible to express the delay $f'(x)$ in several ways. In particular, any spanner $S$ of a finite graph $G$ is an $(x+c)$-spanner where $c$ is the maximum delay between any pair of vertices in $S$. A more careful analysis of $S$ may reveal a closer relationship between the distance in $G$ and the delay in $S$. For example, the $(x+c)$-spanner mentioned above may also be determined to be a $2x$-spanner. In general, we prefer to express $f(x)$ in a manner that bounds the delay as clearly as possible.

Peleg and Schäffer [6] investigated the existence and constructability of sparse spanners for various classes of graphs. They showed that to determine for a given graph $G$ and integer $m \geq 1$ whether $G$ has a $2x$-spanner with $m$ or fewer edges is NP-complete. Cai [1] showed that, for any integer $t \geq 2$, to determine for a given graph $G$ and integer $m \geq 1$ whether $G$ has a $tx$-spanner with $m$ or fewer edges is NP-complete. Degree-constrained spanners were studied by Richards and Liestman [8] (for pyramids) and by Liestman and Shermer [3] (for multidimensional grids). In [4], we provided the more general formulation of spanners and initiated the study of $(t+x)$-spanners, giving constructions for low-degree $(t+x)$-spanners of hypercubes. We showed [5] that for any integer $t \geq 1$, to determine for a given graph $G$ and integer $m$ whether $G$ has a $(t+x)$-spanner with $m$ or fewer edges is NP-complete. In the same paper, we also described how to construct low-degree $(t+x)$-spanners for X-trees, pyramids, and two-dimensional grids (both finite and infinite). Other related work is discussed in the recent paper of Cai and Corneil [2] and the survey of Soares [9].

In this paper, we return to the study of degree-constrained spanners. Section 2 details our results on spanners of two-dimensional grids. We begin by proving a lower bound of $\sqrt{\frac{x}{2}}$ on the delay of a maximum degree 3 spanner of the grid in §2.1. In §2.2, we describe the construction of $(2.5\sqrt{(3x+6)/4}+6+x)$-spanners for two-dimensional grids with maximum degree 3. The delay of these spanners is within a constant factor of the lower bound.

Section 3 contains our results on spanners of X-trees and pyramids. In §3.2, we prove that the delay of a maximum degree 3 spanner of the X-tree must be at least linear in $x$. In §3.3, we describe the construction of a $(\frac{1}{k}x+k+8-\frac{7}{k}+x)$-spanner for X-trees with maximum degree 3. In §3.5, we show how to construct a $(2+x)$-spanner for pyramids with maximum degree 6. In §3.6, we prove that the delay of a maximum degree 5 spanner of the pyramid must be at least linear in $x$. In §3.7, we present a construction of a $(\frac{1}{k}x+k+8-\frac{7}{k}+x)$-spanner for pyramids with maximum degree 5.

## 2. Spanners of two-dimensional grids.

A grid $G_{n_1,n_2}$, where $n_1, n_2 \geq 2$, has the vertex set $V = \{(i_1,i_2) | 1 \leq i_1 \leq n_1, 1 \leq i_2 \leq n_2\}$. Its edges are between pairs of vertices whose labels differ by 1 in exactly one position, that is, vertex $(i_1,i_2)$ is connected to vertex $(i_1, i_2+1)$ for all $i_1$ and $1 \leq i_2 < n_2$ and vertex $(i_1,i_2)$ is connected to vertex $(i_1+1, i_2)$ for all $i_2$ and $1 \leq i_1 < n_1$. In order to simplify some of the notation and terminology, we assume the natural embedding of these grids in the plane.

### 2.1. Lower bound on delay.

In this section we show that any maximum degree 3 spanner of a sufficiently large finite grid must have delay at least $\sqrt{\frac{x}{2}}$.

THEOREM 2.1. *Any spanner $S$ with maximum degree 3 of any grid $G_{n_1,n_2}$ with $n_1, n_2 \geq 2k^2+1$ for any even $k \geq 2$ has a pair of vertices $u$ and $v$ such that $d_G(u,v) = 2k^2$ and $d'_S(u,v) \geq \sqrt{d_G(u,v)/2} = k$.*

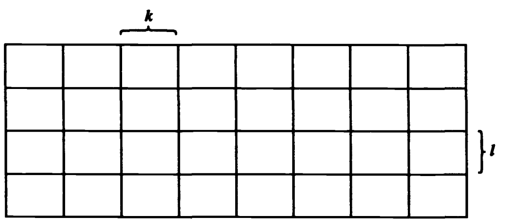


FIG. 1. *A highway spanner.*

*Proof.* Let $S$ be any spanner of $G = G_{n_1,n_2}$ with maximum degree 3. Consider the paths in $G$ and in $S$ between pairs of vertices $(ki+\frac{k}{2}+1,1)$ and $(ki+\frac{k}{2}+1,2k^2+1)$ where $0 \leq i \leq 2k-1$. In $G$, the shortest path between such a pair of vertices contains exactly $2k^2$ vertical edges. In $S$, any path between such a pair of vertices $u$ and $v$ must contain at least $2k^2$ vertical edges and, possibly, some horizontal edges. Thus, the delay between $u$ and $v$ in $S$ is at least as large as the number of horizontal edges in the shortest path between $u$ and $v$ in $S$. This implies that the shortest path between any such pair must stay in the subgrid with $x_1$-coordinates between $ki+2$ and $ki+k$, otherwise the number of horizontal edges (and thus the delay between these vertices) is at least $k$ and we are done.

Similarly, each shortest path in $S$ between a pair of vertices $(1, kj + \frac{k}{2} + 1)$ and $(2k^2 + 1, kj + \frac{k}{2} + 1)$ where $0 \leq j \leq 2k - 1$ must have delay at least as large as the number of vertical edges and thus stay in the subgrid with $x_2$-coordinates between $kj + 2$ and $kj + k$.

Consider the subgrid with $x_1$-coordinates between $ki + 2$ and $ki + k$ and $x_2$-coordinates between $kj + 2$ and $kj + k$. The paths between $(ki + \frac{k}{2} + 1, 1)$ and $(ki + \frac{k}{2} + 1, 2k^2 + 1)$ and between $(1, kj + \frac{k}{2} + 1)$ and $(2k^2 + 1, kj + \frac{k}{2} + 1)$ must intersect in this subgrid. Since these paths cannot intersect at a degree 4 vertex, at least one of the paths must turn upon meeting the other, thereby incurring at least one unit of delay.

As there are $4k^2$ subgrids of this type, and these subgrids are disjoint, the $4k$ paths must incur a total of at least $4k^2$ units of delay. Thus, some particular path $P$ must incur at least $\frac{4k^2}{4k} = k$ units of delay, proving the claim.

Letting $u$ and $v$ be the endpoints of $P$, $d_G(u,v) = 2k^2$ and $d'_S(u,v) \geq k$. □

Thus, a spanner $S$ of a sufficiently large grid $G$ must have various pairs of vertices at distances $x$ in $G$ that have delay at least $\sqrt{\frac{x}{2}}$ in $S$.

**2.2. Constructions.** We present two methods to construct $(O(\sqrt{x})+x)$-spanners with maximum degree 3 for finite grids.

In [5], we constructed *highway spanners* for finite grids by using all of the edges of every $l$th row and every $k$th column (referred to as *highways*) to subdivide the grid into $k \times l$ rectangles as illustrated in Fig. 1. To complete the spanner, we added some additional edges from within each rectangle. These edges can be added so that the only vertices of degree 4 are the intersections of the highways.

These highway spanners can be modified to produce spanners of maximum degree 3. One simple modification is to replace each highway intersection by a *roundabout* as shown in Fig. 2. In this scheme, each highway becomes a union of segments and is now contained in two horizontal or two vertical lines. Let $u$ and $v$ be two vertices at distance $x$ in $G$ on the same horizontal (or vertical) highway in such a spanner $S$. The shortest path between $u$ and $v$ in $S$ follows that highway and passes through

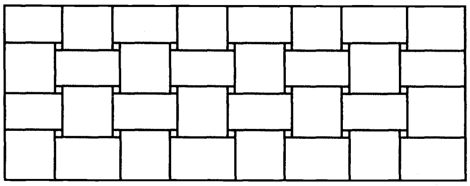


FIG. 2. *A modified highway spanner.*

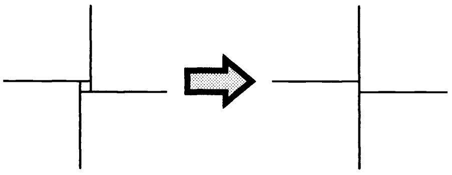


FIG. 3. *Collapsing a roundabout.*

approximately $\frac{x}{k}$ roundabouts, incurring a delay of at least 1 unit in each roundabout. Thus, the delay of such a spanner is at least $\frac{x}{k}$ or $O(x)$. We note that this delay can be eliminated in one of the dimensions by "collapsing" the roundabout into one straight path and one path with a one unit jog as shown in Fig. 3. The resulting spanner, illustrated in Fig. 4, has smaller average degree and a constant delay for pairs of vertices that are separated mainly in the $x_2$-direction. However, this type of spanner still has $O(x)$ delay for pairs of vertices that are separated mainly in the $x_1$-direction.

**2.2.1. Roundabout spanners.** A smaller maximum delay can be obtained by collapsing some of the roundabouts horizontally and some vertically. One possible method of doing this yields the *roundabout spanner*, which has delay within a constant factor of optimal.

To construct a roundabout spanner $R_{2^k,2^k}$ for the square grid $G_{2^k,2^k}$, first include the edges along the border of this grid, forming two horizontal and two vertical highways. These highways are said to have *index k*. Next, construct a roundabout on the four center vertices of the grid and extend highways in all four directions to the borders as shown in Fig. 5(a). The resulting structure consists of a roundabout with four *highway sections* radiating from it. These sections form one horizontal and one vertical highway, both of which have index $k - 1$. In order to include the remaining vertices on these lines in the spanner, we add some additional edges as shown in Fig. 5(b). When we include these edges in our discussion, we refer to highways as *augmented highways* and highway sections as *augmented highway sections*. We recursively apply this construction to each of the four quadrants of the grid, placing in each a roundabout and its four radiating augmented highway sections (see Fig. 5(c)). Note that although we have just added 16 augmented highway sections, we have only added 4 augmented highways, each with index $k-2$. A further level of subdivision is shown in Fig. 5(d). This recursion continues with index decreasing by one at each further level of subdivision until the remaining quadrants are $2 \times 2$ grids. The last augmented highways added have index 1. At this point, the structure forms a connected spanning subgraph of $G_{2^k,2^k}$. The roundabout spanner $R_{32,32}$ is shown in Fig. 6, with the indices of vertical highways indicated.

Consider a vertex $u$ of $G_{2^k,2^k}$. If $u$ is on the border of $G_{2^k,2^k}$, then it has degree

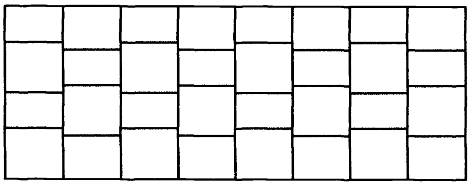


FIG. 4. *A further modified highway spanner.*

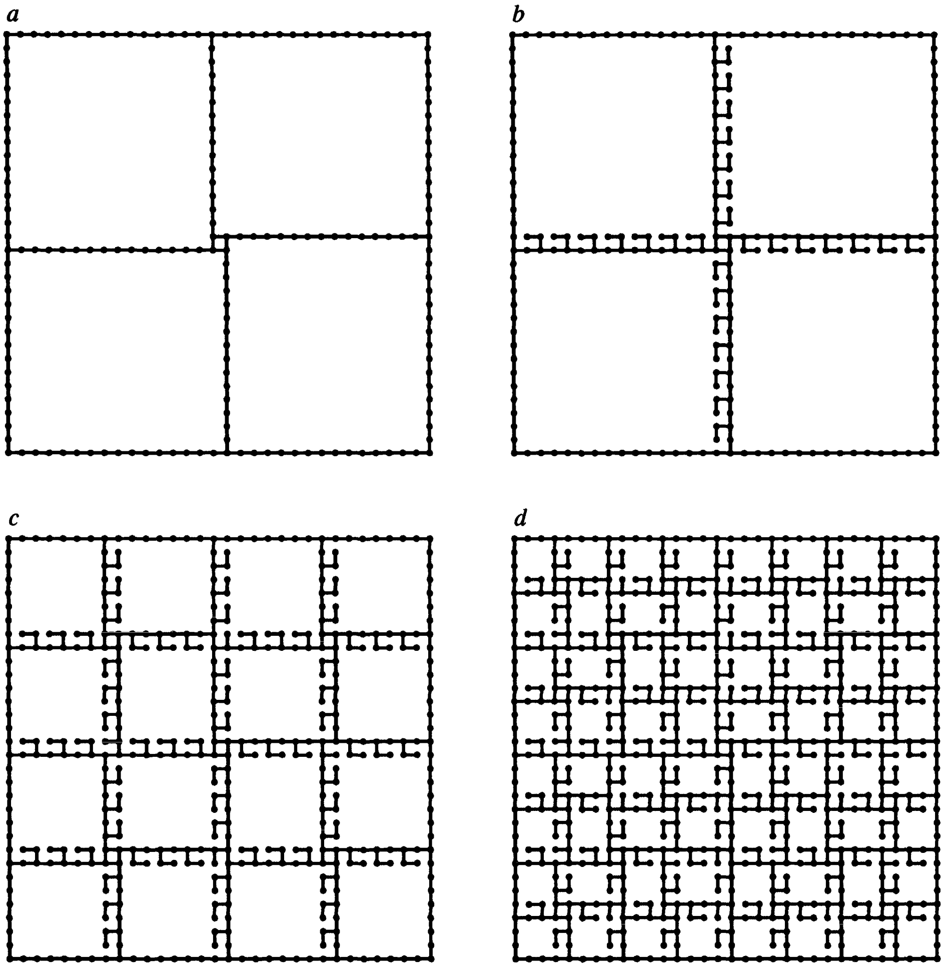


FIG. 5. *Building a roundabout spanner.*

at most 3 in the grid and, therefore, degree at most 3 in the spanner. Otherwise, the vertex $u$ is first encountered during the construction of $R_{2^k,2^k}$ when a roundabout or augmented highway section is added. Note that $u$ is included in and is a border vertex of one quadrant in which the recursive construction is applied. Being a border vertex, it could receive at most one additional incident edge from this recursive construction. Thus, if $u$ has degree at most 2 at this point of the construction, it will have degree at most 3 when the construction is completed. If $u$ is a roundabout vertex, then

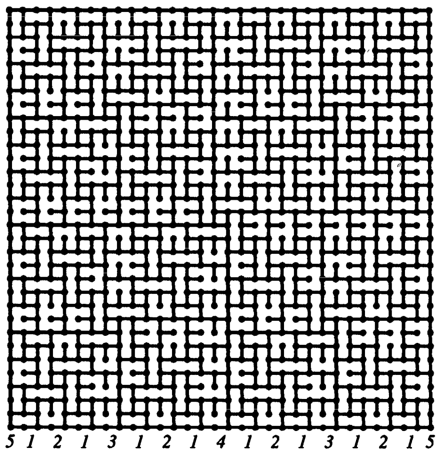


FIG. 6. *A* 32 × 32 *roundabout spanner.*

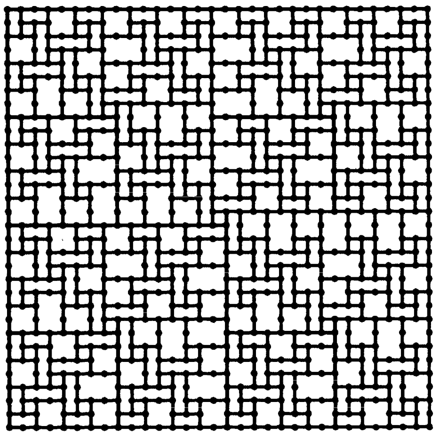


FIG. 7. *The frame of* $R_{32,32}$.

it has degree 3 at this level of the construction and is not connected to during any subsequent level. If $u$ is a degree 3 vertex which is not on the roundabout, then it is on the "clockwise" border of its quadrant when viewed from the roundabout and is at odd distance from the roundabout. However, the recursive construction ensures that only the vertices at even distance on this border are given additional incident edges. Thus, in all cases, the degree of $u$ is at most 3.

We use the term *frame* to refer to the maximum 2-connected subgraph of this (or any) spanner. The frame of $R_{32,32}$ is shown in Fig. 7. The natural embedding of the frame divides the plane into areas that we referred to as *tiles* in [3]. We use the term *rectangular frame* to refer to any frame whose tiles are rectangles. Note that, aside from the corners of such frames, all degree 2 vertices are incident on either two horizontal or two vertical edges in the frame.

Consider any two vertices $u$ and $v$ of a grid $G$. All shortest paths between $u$ and $v$ are contained in the rectangle defined by $u$ and $v$ and either increase or decrease monotonically in both the $x_1$ and $x_2$ directions. Due to the shape of such paths, we

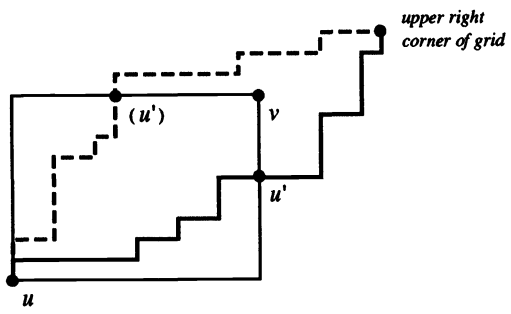


FIG. 8. *Finding the intersection point.*

call them *staircase paths.*

LEMMA 2.2. *From any vertex $u$ on a rectangular frame $F$, there is a staircase path in $F$ from $u$ to each of the four corners of $F$.*

*Proof.* Let $v$ be the upper right corner of $F$. By symmetry, we need only establish that there is a staircase path from $u$ to $v$. We observe that, for any vertex $w$ of $F$, there must be a vertex $w'$ adjacent to $w$ that is on $F$ and lies either above or to the right of $w$, unless $w = v$. Thus, we can construct a staircase path from $u$ to $v$ by including an edge to a neighbor above or to the right at each step. □

LEMMA 2.3. *Let $u$ and $v$ be two vertices in a rectangular frame $F$. There exists a vertex $u'$ in $F$ such that:*

1. *$u'$ and $v$ have either the same $x_1$-coordinate or the same $x_2$-coordinate,*
2. *$d_G(u', v) \leq d_G(u, v)$, and*
3. *$d'_F(u', v) \geq d'_F(u, v)$.*

*Proof.* If $u$ and $v$ already share an $x_1$- or $x_2$-coordinate, let $u' = u$. Otherwise, by symmetry, assume that $v$ is above and to the right of $u$. Find a staircase path from $u$ to the upper right corner of $F$. This path must intersect either the upper or right side of the rectangle defined by $u$ and $v$ in the grid $G$. Let the first such intersection point be $u'$, as shown in Fig. 8. This vertex satisfies properties 1 and 2. Noting that $d_G(u, u') = d_F(u, u')$ as there is a staircase path from $u$ to $u'$, we get $d'_F(u, v) = d_F(u, v) - d_G(u, v)$ $= d_F(u, v) - (d_G(u, u') + d_G(u', v)) \leq (d_F(u, u') + d_F(u', v)) - d_G(u, u') - d_G(u', v)$ $= d_F(u', v) - d_G(u', v) = d'_F(u', v)$, satisfying property 3. □

This lemma implies that we need only consider co-horizontal and co-vertical pairs of vertices to determine the delay function $f'_F(x)$ for a rectangular frame $F$:

COROLLARY 2.4. *Let $F$ be a rectangular frame with nondecreasing delay function $f'_F(x)$. For any positive integer $x$, if $f'_F(x) > f'_F(x - 1)$, then there exists a pair of co-horizontal or co-vertical vertices $u$ and $v$ with $d_G(u, v) = x$ and $d'_F(u, v) = f'_F(x)$.*

*Proof.* Assume, by way of contradiction, that $f'_F(x) > f'_F(x-1)$ but that no such $u$ and $v$ exist. There must be a pair of vertices $a$ and $b$ that are not colinear but with $d_G(a, b) = x$ and $d'_F(a, b) = f'_F(x)$. By Lemma 2.3, we can find an $a'$ such that $a'$ and $b$ are colinear, $d_G(a', b) \leq d_G(a, b) = x$, and $d'_F(a', b) \geq d'_F(a, b) = f'_F(x)$. Let $u = a'$ and $v = b$. If $d_G(u, v) = x$, then $d'_F(u, v) = f'_F(x)$, contradicting our assumption. Otherwise, $d_G(u, v) < x$. In this case, $d'_F(u, v) \geq f'_F(x)$ contradicts either the property that $f'_F(x)$ is nondecreasing or the hypothesis that $f'_F(x) > f'_F(x - 1)$. □

LEMMA 2.5. *If $u$ and $v$ are two vertices of $S = R_{2^k,2^k}$ that are on a vertical (or horizontal) highway with index $i$, then $d'_S(u, v) \leq \lfloor \frac{d_G(u,v)}{2^i} \rfloor + 1$.*

*Proof.* Without loss of generality, we consider a pair $u, v$ on a vertical highway. The proof for $u, v$ on a horizontal highway is similar.

A vertical highway with index $i$ is composed of sections consisting of a roundabout and its associated vertical highway sections. These sections are separated by horizontal highways of index $> i$. The intersections with these horizontal highways consist of one horizontal and one vertical edge. The order of these edges on the vertical highway is unimportant but depends on the index of the horizontal highway and the position of the vertical highway. The net effect of the roundabouts and intersections is that a vertical highway of index $i$ consists of segments of $2^i - 1$ vertical edges alternating with "turn" sections composed of one horizontal and one vertical edge in some order.

Consider the path from $u$ to $v$ along this highway. The vertical distance from $u$ to $v$ is at most $x$. We are interested in the number of turn sections encountered in the path from $u$ to $v$. Since both $u$ and $v$ may be in turn sections and since one turn section is included in every vertical interval of length $2^i$, the total number of turn sections encountered is $\leq \lceil \frac{x+2}{2^i} \rceil = \lfloor \frac{x+1}{2^i} \rfloor + 1$. Since no vertical backtracking occurs, the delay on this path is no more than the number of horizontal edges, which is no more than the number of turn sections encountered. Thus, $d'_S(u,v) \leq \lfloor \frac{x+1}{2^i} \rfloor + 1$. $\square$

Although this lemma gives us a bound on the delay between the two vertices, in general we may be able to achieve a smaller delay by following paths through highways of larger index.

THEOREM 2.6. *$R_{2^k,2^k}$ is a $(2.5\sqrt{(3x+6)/4} + 6 + x)$-spanner of $G_{2^k,2^k}$ with maximum degree* 3 *and average degree* $\frac{8}{3}(1 - \frac{1}{4^k})$.

*Proof.* By Corollary 2.4, we need only consider co-vertical (or co-horizontal) pairs of vertices. We first consider the case where $u$ and $v$ are co-vertical vertices on the frame but not on the border of the grid. Note that for any $i$, $1 \leq i < k$, the number of columns between augmented highways of index $\geq i$ is $2^i - 2$. Thus, $u$ and $v$ are either on an augmented highway of index $\geq i$ or lie in a column which is distance $\alpha$ from an augmented highway of index $i$ and distance $2^i - 1 - \alpha$ from an augmented highway of index $> i$.

Let $u'$ and $v'$ be vertices at the same horizontal positions as $u$ and $v$, respectively, on the vertical highway of index $i$. Then $\alpha \leq d_G(u,u') \leq \alpha + 1$ and $\alpha \leq d_G(v,v') \leq \alpha + 1$. Let $P_{u',v'}$ be the path from $u'$ to $v'$ along that vertical highway of index $i$.

Without loss of generality, we assume that $u$ is below $v$ and that $u'$ (and $v'$) are to the left of $u$ and $v$. Consider the staircase path $P_u$ from $u$ to the upper left corner and the staircase path $P_v$ from $v$ to the lower left corner as guaranteed by Lemma 2.2. If $P_u$ and $P_v$ intersect at a vertex $w$ to the right of $P_{u',v'}$, then the path $P$ constructed by following $P_u$ from $u$ to $w$ and then $P_v$ from $w$ to $v$ gives a path from $u$ to $v$ with delay $\leq 2\alpha$ (see Fig. 9(a)). Otherwise, $P_u$ and $P_v$ must intersect $P_{u',v'}$ at vertices $u''$ and $v''$, respectively (see Fig. 9(b)). In this case, the path $P$ from $u$ to $v$ following $P_u$ from $u$ to $u''$, $P_{u',v'}$ from $u''$ to $v''$, and $P_v$ from $v''$ to $v$ gives a path from $u$ to $v$ with delay $\leq 2(\alpha + 1) +$ the delay from $u''$ to $v''$ along $P_{u',v'}$. This latter quantity is at most the delay from $u'$ to $v'$ along $P_{u',v'}$, which is at most $\lfloor \frac{d_G(u',v')}{2^i} \rfloor + 1$ by Lemma 2.5. Thus, the delay along $P$ is at most $2(\alpha + 1) + \lfloor \frac{d_G(u,v)}{2^i} \rfloor + 1$.

We now consider the path $Q$ along the augmented highway of index $> i$ at a distance $2^i - 1 - \alpha$ from $u$ and $v$. A similar argument establishes that the delay on such a path is at most $2(2^i - \alpha) + \lfloor \frac{d_G(u,v)}{2^{i+1}} \rfloor + 1$ . (The denominator $2^{i+1}$ is due to the fact that this highway has index $\geq i + 1$.)

Thus, the delay between $u$ and $v$ is at most the minimum of the delay along $P$ and the delay along $Q$. To find an upper bound on this delay, we consider $\alpha$ to be a

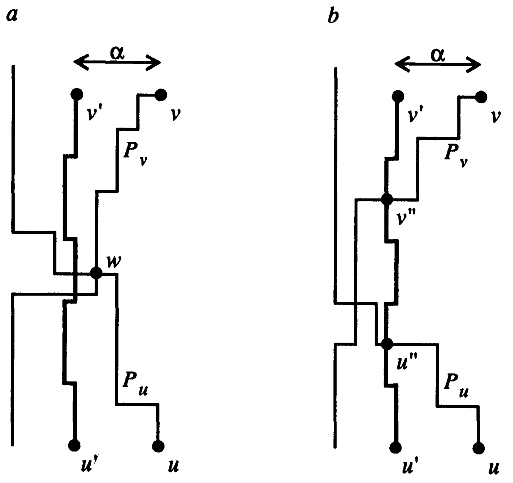


FIG. 9. *Paths between u and v.*

continuous variable and maximize

$$\min(2(\alpha+1)+\lfloor\tfrac{d_G(u,v)}{2^i}\rfloor+1, 2(2^i-\alpha)+\lfloor\tfrac{d_G(u,v)}{2^{i+1}}\rfloor+1).$$

This minimum is maximized when both delays are the same or when $\alpha = 2^{i-1}-\frac{1}{2}-\frac{x}{2^{i+3}}$ and the delay is $2^i+\frac{3x}{2^{i+2}}+2$.

The above bound on delay holds for any $i$ between 1 and $k-1$ inclusive. Considered together, these $i$ imply an upper bound of

$$\min_{1\le i\le k-1} 2^i+\frac{3x}{2^{i+2}}+2$$

on the delay between $u$ and $v$. This is, in turn, bounded above by $2^i+\frac{3x}{2^{i+2}}+2$ for any particular choice of $i$. Consider $i=\lfloor\log_2(\sqrt{3x/4})\rfloor$. (Note that $i=\log_2(\sqrt{3x/4})$ minimizes the continuous function $2^i+\frac{3x}{2^{i+2}}+2$.) This gives an upper bound on the delay of

$$2^{\lfloor\log_2(\sqrt{3x/4})\rfloor}+\frac{3x}{4\cdot 2^{\lfloor\log_2(\sqrt{\frac{3x}{4}})\rfloor}}+2,$$

which we claim is at most $2.5\sqrt{3x/4}+2$.

Let $y=\sqrt{3x/4}$. Substituting $y$ and subtracting 2 from both sides, the claim becomes

$$2^{\lfloor\log y\rfloor}+y^2\cdot\frac{1}{2^{\lfloor\log y\rfloor}}\le 2.5y.$$

Let $\phi=\log y-\lfloor\log y\rfloor$. Thus, $0\le\phi<1$. Then $2^{\lfloor\log y\rfloor}=2^{\log y-\phi}=\frac{2^{\log y}}{2^\phi}=\frac{y}{2^\phi}$. Let $\varepsilon=2^\phi$. As $0\le\phi<1$, we have that $1\le\varepsilon<2$ and $2^{\lfloor\log y\rfloor}=\frac{y}{\varepsilon}$. The claim now becomes

$$\frac{y}{\varepsilon}+y^2\cdot\frac{1}{y/\varepsilon}\le 2.5y,$$

$$\frac{y}{\varepsilon} + \varepsilon y \leq 2.5y,$$

$$\frac{1}{\varepsilon} + \varepsilon \leq 2.5,$$

$$1 + \varepsilon^2 \leq 2.5\varepsilon,$$

$$\varepsilon^2 - 2.5\varepsilon + 1 \leq 0,$$

which is true for $\frac{1}{2} \leq \varepsilon < 2$. Recalling that $1 \leq \varepsilon < 2$, the claim is proved.

This gives an upper bound of $2.5\sqrt{3x/4}+2$ for pairs of vertices on the frame that are separated by distance $x$ in the grid.

Consider a pair of vertices $u$ and $v$ that are not necessarily on the frame. As any non-frame vertex is at distance 1 from a frame vertex in $S$, we can find $u'$ and $v'$, both on the frame, such that $d_S(u,u') \leq 1$ and $d_S(v,v') \leq 1$. This implies $d_G(u',v') \leq d_G(u,v) + 2$ and $d_S(u',v') = d_G(u',v') + 2.5\sqrt{3d_G(u',v')/4} + 2$. This means that $d_S(u',v') \leq (d_G(u,v) + 2) + 2.5\sqrt{3(d_G(u,v) + 2)/4} + 2$ or $d_S(u,v) \leq d_S(u',v') \leq (d_G(u,v) + 2 + 2.5\sqrt{3(d_G(u,v) + 2)/4} + 2) + 2$. Simplifying gives a delay of at most $2.5\sqrt{3d_G(u,v) + 6/4} + 6$ between $u$ and $v$.

To calculate the average degree, we begin by summing the number of edges in highways of each index. The borders of the grid constitute the highways of index $k$. There are $4(2^k - 1)$ edges in these highways. A single roundabout of index $i$ with its four associated augmented highway sections contributes $2^{i+3} - 8$ edges. Since there are $4^{k-i-1}$ roundabouts of index $i$, there are a total of $4^{k-i-1}(2^{i+3} - 8)$ edges on augmented highways of index $i$. Summing over all indices, this gives

$$\begin{aligned}
&4(2^k - 1) + \sum_{i=1}^{k-1} 4^{k-i-1}(2^{i+3} - 8)\\
&= 2^{k+2} - 4 + \sum_{i=1}^{k-1} 2^{2k-i+1} - 2 \cdot \sum_{i=1}^{k-1} 4^{k-i}\\
&= 2^{k+2} - 4 + 2^{k+1} \cdot \sum_{i=1}^{k-1} 2^{k-i} - 2 \cdot \sum_{i=1}^{k-1} 4^{k-i}\\
&= 2^{k+2} - 4 + 2^{k+1} \cdot \sum_{i=1}^{k-1} 2^{i} - 2 \cdot \sum_{i=1}^{k-1} 4^{i}\\
&= 2^{k+2} - 4 + 2^{k+1}(2^k - 2) - 2\left(\frac{4^k - 4}{3}\right)\\
&= 2^{k+2} - \frac{12}{3} + 2^{2k+1} - 2^{k+2} - \frac{2^{2k+1}}{3} + \frac{8}{3}\\
&= \frac{2^{2k+2}}{3} - \frac{4}{3}
\end{aligned}$$

edges in the spanner. The average degree is

$$\frac{2 \cdot (\frac{2^{2k+2}}{3} - \frac{4}{3})}{2^k \cdot 2^k} = \frac{8}{3}\left(1 - \frac{1}{4^k}\right). \qquad \square$$

To construct a $(2.5\sqrt{(3x+6)/4}+6+x)$-spanner for $G_{n_1,n_2}$ with maximum degree 3, first construct $R_{2^k,2^k}$ where $k$ is chosen to be as small as possible such that $n_1 \le 2^k$ and $n_2 \le 2^k$. Add all of the edges of column $n_1$ and all of the edges of row $n_2$. Discard the portion of the resulting structure that lies outside of $G_{n_1,n_2}$ to obtain the spanner. The delay between any pair of vertices in the resulting spanner cannot exceed the delay between the corresponding vertices of $R_{2^k,2^k}$.

Note that in the roundabout spanner we have collapsed about $\frac{2}{3}$ of the roundabouts that occurred in the simple construction. If we collapse all of the roundabouts, we obtain a different spanner with smaller average degree.

**2.2.2. Jogging spanners.** To construct the *jogging spanner* $J_{2^k,2^k}$ for the square grid $G_{2^k,2^k}$, first include the edges along the border of this grid. Add the edges of row $2^{k-1}+1$ and alternate edges of row $2^{k-1}$ as shown in Fig. 10(a). This creates three augmented horizontal and two augmented vertical highways of index $k$. Next, we construct an augmented vertical highway of index $k-1$ on columns $2^{k-1}$ and $2^{k-1}+1$. The highway consists of the column $2^{k-1}$ edges from row 1 to row $2^{k-1}+1$ where it jogs to column $2^{k-1}+1$ and continues to row $2^k$. In addition to these edges, the augmented highway includes every second edge in the unused portion of these columns as shown in Fig. 10(b). We now insert two augmented horizontal highways of index $k-1$ in the same fashion: For every adjacent pair of existing augmented horizontal highways, insert a new augmented horizontal highway on the two rows midway between the existing highways. To insert a highway, begin on the even numbered row at column 1, insert horizontal edges across to column $2^k-1$, jogging to the other row of the highway whenever encountering the solid section of a vertical highway. Include every second edge in the unused portion of these rows as before to form augmented highways. The result is shown in Fig. 10(c). Repeat this step in the vertical direction to achieve the structure shown in Fig. 10(d); this figure also indicates the indices of the augmented highways created so far. The construction continues by repeating this step, alternating between the vertical and horizontal augmented highway insertions. Each set of insertions creates augmented highways with index one less than the previous set of insertions in that direction. A full construction for a $32 \times 32$ grid is shown in Fig. 11(d).

THEOREM 2.7. *$J_{2^k,2^k}$ is a $(2.5\sqrt{(3x+6)/4}+6+x)$-spanner of $G_{2^k,2^k}$ with maximum degree 3 and average degree $\frac{5}{2}-\frac{2}{4^k}$.*

*Proof.* Note that since $J_{2^k,2^k}$ has a rectangular frame, Lemma 2.2, Lemma 2.3, and Corollary 2.4 apply. By our assignment of index numbers to highways, Lemma 2.5 also holds.

To prove the upper bound $2.5\sqrt{(3x+6)/4}+6$ on the delay of $J_{2^k,2^k}$, we proceed as in the proof of Theorem 2.6 noting that the number of columns between augmented vertical highways of index $\ge i$ is $2^i-2$ (as in the previous proof) while the number of rows between augmented horizontal highways of index $\ge i$ is $2^{i-1}-2$. Following the analysis of the proof of Theorem 2.6 gives delay between co-vertical pairs of vertices of at most $2.5\sqrt{(3x+6)/4}+6$ while the delay between co-horizontal pairs of vertices is smaller.

To calculate the average degree, we begin by summing the number of edges in augmented highways of each index. The borders of the grid contribute $4(2^k-1)$ edges. The other index $k$ augmented highway is the first horizontal highway added, which contributes $(2^k-1)+(2^{k-1}-1)$ edges. Generally, each of the $2^{k-i}$ index $i$ augmented horizontal highways contributes $(2^k-1)+2^{k-i}(2^{i-1}-1)$ edges. We construct augmented horizontal highways for each index from 2 to $k$. Similarly, each

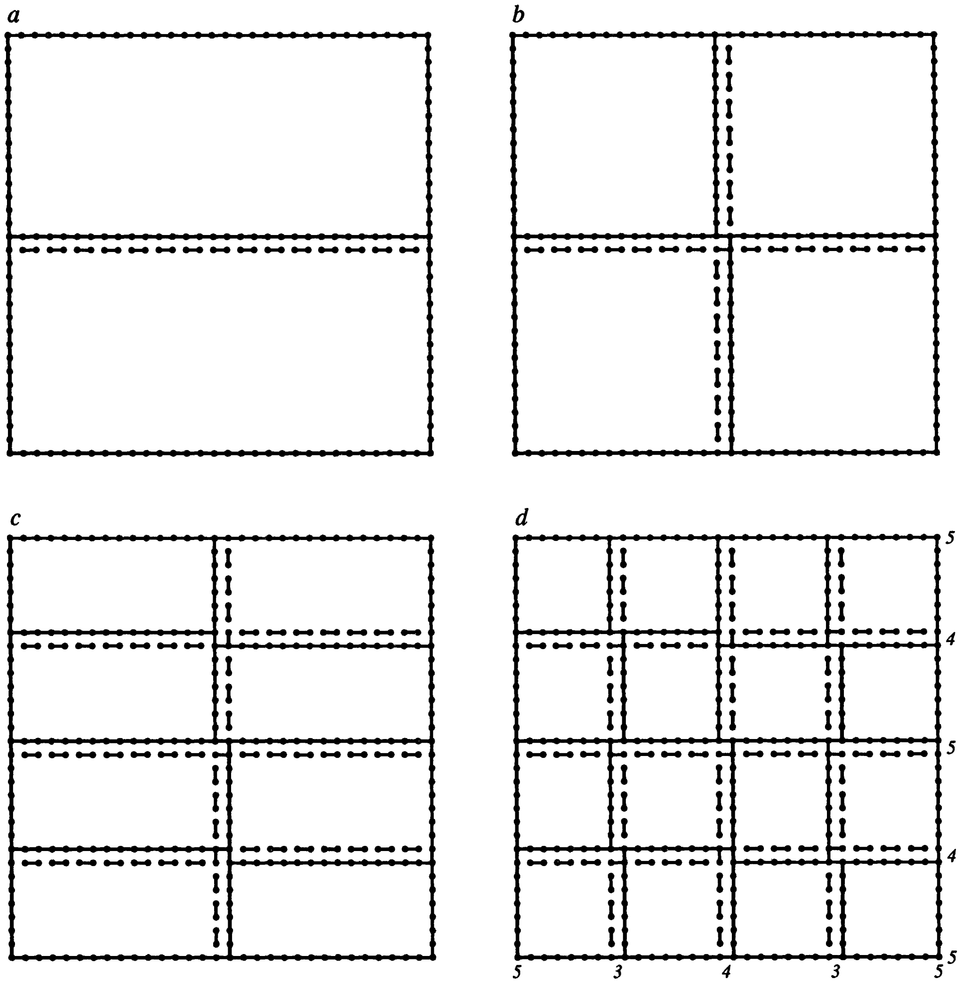


FIG. 10. *Building a jogging spanner.*

of the $2^{k-i-1}$ index $i$ augmented vertical highways contributes $(2^k-1)+2^{k-i}(2^{i-1}-1)$ edges. We construct augmented vertical highways for each index from 1 to $k-1$. Thus, the total number of edges added during the construction of $J_{2^k,2^k}$ is

$$\begin{aligned}
&4(2^k-1)+\sum_{i=2}^{k}2^{k-i}[(2^k-1)+2^{k-i}(2^{i-1}-1)]\\
&+\sum_{i=1}^{k-1}2^{k-i-1}[(2^k-1)+2^{k-i}(2^{i-1}-1)]\\
&=(2^{k+2}-4)+\sum_{i=1}^{k-1}[3\cdot 2^{2k-i-2}-2^{2k-2i-2}-2^{k-i-1}]\\
&\quad+\sum_{i=1}^{k-1}[3\cdot 2^{2k-i-2}-2^{2k-2i-1}-2^{k-i-1}]
\end{aligned}$$

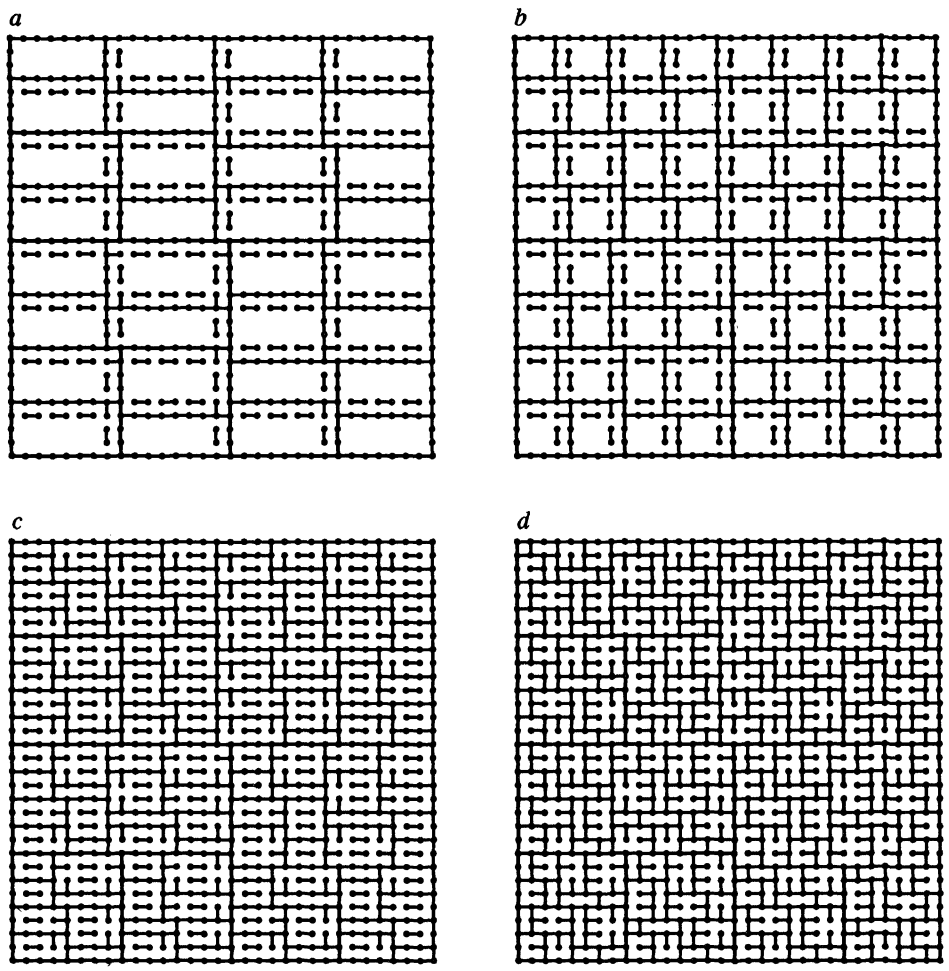


FIG. 11. *Final steps in construction of jogging spanner.*

$$
\begin{aligned}
&= (2^{k+2}-4) + 2\cdot\sum_{i=1}^{k-1} 3\cdot 2^{2k-i-2} - \sum_{i=1}^{k-1} 2^{2k-2i-2} - \sum_{i=1}^{k-1} 2^{2k-2i-1} - 2\cdot\sum_{i=1}^{k-1} 2^{k-i-1}\\
&= (2^{k+2}-4) + 3\cdot 2^{k-1}\cdot\sum_{i=1}^{k-1} 2^{k-i} - 3\cdot\sum_{i=2}^{k} 4^{k-i} - \sum_{i=1}^{k-1} 2^{k-i}\\
&= (2^{k+2}-4) + 3\cdot 2^{k-1}\cdot\sum_{i=1}^{k-1} 2^{i} - 3\cdot\sum_{i=0}^{k-2} 4^{i} - \sum_{i=1}^{k-1} 2^{i}\\
&= (2^{k+2}-4) + 3\cdot 2^{k-1}\cdot(2^k-2) - 3\cdot\left(\frac{4^{k-1}-1}{3}\right) - (2^k-2)\\
&= 5\cdot 4^{k-1} - 1.
\end{aligned}
$$

The average degree is

$$\frac{2 \cdot (5 \cdot 4^{k-1} - 1)}{2^k \cdot 2^k} = \frac{5}{2} - \frac{2}{4^k}. \qquad \square$$

As with the roundabout spanner, we can construct a $(2.5\sqrt{(3x+6)/4} + 6 + x)$-spanner of this type for $G_{n_1,n_2}$. First construct $J_{2^k,2^k}$ where $k$ is chosen to be as small as possible such that $n_1 \leq 2^k$ and $n_2 \leq 2^k$. Add all of the edges of column $n_1$ and all of the edges of row $n_2$. Discard the portion of the resulting structure that lies outside of $G_{n_1,n_2}$ to obtain the spanner. The delay between any pair of vertices in the resulting spanner cannot exceed the delay between the corresponding vertices of $J_{2^k,2^k}$. This spanner has maximum degree 3 and average degree less than that of the spanner of $G_{n_1,n_2}$ derived from the roundabout construction.

## 3. Spanners of X-trees and pyramids.

**3.1. X-tree spanners with maximum degree 3.** We begin our investigation of spanners of X-trees by reviewing the previous results in this area. Richards and Liestman [8] investigated degree-constrained $(tx)$-spanners of X-trees. In particular, they constructed a $3x$-spanner with maximum degree 3 and a $2x$-spanner with maximum degree 4. Liestman and Shermer [5] constructed a $(1+x)$-spanner of the X-tree with maximum degree 4. In this section, we are interested in constructing spanners with maximum degree 3 and improved $f(x)$.

An *X-tree* consists of a full balanced binary tree with all the leaves on the same level plus some additional edges. Edges are added to the tree such that the vertices on each level are connected, from left to right, in a path. We will use the term *horizontal edge* to denote an edge of such a path. Horizontal edges are of two types. The term *sibling edge* denotes a horizontal edge that connects two vertices with the same parent and the term *cousin edge* denotes any of the remaining horizontal edges. The term *vertical edge* denotes a tree edge. Vertical edges from a parent to a left or right child will be referred to as *left* or *right* vertical edges, respectively. We will imagine that the tree is positioned with the root on level 0 at the top and will refer to movement up and down the tree and use the terms *above* and *below* with that orientation in mind. Note that an X-tree has maximum degree 5, a non-boundary vertex has degree 5, and the average degree of a vertex in the X-tree is approximately 4. Let height$(G)$ denote the number of levels in the X-tree $G$ and let level$_G(v)$ denote the level associated with vertex $v$ the X-tree $G$. We will use $T_u$ to denote the subtree of the X-tree rooted at vertex $u$.

The following result (from [5]) will be useful in our proofs.

LEMMA 3.1. *Given any two vertices $a$ and $b$ in an X-tree, there is some shortest path between $a$ and $b$ that contains at most one sibling edge and at most two cousin edges with these horizontal edges at the top of the path.*

We note that Richards and Liestman [8] produced results under the original definition of spanner, which is concerned only with the delay between vertices adjacent in the original graph $G$. Among these results, they showed that (in their terminology) "2-spanners" of sufficiently large X-trees cannot be constructed with maximum degree 3. Although "2-spanner" corresponds to "$2x$-spanner" in our terminology, their lower bound proof only shows that $f(x)$-spanners of sufficiently large X-trees cannot be constructed with maximum degree 3 and $f(1) = 2$. (The upper bounds translate readily between the terminologies: their 3-spanner is a $3x$-spanner.) Consequently, we are still left with the problem of constructing good spanners of the X-tree with maximum degree 3.

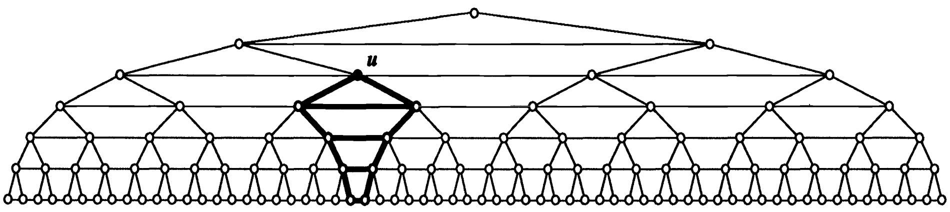


FIG. 12. *The funnel of u.*

**3.2. Lower bound for X-tree spanners with maximum degree 3.** In this section, we show that any X-tree spanner with maximum degree 3 must have delay that is linear in $x$.

THEOREM 3.2. *There is no $f(x)$ such that $f'(x) \in o(x)$, and there exist $f(x)$-spanners $S_h$ with maximum degree 3 for X-trees of every height $h+1$.*

*Proof.* If we can show that such a spanner $S$ must contain some horizontal edges, we observe that as we are limited to maximum degree 3, any endpoint of an included horizontal edge must also be the endpoint of a vertical edge that is not included in $S$. We now show that some horizontal edges must be included in $S$.

The *funnel* of a non-leaf vertex $u$, denoted $F(u)$, is the subgraph induced by $u$ and all of the endpoints of the horizontal edges directly below $u$ in the usual embedding of the X-tree as shown for the example in Fig. 12.

We claim that given a funnel $F(u)$ of such a spanner $S$, there must be a horizontal edge of $S$ in that funnel within $f(1)+1$ levels of $u$. Consider the horizontal edge $(a,b)$ of $F(u)$ that is $\lceil \frac{f(1)+1}{2} \rceil$ levels below $u$. If this edge or any horizontal edge above it in $F(u)$ is contained in $S$, then the claim is satisfied. Otherwise, any path from $a$ to $b$ in $S$ either ascends at least as far as $u$ or crosses the funnel below $(a,b)$. In the former case, the path must go up at least $\lceil \frac{f(1)+1}{2} \rceil$ levels and then down at least $\lceil \frac{f(1)+1}{2} \rceil$ levels and thus be at least $f(1)+1$ edges long. Since $S$ is assumed to be an $f(x)$-spanner, a shortest path from $a$ to $b$ must cross the funnel on a horizontal edge below $(a,b)$. If this horizontal edge is more than $\frac{f(1)-1}{2}$ levels below $(a,b)$, then this path is of length greater than $2 \cdot (\frac{f(1)-1}{2}) + 1 = f(1)$, a contradiction. Thus, in this case there must be a horizontal edge within $\frac{f(1)-1}{2}$ levels below $(a,b)$. This implies that there is a horizontal edge within $\lceil \frac{f(1)+1}{2} \rceil + \frac{f(1)-1}{2} \le f(1)+1$ levels of $u$.

Since these horizontal edges must be included, we know that some corresponding vertical edges are not included in $S$.

Let $u$ and $v$ be two vertices of an X-tree. We call $v$ an *interior descendant* of $u$ if $v$ is a descendant of $u$ and the vertical path from $u$ to $v$ includes both left and right vertical edges. If $v$ is an interior descendant of $u$, then $T_v$ is said to be *properly nested* in $T_u$, that is, no node of $T_v$ is adjacent to a node that is not in $T_u$.

Consider a shortest path in the X-tree between the root $r$ and some node $w$ at level $h$. In the X-tree, this path consists of $h$ vertical edges. In a spanner of the X-tree, such a path will include at least $h$ vertical edges and may also include some horizontal edges. Thus, the delay in the spanner is at least as large as the number of horizontal edges. Now suppose that $u$ is an ancestor of $w$ such that the edge from $u$ to its parent in the X-tree is not included in the spanner. As this is the only vertical edge out of the subtree $T_u$ rooted at $u$, then any path from $w$ to $r$ in the spanner must include a horizontal edge out of $T_u$.

We can extend this argument. Suppose that we have a sequence of vertices $r =$

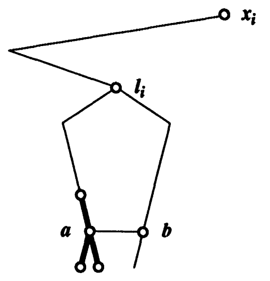


FIG. 13. *Candidates for* $x_{i+1}$.

$x_0, x_1, x_2, \ldots, x_l, w$ such that $w$ is in $T_{x_l}$, each $x_i$ is an interior descendant of $x_{i-1}$, and the edge from each $x_i$ to its parent is not included in the spanner. In this situation, any path from $w$ to $r$ in the spanner must contain at least $l$ horizontal edges and, hence, have delay at least $l$. (Due to the definition of interior descendant, any horizontal edge with an endpoint in $T_{x_i}$ has its other endpoint in $T_{x_{i-1}}$.)

We now show how to find a sequence of vertices $r = x_0, x_1, x_2, \ldots, x_l, w$ as described above with $l = \lfloor \frac{h}{f(1)+4} \rfloor$ in any $f(x)$-spanner with maximum degree 3 of an X-tree of height $h+1$. Start with $x_0 = r$. We obtain $x_{i+1}$ from $x_i$ as follows. Let $l_i$ be the "left interior grandchild of $x_i$", i.e., the vertex that is the right child of the left child of $x_i$. Note that $l_i$ and all of its descendants are interior descendants of $x_i$. Consider the funnel $F(l_i)$. There is a horizontal edge $(a, b)$ in this funnel that is included in the spanner and is at most $f(1)+1$ levels below $l_i$. The endpoint $a$ of this edge must have degree no more than 3. Thus, either the edge from $a$ to its parent is not in the spanner or the edge from $a$ to one of its children $a'$ is not in the spanner as indicated in Fig. 13. In the former case, let $x_{i+1} = a$ and in the latter case let $x_{i+1} = a'$. Thus, we can find $x_{i+1}$, which is at most $f(1)+4$ levels below $x_i$. (The vertex $l_i$ is two levels below $x_i$, $a$ is at most $f(1)+1$ levels below $l_i$, and $a'$ is one level below $a$.)

If the X-tree has $h+1$ levels, we may repeat this process $\lfloor \frac{h}{f(1)+4} \rfloor$ times. We, therefore, let $l = \lfloor \frac{h}{f(1)+4} \rfloor$ and have obtained $x_0, x_1, x_2, \ldots, x_l$. Choosing $w = x_l$, we have found the desired sequence in the spanner.

By the preceding argument, this implies that the path from $w$ to $r$ in the spanner has delay at least $l = \lfloor \frac{h}{f(1)+4} \rfloor$. Furthermore, the path from $r$ to any descendant of $w$ also has delay at least $l$.

Assume by way of contradiction that we can construct an $f(x)$-spanner $S_h$ of any X-tree of height $h+1$ such that $f'(x) \in o(x)$. This implies that for any constant $c$ there exists $x_0$ such that $f'(x) < cx$ for all $x \geq x_0$. We choose $c = \frac{1}{f(1)+4}$. By our assumption, some $x_0$ exists satisfying $f'(x) < \frac{x}{f(1)+4}$ for all $x \geq x_0$. Consider the spanner $S_h$ of an X-tree of height $h+1$ where $h$ is the smallest multiple of $f(1)+4$ that is greater than $x_0$. By the above argument, we can find a vertex $w$ at the bottom of the X-tree (i.e., at a distance greater than $x_0$ from $r$) such that the path from the root $r$ to $w$ has delay at least $l = \frac{h}{f(1)+4} = \frac{x}{f(1)+4}$. This contradicts $f'(x) < \frac{x}{f(1)+4}$ and, therefore, our assumption that $f'(x) \in o(x)$. □

**3.3. Construction of X-tree spanners with maximum degree 3.** Before describing some new spanners of this type, we note that the $3x$-spanner of Richards and Liestman [8], which has maximum degree 3, can be shown to be a $(2x+1)$-spanner

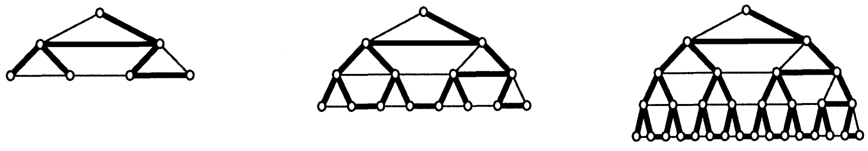


FIG. 14. 2-, 3-, *and* 4-*level shrubs.*

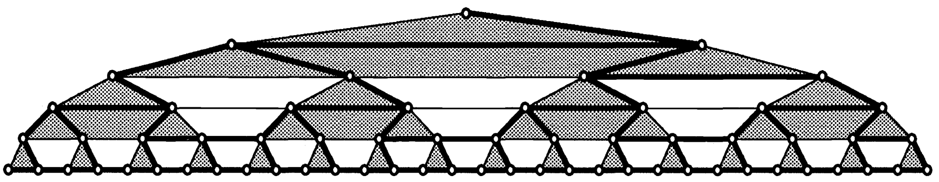


FIG. 15. 2-*level shrub spanner.*

of the X-tree. The proof is simple and is omitted here.

A *$k$-level shrub* is a spanner of an X-tree of height $k$, for $k \geq 2$. To construct a $k$-level shrub, connect the root $r$ to its right child $r_1$ and continue adding right vertical edges to form a path of $k-1$ right vertical edges from the root through vertices $r_1, r_2, \ldots, r_{k-1}$. At each vertex $r_i$ along this path, add the sibling edge to its left sibling $l_i$. Include the edges of a complete binary tree of height $k-i$ rooted at $l_i$, $1 \leq i \leq k-1$. Add all of the cousin edges at the bottom of each of these binary trees (that is, $k$ levels below $r$) and those cousin edges on the bottom level that connect these binary trees. Add the left vertical edge from $r_{k-1}$ to $l_k$ and the sibling edge from $l_k$ to $r_k$. The resulting structure is a $k$-level shrub. Figure 14 shows $k$-level shrubs for $k = 2, 3, 4$. Note that the root has degree 1 and the vertices on level $k$ below the root have degree at most 2.

To construct a $k$-level shrub spanner of an X-tree of height $h+1$, begin by constructing a $k$-level shrub rooted at the root of the X-tree. As long as $k$ levels remain below the level of the lowest vertices included in the structure, add a $k$-level shrub rooted at each vertex on the lowest level of the existing structure. Connect these newly added $k$-level shrubs by adding cousin edges between adjacent vertices from different shrubs on the bottom level of these shrubs. Note that adding these edges does not increase the degree of the vertices beyond 2. When $i \leq k$ levels remain, add the top $i$ levels of a $k$-level shrub rooted at each vertex on the lowest level of the existing structure. Add all of the missing cousin edges of the bottom level of the X-tree. Figure 15 shows a 2-level shrub spanner for an X-tree of height 5.

LEMMA 3.3. *The largest face on a $k$-level shrub spanner is bounded by at most $2k+6$ edges.*

*Proof.* The largest face within a $k$-level shrub for $k \geq 3$ is the face below the sibling edge between $l_1$ and $r_1$ that is bounded by $2(k-1)$ vertical and 3 horizontal edges, in total $2k+1$ edges. The 2-level shrub has no faces. Other faces are formed by edges from two consecutive layers of $k$-level shrubs. The largest face of this type is the face lying below a sibling edge between vertices $l_{k-1}$ and $r_{k-1}$ of some $k$-level shrub. This face is bounded by $2(k+1)$ vertical and 4 horizontal edges, in total, $2k+6$ edges. Due to the inclusion of cousin edges at the bottom of each layer, no faces are bounded by edges from more than two consecutive layers. $\square$

THEOREM 3.4. *The $k$-level shrub spanner $S$ is a $(\frac{1}{k}x + k + 8 - \frac{7}{k} + x)$-spanner of* *the X-tree with maximum degree* 3 *and average degree* $< 2\frac{3}{4}$.

*Proof.* From Lemma 3.1, we know that the shortest path $P$ between any pair of vertices $a$ and $b$ in an X-tree can be divided into two vertical sections and one horizontal section. Let $v_1$ and $v_2$ denote the number of edges in the two vertical sections and $h$ denote the number of edges in the horizontal section.

Given a $k$-level shrub spanner $S$, we note that a vertical path of length $v$ can touch upon at most $\lfloor\frac{v-2}{k}\rfloor + 2 = d$ shrubs. Since each shrub can add at most one horizontal edge (and no vertical edges) to a vertical path, the maximum number of horizontal edges added to a vertical path of length $v$ is $d$. Thus, the vertical sections of the path $P$ in the X-tree correspond to path sections in the spanner containing no more than $\lfloor\frac{v_1-2}{k}\rfloor + 2 + \lfloor\frac{v_1-2}{k}\rfloor + 2 + \leq \lfloor\frac{v_1+v_2-4}{k}\rfloor + 4$ units of delay.

Let $d_h$ denote the delay in $S$ between the endpoints of the horizontal section of $P$. The total delay along the path in $S$ corresponding to $P$ is at most $\lfloor\frac{v_1+v_2-4}{k}\rfloor + 4 + d_h = \lfloor\frac{x-h-4}{k}\rfloor + 4 + d_h$.

If the horizontal section of $P$ consists of a single sibling edge that is missing from the $k$-level shrub spanner, it can be replaced by the two edges to the parent, giving $d_h = 1$. In this case, the total delay along the path in $S$ corresponding to $P$ is at most $\lfloor\frac{x-1-4}{k}\rfloor + 4 + 1 = \frac{x-5}{k} + 5 \leq \frac{x}{k} + \frac{k-1}{k}$.

If the horizontal section of $P$ consists of a single cousin edge that is missing from the $k$-level shrub spanner, its endpoints are on a face of no more than $2k + 6$ edges, so it can be replaced by a path of length at most $k + 3$ giving $d_h \leq k + 3 - 1 = k + 2$. In this case, the total delay along the path in $S$ corresponding to $P$ is at most $\lfloor\frac{x-1-4}{k}\rfloor + 4 + (k + 2) = \frac{x-5}{k} + k + 6 \leq \frac{x}{k} + \frac{k^2+6k-5}{k}$.

If the horizontal section of $P$ consists of one sibling and one cousin edge and both are missing from $S$, then the edges to the parent $(x)$ of the sibling vertices are in the spanner. This vertex and the remaining endpoint $(w)$ of the cousin edge are on the same face of $S$. Thus, there is a path between $w$ and $x$ of no more than $k + 3$ edges giving delay of $d_h \leq (k+3+1)-2 = k+2$. In this case, the total delay along the path in $S$ corresponding to $P$ is at most $\lfloor\frac{x-2-4}{k}\rfloor + 4 + (k+2) = \frac{x-6}{k} + k + 6 \leq \frac{x}{k} + \frac{k^2+6k-6}{k}$. Note that if the horizontal section of $P$ is of this form and only one of the edges is missing, then one of the previous cases applies.

If the horizontal section of $P$ consists of one sibling edge $(v, w)$ and two cousin edges, $(u, v)$ and $(w, x)$, both of which are missing from $S$, then one of the cousin edges (say $(w, x)$) must be immediately below a sibling edge. Let $y$ be the parent of $v$ and $w$, and let $z$ the parent of $x$ (and sibling to $y$). By the structure of $k$-level shrubs, we know that there is a path in $S$ from $x$ to $z$ of at most 2 edges. There is also a path in $S$ from $z$ to $y$ of at most 2 edges consisting of either the sibling edge between them or the two edges to their parent. Finally, there is a path in $S$ from $y$ to $u$ of at most $k+3$ edges since $y$ and $u$ are on the same face of $S$. Thus, there is a path in $S$ from $u$ to $x$ of length at most $k+7$ giving delay of $d_h \leq (k+7)-3 = k+4$. In this case, the total delay along the path in $S$ corresponding to $P$ is at most $\lfloor\frac{x-3-4}{k}\rfloor + 4 + (k+4) = \frac{x-7}{k} + k + 8 \leq \frac{x}{k} + \frac{k^2+8k-7}{k}$. If the horizontal section of $P$ is of this form and either of the cousins is present, then one of the previous cases applies.

Thus, the total delay along the path in $S$ corresponding to $P$ is at most $\frac{x}{k} + \frac{k^2+8k-7}{k}$, so $f(x) \leq \frac{k+1}{k}x + k + 8 - \frac{7}{k}$.

To bound the average degree, we note that most of the vertices that are not at the bottom level of the spanner are of degree 3. Consider a vertex not at the bottom level of the spanner. If this vertex is the second vertex from the right on the bottom level of its shrub, then it has degree 2; if it is the root of either the leftmost or rightmost shrub on its level, then it has degree 2; if it is the root of the X-tree, then it has degree 1. Otherwise, it has degree 3. As the vertices of degree $< 3$ are relatively sparse, we can obtain a good upper bound on the average degree by assuming that all vertices not at the bottom level of the X-tree have degree 3.

Consider now the vertices at the bottom of the X-tree. By our construction, at most one such vertex in each shrub (or partial shrub) is of degree 3. The other vertices at this level are of degree at most 2. There are at most $2^{h-1}$ shrubs (or partial shrubs) at this level. As there are $2^h$ vertices at the bottom level and $2^h - 1$ vertices not on the bottom level, the average degree is at most

$$\frac{3 \cdot 2^{h-1} + 2 \cdot 2^{h-1} + 3(2^h - 1)}{2^{h+1} - 1} = \frac{11 \cdot 2^{h-1} - 3}{2^{h+1} - 1} = \frac{\frac{11}{4} \cdot 2^{h+1} - \frac{11}{4} - \frac{1}{4}}{2^{h+1} - 1} < \frac{11}{4} = 2\frac{3}{4}. \quad \square$$

**3.4. Pyramid spanners.** We begin our investigation of spanners of pyramids by reviewing the previous results in this area. Richards and Liestman [8] investigated degree-constrained $(tx)$-spanners of pyramids. In particular, they constructed a $2x$-spanner with maximum degree 6, a $3x$-spanner with maximum degree 4, and a $7x$-spanner with maximum degree 3. Liestman and Shermer [5] constructed a $(1 + x)$-spanner of the pyramid with maximum degree 7. In this section, we are interested in constructing spanners with lower maximum degree and improved $f(x)$.

A three-dimensional pyramid consists of a series of levels with $4^i$ vertices on level $i$, starting with 1 vertex on level 0. The vertices on each level are arranged in a $2^i \times 2^i$ square grid graph or "mesh," where each vertex is connected to its four orthogonal neighbors. Further, each vertex is connected to the four corresponding vertices on the level below it. A pyramid is shown in Fig. 16. As with X-trees, we can partition the edges of the pyramid into *horizontal* and *vertical* edges, further classifying the horizontal edges as *sibling* or *cousin* edges. The horizontal edges also naturally fall into two classes of parallel edges corresponding to dimensions that we call $x_1$ and $x_2$. We will use the terms *left* and *right* to refer to the relations $<$ and $>$ in the $x_1$ dimension and similarly use the terms *front* and *back* to refer to the relations $<$ and $>$ in the $x_2$ dimension.

Let $P$ be a maximum length path in one dimension ($x_1$ or $x_2$) in the bottom level of the pyramid. The set $A_P$ consisting of nodes of $P$ and all of their ancestors induces an X-tree $X_P$ in the pyramid. If $P$ is in dimension $x_i$, we call such an X-tree an *induced X-tree in dimension* $x_i$. An induced X-tree is shown in Fig. 17.

Let height$(G)$ denote the number of levels in the pyramid $G$, and let level$_G(v)$ denote the level associated with vertex $v$ in the pyramid $G$. We will use $T_u$ to denote the subpyramid of the pyramid rooted at vertex $u$.

The following result (from [5]) will be useful in our proofs.

LEMMA 3.5. *Given any two vertices $a$ and $b$ in a pyramid, there is some shortest path between $a$ and $b$ that contains at most one sibling edge and at most three cousin edges with these horizontal edges at the top of the path.*

As discussed above, the lower bounds given by Richards and Liestman [8] do not directly apply with our definition of $f(x)$-spanner. Their bounds show that there is no $f(x)$-spanner of sufficiently large pyramids with:

1. maximum degree 5 and $f(1) = 2$;
2. maximum degree 3 and $f(1) = 3$;
3. maximum degree 2 and $f(1) = O(1)$.

Below, we give improved bounds on $f(x)$ for degree-constrained $f(x)$-spanners of pyramids.

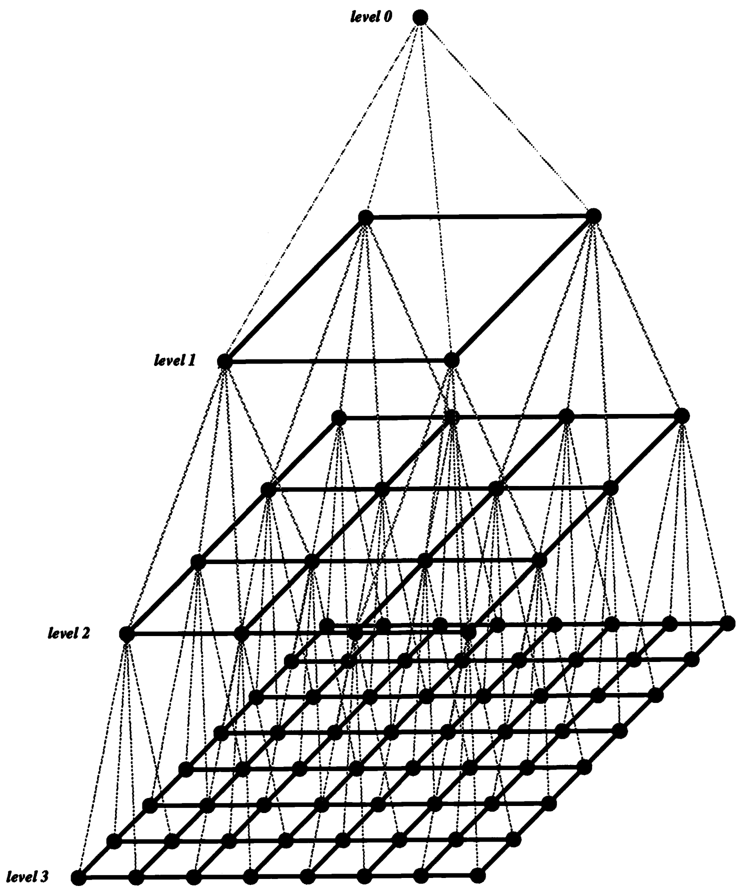


FIG. 16. *A pyramid.*

**3.5. Construction of pyramid spanners with maximum degree 6.** Although the focus of this paper is on spanners with nonconstant delay, we include the following result for the sake of completeness. In [5], we showed how to construct a $(1+x)$-spanner of the pyramid with maximum degree 7. A constant delay spanner with maximum degree 6 can also be constructed.

THEOREM 3.6. *For any pyramid $P$, there exists a $(2+x)$-spanner $S$ with maximum degree* 6 *and average degree* $< \frac{15}{4}$.

*Proof.* To construct such a spanner $S$, omit all of the sibling edges from $P$. Omit all of the cousin edges in one dimension on even levels and all of the cousin edges in the other dimension on odd levels of the pyramid. All of the cousin edges on the lowest level of the pyramid can remain. The resulting graph has maximum degree 6 and average degree less than $\frac{15}{4}$.

To prove that $S$ is a $(2+x)$-spanner is relatively straightforward and can be done as in the proofs of [5]. We omit the details here.

To calculate the average degree, note that the spanner contains $\sum_{i=0}^{h} 4^i = \frac{4^{h+1}-1}{3}$ vertices, $\sum_{i=1}^{h} 4^i$ vertical edges, and $\sum_{i=2}^{h-1}((2^{i-1}-1)2^i)+2^{h+1}(2^{h-1}-1)$ cousin edges

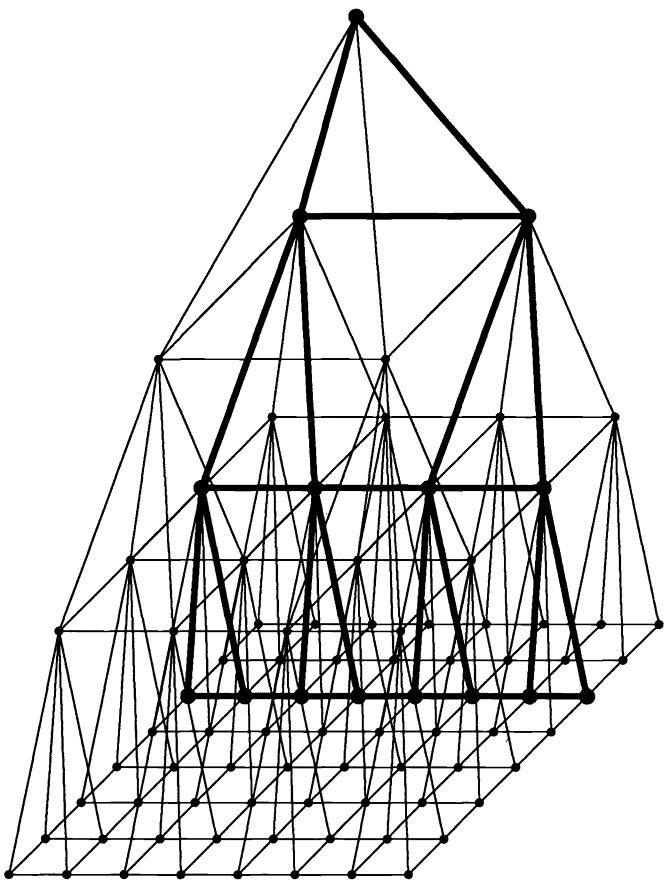


FIG. 17. *An induced X-tree.*

where $h+1$ is the number of levels in the pyramid $P$. The total number of edges in $S$ is

$$\sum_{i=1}^{h} 4^i + \sum_{i=2}^{h-1}((2^{i-1}-1)2^i) + 2^{h+1}(2^{h-1}-1)$$

$$= \sum_{i=1}^{h} 4^i + \sum_{i=2}^{h-1} 2^{2i-1} - \sum_{i=2}^{h-1} 2^i + 2^{2h} - 2^{h+1}$$

$$= \sum_{i=1}^{h} 4^i + \frac{1}{2}\sum_{i=2}^{h-1} 4^i - \sum_{i=2}^{h-1} 2^i + 2^{2h} - 2^{h+1}$$

$$= \sum_{i=1}^{h} 4^i + \frac{1}{2}\left(\sum_{i=1}^{h} 4^i - 4 - 4^h\right) - \sum_{i=1}^{h} 2^i + 2 + 2^h + 2^{2h} - 2^{h+1}$$

$$= \sum_{i=1}^{h} 4^i + \frac{1}{2}\sum_{i=1}^{h} 4^i + \frac{1}{2}4^h - 2^h - \sum_{i=1}^{h} 2^i$$

$$\begin{aligned}
&= \frac{3}{2}\left(\frac{1-4^{h+1}}{1-4}\right) - \frac{3}{2} + \frac{1}{2}4^h - 2^h - 2^{h+1} + 2 \\
&= \frac{1}{2}4^{h+1} + \frac{1}{2}4^h - \frac{3}{2}2^h \\
&= \frac{5}{2}\cdot 4^h - 3\cdot 2^{h-1}.
\end{aligned}$$

The average degree is

$$(2(\tfrac{5}{2}\cdot 4^h - 3\cdot 2^{h-1}))/(\frac{4^{h+1}-1}{3}) = \frac{15\cdot 4^h - 18\cdot 2^{h-1}}{4^{h+1}-1} < \tfrac{15}{4}. \qquad \square$$

**3.6. Lower bound for pyramid spanners with maximum degree 5.** In this section, we show that any pyramid spanner with maximum degree 5 must have delay that is linear in $x$.

THEOREM 3.7. *There is no $f(x)$ such that $f(x) \in o(x)$, and there exist $f(x)$-spanners $S_h$ with maximum degree* 5 *for pyramids of every height $h+1$.*

*Proof.* If we can show that such a spanner $S$ must contain some horizontal edges, we observe that as we are limited to maximum degree 5, any endpoint of an included horizontal edge must be incident on a vertical edge that is not included in $S$. We now show that some horizontal edges must be included in $S$.

The *funnel* of a non-leaf vertex $u$ of the pyramid, denoted $F(u)$, is the subgraph induced by $u$ and the four innermost descendants of $u$ on each level below $u$. A funnel is shown in Fig. 18.

We claim that there must be a horizontal edge of $S$, in $T_u$, within $f(1)+1$ levels of $u$. Consider a horizontal edge $(a,b)$ of $F(u)$ that is $\lceil\frac{f(1)+1}{2}\rceil$ levels below $u$. Any path of length $f(1)$ between $a$ and $b$ is contained in $T_u$. If $(a,b)$ or any horizontal edge at least as high as $(a,b)$ in $T_u$ is contained in $S$, then the claim is satisfied. Otherwise, a shortest path from $a$ to $b$ in $S$ either ascends at least as far as $u$ or includes a horizontal edge parallel to $(a,b)$ below $(a,b)$. In the former case, the path must go up at least $\lceil\frac{f(1)+1}{2}\rceil$ levels and then down at least $\lceil\frac{f(1)+1}{2}\rceil$ levels and thus be at least $f(1)+1$ edges long. This is a contradiction, so a shortest path from $a$ to $b$ must include a horizontal edge below $(a,b)$. If this horizontal edge is more than $\frac{f(1)-1}{2}$ levels below $(a,b)$, then this path is of length greater than $2\cdot(\frac{f(1)-1}{2})+1 = f(1)$, a contradiction. Thus, there must be a horizontal edge within $\lceil\frac{f(1)+1}{2}\rceil$ levels below $(a,b)$. This implies that there is a horizontal edge within $\lceil\frac{f(1)+1}{2}\rceil + \frac{f(1)-1}{2} \le f(1)+1$ levels of $u$.

Since these horizontal edges must be included, we know that some corresponding vertical edges are not included in $S$.

Given any $f(x)$-spanner with maximum degree 5 of a pyramid of height $h+1$, we may proceed, as in the proof of Theorem 3.2, to construct a sequence of vertices $r = x_0, x_1, x_2, \ldots, x_l, w$ with $l = \lfloor\frac{h}{f(1)+4}\rfloor$ such that $w$ is in $T_{x_l}$, each $x_i$ is an interior descendant of $x_{i-1}$, and the edge from each $x_i$ to its parent is not included in the spanner. (In a pyramid, vertex $v$ is considered to be an *interior descendant* of vertex $u$ if $v$ is an interior descendant of $u$ in induced X-trees in both the $x_1$ and $x_2$ dimensions.) As before, this implies that the path from $r$ to $w$ (or any descendant of $w$) in the spanner has delay at least $l = \lfloor\frac{h}{f(1)+4}\rfloor$.

Again, as in the proof of Theorem 3.2, by assuming that we can construct an $f(x)$-spanner $S_h$ of any pyramid of height $h+1$ such that $f'(x) \in o(x)$, we arrive at a contradiction. $\square$

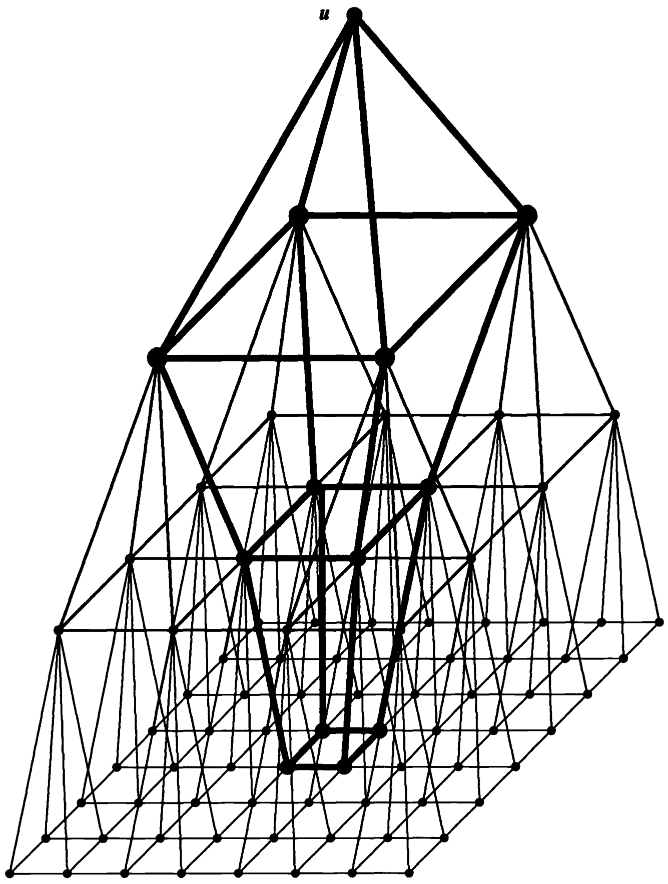


FIG. 18. *The funnel of u.*

**3.7. Construction of pyramid spanners with maximum degree 5.** A *$k$-level hedge* is a spanner of a pyramid of height $k$, for $k \geq 2$. To construct a $k$-level hedge, connect the root $r$ to its two right children and continue adding all right vertical edges to form a binary tree of height $k-1$ on the right face of the pyramid. From each vertex in this tree, add the sibling edge to its left sibling. Under each such left sibling, construct a tree by including all of the vertical edges down to the bottom level of the pyramid. Add all of the cousin edges at the bottom of each of these trees (that is, $k$ levels below $r$) and those cousin edges on the bottom level that connect these trees. Note that the binary tree constructed on the right face of the pyramid has its leaves one level above the bottom of the pyramid. From each of these leaves, $l$, add the vertical edges to connect to the two left children. From each of these children, add the sibling edge to the right sibling. From all four children of $l$, add cousin edges in the $x_2$ dimension (from front to back). The resulting structure is a $k$-level hedge. Figure 19 shows the vertical and sibling edges of a 3-level hedge where the solid areas indicate complete trees of vertical edges with no horizontal edges present except those horizontal edges explicitly included on the bottom level. Figure 20 shows the cousin

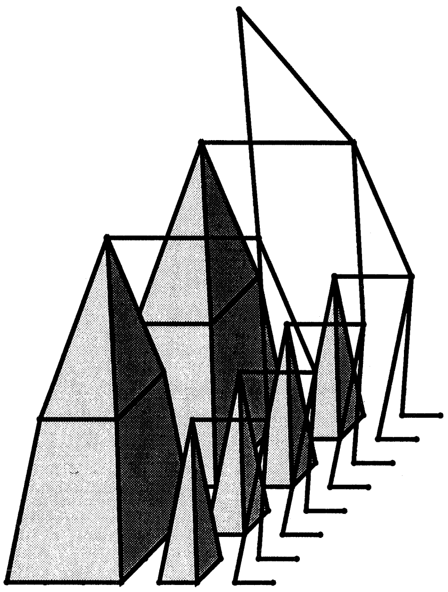


FIG. 19. *Vertical and sibling edges of a 3-level hedge.*

and sibling edges on the bottom level of a 3-level hedge. Note that the root has degree 2 and the vertices on lowest level have degree at most 3.

To construct a $k$-level hedge spanner of a pyramid of height $h+1$, begin by constructing a $k$-level hedge rooted at the root of the pyramid. As long as $k$ levels remain below the level of the lowest vertices included in the structure, add a $k$-level hedge rooted at each vertex on the lowest level of the existing structure. Connect these newly added $k$-level hedges by adding cousin edges between adjacent vertices from different hedges on the bottom level of these hedges. Note that adding these edges does not increase the degree of the vertices beyond 3. When $i \leq k$ levels remain, add the top $i$ levels of a $k$-level hedge rooted at each vertex on the lowest level of the existing structure. Add all of the missing cousin edges of the bottom level of the pyramid.

Consider a $k$-level hedge spanner $S$ of the pyramid. We note that, by our construction, if $P$ is a path in dimension $x_1$, then $A_P$ induces a $k$-level shrub spanner $S'$ of $X_P$. An induced shrub spanner is shown in Fig. 21. If $P$ is a path in dimension $x_2$, then $A_P$ induces a structure $Y_P$ in $X_P$ (and in the pyramid). $Y_P$ can be constructed as follows: The vertices of $A_P$ are on $h+1$ different levels of $X_P$ (and of the pyramid). From the top of $X_P$, group these levels into $\lceil \frac{h+1}{k} \rceil$ groups of $k$ levels each (except the bottom group that may have $< k$ levels). From each of these groups, choose at most one level. Include all vertical edges of $X_P$ except those going down from the chosen levels. Beginning at the root, include all cousin edges at each $k$th level. Finally, include all horizontal edges at the bottom level. An example of $Y_P$ in a single hedge is shown in Fig. 22. Note that the vertical edges missing from $Y_P$ are the endpoints of a path of length 2 in the spanner $S$ consisting of a left to right sibling edge and a vertical edge.

THEOREM 3.8. *The $k$-level hedge spanner $S$ is a $(\frac{1}{k}x + k + 8 - \frac{7}{k} + x)$-spanner of the pyramid with maximum degree 5 and average degree $< 3\frac{7}{8}$.*

*Proof.* Consider a $k$-level hedge spanner $S$. From Lemma 3.5, we know that the

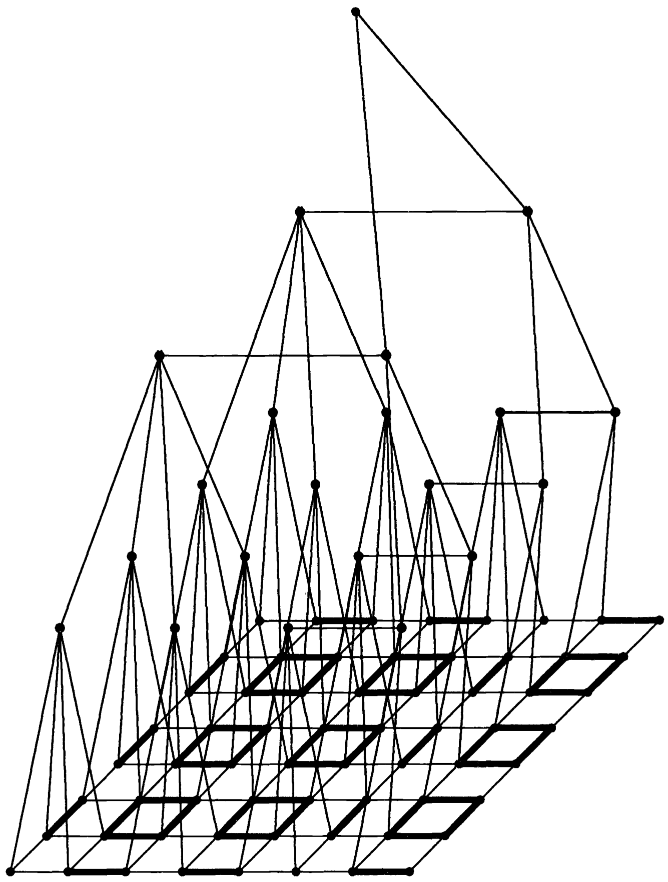


FIG. 20. *Cousin and sibling edges on the bottom level of a 3-level hedge.*

shortest path $P$ between any pair of vertices $a$ and $b$ in a pyramid can be divided into two ascending vertical sections and one horizontal section. Let $v_1$ and $v_2$ denote the number of edges in the vertical sections and $h$ denote the number of edges in the horizontal section. Note that any ascending vertical path is contained in some induced X-tree in dimension $x_1$. Since this X-tree induces a $k$-level shrub spanner, the analysis in the proof of Theorem 3.4 holds. Thus, the vertical sections of the path $P$ in the pyramid correspond to path sections in the spanner $S$ containing no more than $\lfloor \frac{v_1+v_2-4}{k} \rfloor + 4$ units of delay. Let $d_h$ denote the delay in $S$ between the endpoints of the horizontal section of $P$. The total delay along the path in $S$ corresponding to $P$ is at most $\lfloor \frac{x-h-4}{k} \rfloor + 4 + d_h$.

The horizontal section of $P$ consists of at most one sibling edge and at most three cousin edges. Using **c** to denote a cousin edge and **s** to denote a sibling edge, the horizontal section of the path must be of one of the following forms: **ccsc, csc, ccs, cc, sc, s**, or **c**. The possible layouts of these forms are shown in Fig. 23. These layouts are either entirely in the $x_1$ or $x_2$ dimension or consist of a section in the $x_1$ dimension and a section in the $x_2$ dimension. In any case, if we consider only the

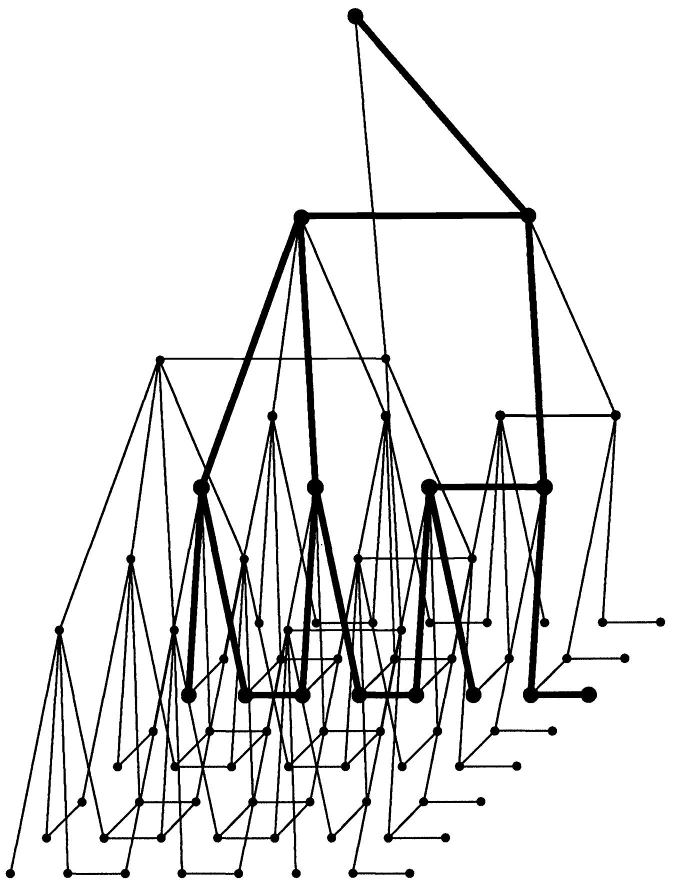


FIG. 21. *An induced shrub spanner.*

sections in one dimension, the only forms that appear are the straight layouts of **csc**, **sc**, **s**, and **c**.

Consider the $x_1$ dimension. Any horizontal path section in this dimension is contained in an induced X-tree in dimension $x_1$. From the proof of Theorem 3.4, we know that the delay between the endpoints of this section in the induced $k$-level shrub spanner is at most $1, k+2, k+2$, and $k+4$ for sections of forms **s**, **c**, **sc**, and **csc**, respectively.

Consider the $x_2$ dimension. Any horizontal path section in this dimension is contained in a structure $Y_P$ as described above.

If the horizontal path section in dimension $x_2$ is a single sibling edge, then either the two edges to the parent vertex are in $S$ or these two edges are missing. In the former case, the delay is 1 and in the latter case the delay is 3 (using the two paths of length 2 to the parent).

If the horizontal path section in dimension $x_2$ is a single cousin edge $(u, x)$, then consider $Y_P$ augmented with the paths of length 2 that substitute for the missing vertical edges. The two endpoints of the cousin edge are on a cycle in this structure

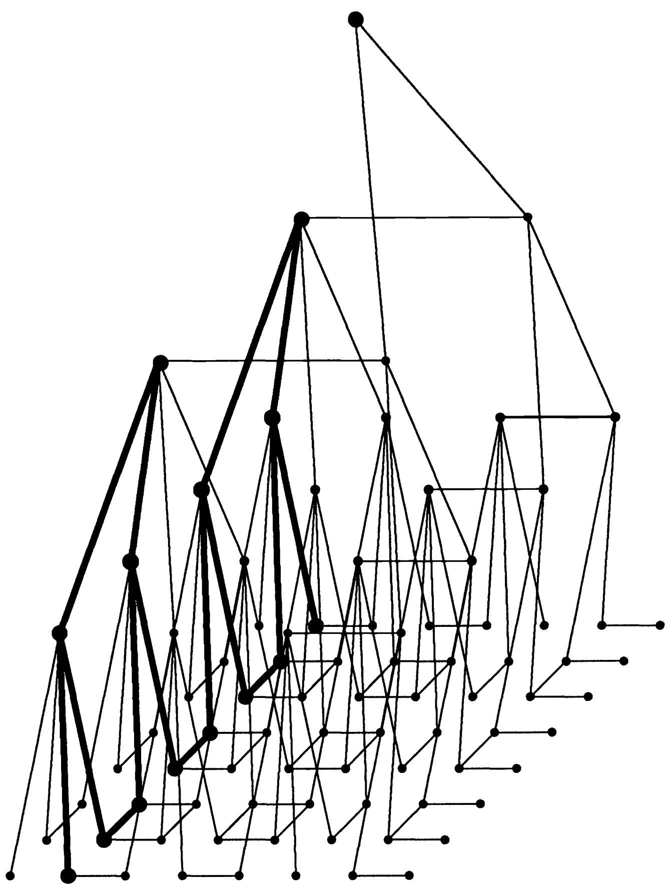


FIG. 22. *An induced $Y_P$ in a single hedge.*

(the augmented $Y_P$) of length at most $2k+4$. The cycle includes paths from $u$ and $x$ upward in the structure to either a common ancestor or up to the next level on which cousin edges are included. The cycle also includes paths from $u$ and $x$ down to the next level on which cousin edges are included. These paths are chosen so that the bottom vertices are connected by a cousin edge that is included in the cycle. This cycle is concluded by joining these upward paths at the common ancestor of $u$ and $x$ or by adding a cousin edge at the top level. The upward and downward paths consist entirely of vertical edges except for there being possibly two horizontal edges included as part of the length 2 paths replacing missing vertical edges. Thus, there is a path from $u$ to $x$ of length at most $\frac{2k+4}{2} = k+2$, so the delay is at most $k+1$.

If the horizontal section consists of one sibling edge $(u, v)$ and one cousin edge $(v, x)$, then the parent of $u$ and $v$ lies on the same cycle of length $2k+4$ as $v$ and $x$ (as described in the preceding case). There must be a path in $S$ of length at most 2 from $u$ to the parent of $u$ and $v$. Thus, there is a path of length at most $k+4$ between $u$ and $x$ and the delay is at most $k+2$.

If the horizontal section consists of one sibling edge $(v, w)$ and two cousin edges

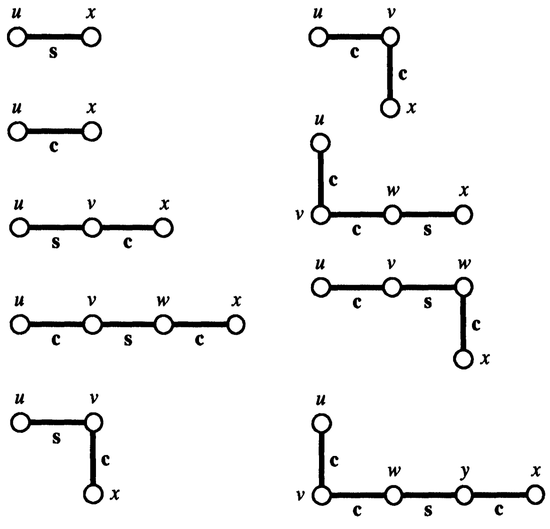


FIG. 23. *Possible layouts of the horizontal section.*

$(u, v)$ and $(w, x)$, then one of the cousin edges (say $(w, x)$) must be immediately below a sibling edge. Let $y$ be the parent of $v$ and $w$, and let $z$ be the parent of $x$ (and sibling of $y$). We now consider two cases depending on whether cousin edges are present at the level of $z$. If there are cousin edges at this level, then we obtain a path of length at most 9 by going from $u$ to its parent, across the cousin edge to $y$, from $y$ to its parent, down to $z$, and then down to $x$. By the structure of the augmented $Y_P$, each of these vertical steps can be accomplished with at most two edges. If there are not cousin edges at this level, then we obtain a path of length at most $k+7$. As in the analysis of the **sc** case, there is a path from $u$ to $y$ of length at most $k+2$. To this path we append a path from $y$ to its parent, down to $z$, and then down to $x$. In this case, however, at most 2 of these appended vertical edges are missing from $Y_P$ and thus the appended path is of length at most 5, giving a path of length at most $k+7$. Because $k \geq 2$, we have in either case that the length of the path from $v$ to $x$ in $S$ is at most $k+7$, giving a delay of at most $k+4$.

We now consider the layouts that contain edges in both dimension.

If the horizontal section consists of one sibling edge $(u, v)$ and one cousin edge $(v, x)$ where $(u, v)$ and $(v, x)$ are in different dimensions, then let $y$ be the parent of $u$ and $v$ . If $(u, v)$ is in the $x_2$ dimension and $(v, x)$ is in the $x_1$ dimension, then there is a path of length at most 2 from $u$ to $y$ and, as noted above, there must be a path from $y$ to $x$ of length at most $k+3$. Thus, the delay is at most $k+3$. If $(u, v)$ is in the $x_1$ dimension and $(v, x)$ is in the $x_2$ dimension, then either $(u, v)$ is in $S$ or $(u, y)$ is in $S$. As in the case of the straight layouts in the $x_2$ dimension, $x$, $y$, and $v$ all lie on a cycle of length at most $2k+4$ in $S$. Thus, there is a path from $u$ to $x$ of length at most $k+3$, giving a delay of at most $k+1$.

Consider the layout of **cc** where one cousin edge $(u, v)$ is in the $x_1$ dimension and the other $(v, x)$ is in the $x_2$ dimension. In the pyramid, there are ascending paths from $u$ and $x$ to either a common ancestor or up to the first level on which cousin edges are included. In the pyramid, there are also descending paths from $u$ and $x$ (such that at each level the vertices on these paths are also separated by two cousin edges) that continue down to the first level encountered on which cousin edges are included.

We follow the convention that if $u$, $v$, and $x$ are on a level with cousin edges, then the upward paths are empty and the downward paths are of length $k$. We use the term *vertical path on* $u$ to denote the union of the upward and downward paths from $u$ described above. In $S$, at most one edge of the vertical path on $u$ (or $x$) is missing. Such a missing edge corresponds to a path of length 2 in $S$, and thus the path in $S$ corresponding to the vertical path on $u$ is of length at most $k + 1$ (and similarly for $x$). The two cousin edges separating the bottom vertices of these vertical paths are present in $S$, as the only missing cousin edges on this level are directly below sibling edges. The top vertices of these paths are either the same (a common ancestor of $u$ and $x$) or they are separated by one or two cousin edges in the pyramid. The cousin edges are present unless (at most) one of them is directly below a sibling edge. This would occur if the missing cousin edge was a rightmost cousin inside a $k$-level hedge. This missing edge corresponds to a length 3 path in $S$ (two vertical edges and one sibling), and thus the top vertices of the vertical paths on $u$ and $v$ are connected by a path of length at most 4 in $S$. The vertical paths on $u$ and $x$ in $S$ together with the two cousin edges at the bottom and the connection at the top comprise a cycle of length at most $2k+8$ in $S$. Thus, there is a path from $u$ to $x$ of length at most $k+4$, giving a delay of at most $k+2$.

If the horizontal section consists of two cousin edges $(u, v)$ and $(v, w)$ followed by a sibling edge $(w, x)$, then let $y$ denote the parent of $w$ and $x$. As in the previous case, we can find a cycle in $S$ of length no more than $2k + 8$ including vertices $u$ and $w$. If no cousin edges are included on the level of $u$, $v$, $w$, and $x$, then $y$ is also on this cycle and, thus, a path from $u$ to $y$ of at most $k + 4$ edges. There is also a path in $S$ from $y$ to $x$ of at most two edges, giving a path from $u$ to $x$ of at most $k + 6$ edges and a delay from $u$ to $x$ of at most $k+3$. If, however, the level of $u$, $v$, $w$, and $x$ does include some cousin edges, then one of $(u, v)$ and $(v, w)$ is present in $S$. If $(u, v)$ is in S, then this reduces to the case of the straight layout of **sc** (from $v$ to $x$). If $(v, w)$ is in S, then let $w'$ be the vertex other than $v$ that is a cousin of both $u$ and $w$. Due to our construction, $(u, w')$ must also be in $S$, and this reduces to the case of the bent layout of **sc** (from $w'$ to $x$).

If the horizontal section of the path consists of the bent form of **csc** from $u$ to $x$, then there is also a path with the horizontal section from $u$ to $x$ of form **ccs** and the previous analysis applies.

Finally, we consider the form **ccsc**. Let $(u, v)$, $(v, w)$, and $(y, x)$ be cousin edges and $(w, y)$ be a sibling edge. We can choose one of the two single bend shortest paths from $u$ to $x$ so that $w$ and $x$ have a common grandparent $z$.

Consider the case when cousin edges are not included in the level immediately above $u$ and $v$. As before, we can find a cycle in $S$ of length no more than $2k+8$ that includes $u$ and $w$. This cycle will include $z$, and therefore there is a path of length at most $k + 4$ between $u$ and $z$. Furthermore, at most one of the vertical edges in the path from $z$ to $x$ is not included in $S$, giving a path of length at most 3 from $z$ to $x$. The resulting path from $u$ to $x$ is of length at most $k + 7$, giving a delay of at most $k+3$.

Consider the case when cousin edges are included in the level immediately above $u$ and $v$. Let $u'$ and $v'$ be the parents of $u$ and $v$, respectively. Let $w'$ be the parent of $w$ and $y$. Note that $(v', w')$ must be a cousin edge and that $(u', v')$ may be either a cousin edge or a sibling edge. Also note that if either of these edges is a cousin and is not present in $S$, then the length 3 path between its endpoints through their parents is included in $S$. Consider the case that $(u', v')$ is a cousin edge. If $(u', v')$

and $(v', w')$ are both in $S$, then there is a path from $u$ to $x$ consisting of a vertical path from $u$ to $u'$, the edges $(u', v')$ and $(v', w')$, a vertical path from $w'$ to $z$, and a (two-level) vertical path from $z$ to $x$. As each level of a vertical path may take at most two edges, the total length of this path is at most 10, giving a delay of 6. If $(u', v')$ is in $S$ and $(v', w')$ is not in $S$, then there is a path from $u$ to $x$ consisting of a vertical path from $u$ to $u'$, the edge $(u', v')$, the edge from $v'$ to its parent, the sibling edge from the parent of $v'$ to $z$, and a (two-level) vertical path from $z$ to $x$. This path consists of 3 edges plus 3 levels of vertical paths or at most 9 edges, giving a delay of 5. If $(u', v')$ is not in $S$ and $(v', w')$ is in $S$, then let $z'$ be the vertex other than $v'$ that is a common cousin of $u'$ and $w'$. The edge $(u', z')$ is in $S$ and $(z', w')$ is not in $S$. There is a path from $u$ to $x$ consisting of a vertical path from $u$ to $u'$, the edge $(u', z')$, the edge from $z'$ to its parent, the sibling edge from the parent of $z'$ to $z$, and a (two-level) vertical path from $z$ to $x$. This path consists of 3 edges plus 3 levels of vertical paths or at most 9 edges, giving a delay of 5.

Now, consider the case that $(u', v')$ is a sibling edge. Let $z'$ be the sibling of $w'$ that is also a cousin of $u'$. If $(u', z')$ is in $S$, then there is a path from $u$ to $x$ consisting of a vertical path from $u$ to $u'$, the edge $(u', z')$, a vertical path from $z'$ to $z$, and a (two-level) vertical path from $z$ to $x$. This path consists of 1 edge plus 4 levels of vertical paths or at most 9 edges, giving a delay of 5. If $(u', z')$ is not in $S$, then there is a path from $u$ to $x$ consisting of a vertical path from $u$ to $u'$, the edge from $u'$ to its parent, the sibling edge from the parent of $u'$ to $z$, and a (two-level) vertical path from $z$ to $x$. This path consists of 2 edges plus 3 levels of vertical paths or at most 8 edges, giving a delay of 4.

Thus, the layout **ccsc** gives a delay of at most 6 or $k + 4$, since $k \geq 2$.

In the cases above, the horizontal paths of length 1, 2, 3, and 4 gave maximum delays of at most $k+2$, $k+3$, $k+4$, and $k+4$, respectively. As the total delay along the path in $S$ corresponding to $P$ is at most $\lfloor \frac{x-h-4}{k} \rfloor + 4 + d_h$, these give $\lfloor \frac{x-5}{k} \rfloor + 4 + k + 2$, $\lfloor \frac{x-6}{k} \rfloor + 4 + k + 3$, $\lfloor \frac{x-7}{k} \rfloor + 4 + k + 4$, and $\lfloor \frac{x-8}{k} \rfloor + 4 + k + 4$, respectively. Each of these quantities is at most $\frac{x}{k} + \frac{k^2+8k-7}{k}$ so $f(x) \leq \frac{k+1}{k} x + k + 8 - \frac{7}{k}$.

To bound the average degree, we note that most of the vertices that are not at the bottom level of the spanner are of degree 5. Similar to our analysis of the X-tree shrub spanner, relatively few vertices that are not at the bottom level of the pyramid spanner have degree $< 5$. We can, therefore, obtain a good upper bound on the average degree by assuming that all vertices not at the bottom level of the pyramid have degree 5.

Consider now the vertices at the bottom of the pyramid. By our construction, at most one vertex in each line in the $x_1$ dimension in each hedge (or partial hedge) is of degree 4. The other vertices at this level are of degree 3. Each line in the $x_2$ dimension at this level contains vertices that all have the same degree (except for its endpoints, which have lower degree). Thus, the proportion of degree 3 vertices to degree 4 vertices on this level is the same as the proportion along any line in the $x_1$ dimension at this level (except for the two lines along the boundary of the pyramid). Each of these lines in the $x_1$ dimension encounters at most $2^{h-1}$ vertices of degree 4. Thus, the proportion of degree $< 4$ to degree 4 vertices is at least 1:1.

As there are $4^h$ vertices at the bottom level and $\frac{4^h-1}{3}$ vertices not on the bottom level, the average degree is at most

$$\frac{3 \cdot \frac{1}{2} \cdot 4^h + 4 \cdot \frac{1}{2} \cdot 4^h + 5 \cdot \frac{4^h-1}{3}}{\frac{4^{h+1}-1}{3}} = \frac{\frac{9}{2} \cdot 4^h + 6 \cdot 4^h + 5 \cdot 4^h - 5}{4^{h+1} - 1}$$

$$= \frac{\frac{31}{2} \cdot 4^h - 5}{4^{h+1} - 1} = \frac{\frac{31}{8} \cdot 4^{h+1} - \frac{31}{8} - \frac{9}{8}}{4^{h+1} - 1} < \frac{31}{8} = 3\frac{7}{8}. \qquad \square$$

**4. Summary.** We showed how to construct two different types of $(2.5\sqrt{(3x+6)/4} + 6 + x)$-spanner with maximum degree 3 for two-dimensional grids. The constructions yield different average degrees. Further, we established a lower bound that shows that the delay of these spanners is within a constant factor of optimal. We showed how to construct a $(\frac{1}{k}x + k + 8 - \frac{7}{k} + x)$-spanner of the X-tree with maximum degree 3, and we established a lower bound that shows that the delay of this spanner is within a constant factor of optimal. We showed how to construct a $(2 + x)$-spanner of the pyramid with maximum degree 6 and a $(\frac{1}{k}x + k + 8 - \frac{7}{k} + x)$-spanner of the pyramid with maximum degree 5. We also established a lower bound that shows that the delay of the latter spanner is within a constant factor of optimal.

## REFERENCES

[1] L. Cai, *Tree 2-spanners*, Simon Fraser University Tech. Report No. 91-4, 1991.
[2] L. Cai and D. Corneil, *Tree spanners: An overview*, Congr. Numer., 88 (1992), pp. 65–76.
[3] A. L. Liestman and T. C. Shermer, *Grid spanners*, Networks, 23 (1993), pp. 122–133.
[4] ———, *Additive spanners for hypercubes*, Parallel Process Lett., 1 (1991), pp. 35–42.
[5] ———, *Additive graph spanners*, Networks, 23 (1993), pp. 343–364.
[6] D. Peleg and A. A. Schäffer, *Graph spanners*, J. Graph Theory, 13 (1989), pp. 99–116.
[7] D. Peleg and J. D. Ullman, *An optimal synchronizer for the hypercube*, in Proc. 6th ACM Symposium on Princ. Distr. Comp., 1987, pp. 77–85.
[8] D. Richards and A. L. Liestman, *Degree-constrained pyramid spanners*, J. Parallel Distrib. Comput., to appear.
[9] J. Soares, *Graph spanners: A survey*, Congr. Numer., 89 (1992), pp. 225–238.

SIAM J. DISC. MATH.
Vol. 8, No. 2, pp. 322–345, May 1995

# ANALYSIS OF A RECURRENCE ARISING FROM A CONSTRUCTION FOR NONBLOCKING NETWORKS*

NICHOLAS PIPPENGER[†]

**Abstract.** Define $f$ on the integers $n > 1$ by the recurrence $f(n) = \min\{n, \min_{m|n} 2f(m) + 3f(n/m)\}$. The function $f$ has $f(n) = n$ as its upper envelope, attained for all prime $n$. The goal of this paper is to determine the corresponding lower envelope. It is shown that this has the form $f(n) \sim C(\log n)^{1+1/\gamma}$ for certain constants $\gamma$ and $C$, in the sense that for any $\varepsilon > 0$, the inequality $f(n) \leq (C+\varepsilon)(\log n)^{1+1/\gamma}$ holds for infinitely many $n$, while $f(n) \leq (C-\varepsilon)(\log n)^{1+1/\gamma}$ holds for only finitely many. In fact, $\gamma = 0.7878\ldots$ is the unique real solution of the equation $2^{-\gamma} + 3^{-\gamma} = 1$, and $C = 1.5595\ldots$ is given by the expression $C = (\gamma(2^{-\gamma}\log 2^{\gamma} + 3^{-\gamma}\log 3^{\gamma})^{1/\gamma})/((\gamma+1)(15^{-\gamma}\log^{\gamma+1}\frac{5}{2} + 3^{-\gamma}\sum_{5\leq k\leq 7}\log^{\gamma+1}\frac{k+1}{k} + \sum_{8\leq k\leq 15}\log^{\gamma+1}\frac{k+1}{k})^{1/\gamma})$. This paper also considers the function $f_0$ defined by replacing the integers $n > 1$ with the reals $x > 1$ in the above recurrence: $f_0(x) = \min\{x, \inf_{1<y<x} 2f_0(y) + 3f_0(x/y)\}$. The author shows that $f_0(x) \sim C_0(\log x)^{1+1/\gamma}$, where $C_0 = 1.5586\ldots$ is given by $C_0 = 6e\,(2^{-\gamma}\log 2^{-\gamma} + 3^{-\gamma}\log 3^{-\gamma})^{1/\gamma}\,(\gamma/(\gamma+1))^{1+1/\gamma}$ and is smaller than $C$ by a factor of $0.9994\ldots$.

**Key words.** asymptotic analysis, recurrence relation

**AMS subject classification.** 26A12

## 1. Introduction.

Our goal in this paper is an analysis of the recurrence

$$f(n) = \min\left\{n, \min_{m|n} 2f(m) + 3f(n/m)\right\} \tag{1.1}$$

for the function $f : N \to N$, where $N$ denotes the set of integers exceeding 1. The value of $f(n)$ depends strongly on the factorization of $n$. Thus for example we have $f(n) = n$ whenever $n$ is prime, since then the inner minimization is over an empty set of factorizations. This example characterizes the "upper envelope" of $f$, since the outer minimization ensures that $f(n) \leq n$ always holds.

In the motivation for the study of this recurrence, which will be presented in §2, $f(n)$ is interpreted as a "cost" and $n$ as a "benefit." We are thus led to seek the corresponding "lower envelope" of the function $f$, where the relationship between cost and benefit is most favorable. Our main result, Theorem 6.1, shows that this lower envelope takes the form

$$f(n) \sim C(\log n)^{1+1/\gamma} \tag{1.2}$$

(for certain constants $\gamma$ and $C$), in the sense that for any $\varepsilon > 0$ the inequality

$$f(n) \leq (C + \varepsilon)(\log n)^{1+1/\gamma} \tag{1.3}$$

is satisfied for infinitely many values of $n$, while

$$f(n) \leq (C - \varepsilon)(\log n)^{1+1/\gamma} \tag{1.4}$$

*Received by the editors September 17, 1993; accepted for publication January 21, 1994. This research was partially supported by a Natural Sciences and Engineering Research Council of Canada Operating Grant.

[†]Department of Computer Science, The University of British Columbia, Vancouver, British Columbia V6T 1Z4, Canada (nicholas@cs.ubc.ca).

is satisfied for only finitely many. The constant $\gamma = 0.78788\ldots$ is the unique real solution of the equation

$$2^{-\gamma} + 3^{-\gamma} = 1, \tag{1.5}$$

while the constant $C = 1.5595\ldots$ is given by
(1.6)

$$C = \frac{\gamma\,(2^{-\gamma}\log 2^{\gamma} + 3^{-\gamma}\log 3^{\gamma})^{1/\gamma}}{(\gamma+1)\left(15^{-\gamma}\log^{\gamma+1}\frac{5}{2} + 3^{-\gamma}\sum_{5\le k\le 7}\log^{\gamma+1}\frac{k+1}{k} + \sum_{8\le k\le 15}\log^{\gamma+1}\frac{k+1}{k}\right)^{1/\gamma}}.$$

It may seem surprising that a recurrence as simple as (1.1) can give rise to an expression as complicated as (1.6); nevertheless, we shall find a simple interpretation for each of the twelve terms that are summed in the denominator.

In preparation for the derivation of our main result, it will be convenient to analyze some related recurrences that provide upper and lower bounds for $f$, while being much easier to analyze. First, for any integer $d > 1$, we may consider the function $f_d$ that is defined by the same recurrence as $f$ but with the domain being restricted from the set $N$ of all integers exceeding 1 to the set $N_d$ of all integral powers of $d$ exceeding 1:

$$f_d(n) = \min\left\{n, \min_{m|n} 2f_d(m) + 3f_d(n/m)\right\}. \tag{1.7}$$

The multiplicative semigroup formed by the integral powers of $d$ constitutes a subsemigroup of the multiplicative semigroup of integers. Thus we have $f_d(n) \ge f(n)$ wherever the left-hand side is defined, since any factorization that participates in the minimization on the left-hand side also participates on the right-hand side. On the other hand, the factorizations that participate on the left-hand side are sufficiently uniform as to eliminate the discrepancy between the upper and lower envelopes, so we shall obtain a simple asymptotic expression for $f_d$.

We shall show in Theorem 4.1 that for $d \ge 5$ we have

$$f_d(n) \sim C_d(\log n)^{1+1\gamma}, \tag{1.8}$$

where

$$C_d = \frac{4d\gamma}{\gamma+1}\left(\frac{2^{-\gamma}\log 2^{\gamma} + 3^{-\gamma}\log 3^{\gamma}}{\log^{\gamma+1} d}\right)^{1/\gamma}. \tag{1.9}$$

The expression (1.9) assumes its minimum for $d = 10$, with $C_{10} = 1.6296\ldots$.

For $2 \le d \le 4$ the situation is more complicated, since in these cases the first member of the outer minimization in (1.7) can minorize the second when $n$ is a power of $d$, whereas this occurs only for $n = d$ when $d \ge 5$. Nevertheless, we shall show in Theorem 4.2 that (1.8) continues to hold, with $C_2 = 1.5909\ldots$ given by

$$C_2 = \frac{\gamma}{\gamma+1}\left(\frac{2^{-\gamma}\log 2^{\gamma} + 3^{-\gamma}\log 3^{\gamma}}{(4^{-\gamma} + 12^{-\gamma})\log^{\gamma+1} 2}\right)^{1/\gamma}, \tag{1.10}$$

$C_3 = C_9 = 1.6311\ldots$, and $C_4 = 1.6867\ldots$ given by

$$C_4 = \frac{\gamma}{\gamma+1}\left(\frac{2^{-\gamma}\log 2^{\gamma} + 3^{-\gamma}\log 3^{\gamma}}{(28^{-\gamma} + 36^{-\gamma})\log^{\gamma+1} 4}\right)^{1/\gamma}. \tag{1.11}$$

Thus the minimum of $C_d$ over all $d$ occurs for $d = 2$.

Finally, we may consider the function $f_0$ that is defined by the same recurrence as $f$ but with the domain being extended from the set $N$ of all integers exceeding 1 to the set $N_0$ of all reals exceeding 1:

$$f_0(x) = \min \left\{ x, \inf_{1<y<x} 2f_0(y) + 3f_0(x/y) \right\}. \tag{1.12}$$

(The infimum in (1.12) is in fact achieved as a minimum, as will become clear from the analysis, but we shall not need this fact.) Here we have a supersemigroup of the multiplicative semigroup of integers, so we have $f_0(n) \le f(n)$ for all integers $n > 1$. Again the discrepancy between upper and lower envelopes disappears, and we obtain a simple asymptotic formula for $f_0$.

We shall show in Theorem 5.1 that

$$f_0(x) \sim C_0 (\log x)^{1+1/\gamma}, \tag{1.13}$$

where $C_0 = 1.5586\ldots$ is given by

$$C_0 = 6e\,(2^{-\gamma} \log 2^\gamma + 3^{-\gamma} \log 3^\gamma)^{1/\gamma} \left(\frac{\gamma}{\gamma+1}\right)^{1+1/\gamma}, \tag{1.14}$$

in which $e = 2.7182\ldots$ is the base of natural logarithms.

## 2. Nonblocking networks.

The analysis of the recurrence (1.1) may be followed without reference to or knowledge of nonblocking networks. For the sake of motivation, however, we shall derive the recurrence against its historical background.

A "network" is an interconnection of "nodes" by means of "switches." In a network there are some distinguished nodes called "inputs," some other distinguished nodes called "outputs," and some distinguished sets of switches called "routes," each of which forms a path from an input to an output. A network is "nonblocking" if, given any disjoint set of routes (no two of which have a node or switch in common) and given any free input and free output (neither of which are involved in any of the given routes), there is a free route (disjoint from the given routes) from the given input to the given output. (The knowledgeable reader will recognize here the definition of a "strictly" nonblocking network. As this is the only type with which we shall have to deal in this paper, we shall omit the qualification "strictly.")

One of the basic questions concerning nonblocking networks is: given integers $n > 1$ and $m > 1$, what is the smallest possible number $G(n, m)$ of switches in a non-blocking network with $n$ inputs and $m$ outputs? Since inputs and outputs appear symmetrically in the definitions, we have

$$G(n, m) = G(m, n) \tag{2.1}$$

by taking "mirror images."

A nonblocking network can be constructed by letting the inputs and outputs be the only nodes and by installing a separate switch between each input and each output. Such a network, which is called a "crossbar," shows that

$$G(n, m) \le nm. \tag{2.2}$$

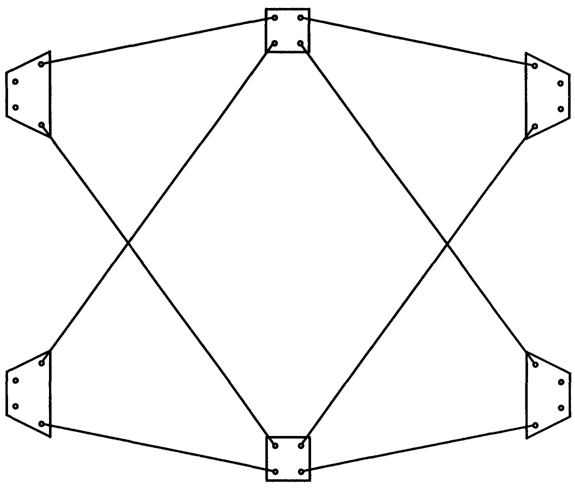


FIG. 1.

In 1953, Clos [Cl] introduced what has become the most widely known method for the construction of nonblocking networks. His idea is to construct a large nonblocking network by interconnecting smaller subnetworks. In his construction the subnetworks are arranged in three "stages," as shown in Fig. 1. The first stage, shown at the left, contains $a$ subnetworks, each with $b$ inputs and $2b$ outputs. The second stage contains $2b$ subnetworks, each having $a$ inputs and $a$ outputs. The inputs of the first-stage subnetworks are the inputs of the overall network; the outputs of the first-stage subnetworks are identified with (that is, connected by "wires" to) the inputs of the second-stage subnetworks, in such a way that each first-stage and each second-stage subnetwork have exactly one node in common. The third stage, shown on the right, contains $a$ subnetworks, each having $2b$ inputs and $b$ outputs. The outputs of the third-stage subnetworks are the outputs of the overall network; the outputs of the second-stage subnetworks are identified with the inputs of the third-stage subnetworks, in such a way that each second-stage and each third-stage subnetwork have exactly one node in common. Each route in the overall network consists of a route through a first-stage subnetwork, its extension through a second-stage subnetwork, and finally its extension through a third-stage subnetwork.

A simple argument based on the pigeon-hole principle shows that the overall network is nonblocking if each of the subnetworks is nonblocking. This construction thus shows that

$$
\begin{aligned}
G(ab, ab) &\le aG(b, 2b) + 2bG(a, a) + aG(2b, b) \\
&\le 2aG(b, 2b) + 2bG(a, a).
\end{aligned}
\tag{2.3}
$$

(The attentive reader may have noticed that the argument remains valid even if $2b$ is replaced by $2b - 1$. We shall ignore this sharpening of the inequality, however, as it leads off the path we wish to follow.)

If crossbars are used in each of the three stages and if the parameters $a$ and $b$ are each chosen to be about $n^{1/2}$, the resulting construction shows that $G(n, n) = O(n^{3/2})$. It is clear that further progress can be made by using the method recursively, but Clos did not succeed in finding the best way of doing this. In 1971, Cantor [Ca] presented the two principles that underlie the best recursive use of Clos's method. Firstly, since the subnetworks in the outer stages have inputs and outputs in the proportion

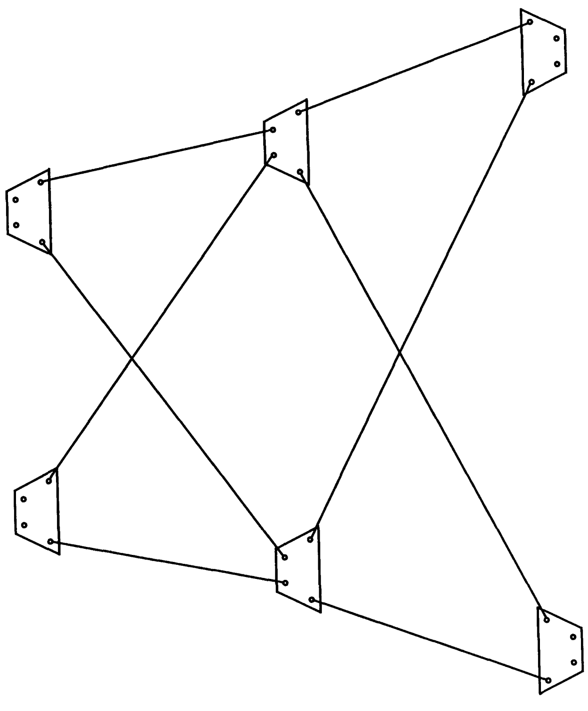


FIG. 2.

1 : 2 (or 2 : 1, which is equivalent by taking mirror images), the recursion should be based entirely on such networks. This can be accomplished by giving the inner subnetworks inputs and outputs in the proportion 1 : 2, whence the overall network will have inputs and outputs in the same proportion. When this has been done, the second-stage subnetworks have $a$ inputs and $2a$ outputs and there are $2a$ third-stage subnetworks. The resulting construction, shown in Fig. 2, shows that

$$\begin{aligned} G(ab, 2ab) &\le aG(b, 2b) + 2bG(a, 2a) + 2aG(2b, b) \\ &\le 3aG(b, 2b) + 2bG(a, 2a). \end{aligned} \tag{2.4}$$

If the parameters $a$ and $b$ were given equal values, the outer subnetworks would be more numerous than the inner ones in the proportion 3 : 2 and any diseconomy of scale would manifest itself more acutely in the outer stages. It follows that the sizes of the outer subnetworks should be reduced and those of the inner subnetworks increased. To discover the optimal choices of $a$ and $b$, let $F$ denote the the largest function defined on the integers exceeding 1 and satisfying the inequalities

$$F(n) \le 2n^2 \tag{2.5}$$

and

$$F(ab) \le 3aF(b) + 2bF(a). \tag{2.6}$$

Comparing (2.2) and (2.4) with (2.5) and (2.6), we see that $F(n)$ is the smallest possible number of switches in a nonblocking network built according to the construction of Clos and Cantor. Furthermore, if we set $f(n) = F(n)/2n$, we see that $f$ satisfies the recurrence (1.1). Thus the minimizations occurring in (1.1) correspond to the optimizations available in the construction of Clos and Cantor.

Cantor [Ca, §3] showed that for any $\varepsilon > 0$,

$$f(n) = O\big((\log n)^{1+1/\gamma+\varepsilon}\big) \tag{2.7}$$

for an infinite sequence of $n$, and Pippenger; [P1, §6] showed that

$$f(n) = O\big((\log n)^{1+1/\gamma}\big) \tag{2.8}$$

for an infinite sequence. Our Theorems 4.1, 4.2, and 6.1 can all be viewed as refinements of and complements to (2.7) and (2.8) for various sequences.

The construction for nonblocking networks that we study is not the best asymptotically. Indeed, Cantor ([Ca], §4) gave a construction using $O\big(n(\log n)^2\big)$ switches, and Bassalygo and Pinsker [BP] gave a probabilistic argument showing the existence of nonblocking networks with $O(n \log n)$ switches. By an old result of Shannon [S], the rate of growth of this last result is the best possible. The result of Bassalygo and Pinkser has since been obtained through an explicit construction; see Pippenger [P1] for a presentation of all these results.

It is interesting to note that the results of §4 for fixed $d > 1$ correspond to the assumption that all crossbars in a nonblocking network have certain fixed sizes, $d$ or a power of $d$; and it is curious that the choices $d = 10$ and $d = 2$ should have certain optimality properties, since precisely these values have been favored historically in the construction of telephone switching networks (following the widespread use of the decimal and binary number representations by humans and computers, respectively). The results of §5 similarly correspond to the assumption (contrary to fact) that crossbars could have any real (not necessarily integral) numbers of inputs and outputs; and it is curious how little could be gained in this way: $C$ and $C_0$ differ by less than one part in one thousand!

**3. Derivations.** In this section we shall reinterpret our problem in terms of trees, which will become the main objects of our attention in later sections. To see the relevance of trees, consider the task of proving that

$$f(n) \leq p \tag{3.1}$$

for some particular $n$ and $p$. If $n \leq p$, then (3.1) follows by the first member of the outer minimization of (1.1). Otherwise, we must have $f(n) = 2f(m) + 3f(n/m)$ for some $m \mid n$. In this case we can reduce the task of proving (3.1) to that of proving

$$f(m) \leq q \tag{3.2}$$

and

$$f(l) \leq r \tag{3.3}$$

for some $m$ and $l$ such that $ml = n$ and some $q$ and $r$ such that $p = 2q + 3r$. In either case, we may represent the proof of (3.1) in the form of a tree: in the first case the tree reduces to a single vertex, its root; in the second case, the root has two children, which are the roots of subtrees representing the proofs of (3.2) and (3.3). In the remainder of this section we present the combinatorial machinery that formalizes this representation.

For the purposes of this paper, the "infinite tree" is the set $V = \{2,3\}^*$ of finite words over the alphabet $\Sigma = \{2,3\}$. The words of $V$ are called "vertices." The empty word $\lambda$ is called the "root." If $v$ is a word, the word $v2$ is called its "left child" and $v3$ is called its "right child," $v$ is called the "parent" of $v2$ and $v3$, and $v2$ and $v3$ are called "siblings" of each other.

A "finite tree" (or simply a "tree") is a nonempty finite subset $T \subseteq V$ that is closed under taking parents and siblings. Every tree contains the root $\lambda$. If vertex $v$ has a child in a tree $T$, then both its children are in $T$ and $v$ is called an "internal vertex" of $T$. If $v$ belongs to $T$ but has no children in $T$, then $v$ is called a "leaf" of $T$. The number of internal vertices in a tree is one less than the number of leaves.

The "weight" $W(v)$ of a vertex $v$ is the product of the letters appearing in $v$, with each letter appearing as a factor with the same multiplicity that it has in $v$. (This definition accounts for our rather unorthodox use of 2 and 3 as the letters of a binary alphabet.)

A "derivation" $D = (T, l)$ is a tree $T = T_D$ together with an assignment $l = l_D$ of integers exceeding 1 to the leaves of $T$. If $D$ is a derivation, the integer assigned to a leaf $v$ will be called the "load" of $v$ and will be denoted $l_D(v)$. The "capacity" $L(D)$ of a derivation $D$ is the product of the loads of its leaves. The "cost" $C(D)$ of a derivation $D$ is the sum, over all leaves, of the product of the weight of the leaf and the load of the leaf.

The main result of this paper is based on the following observation: the solution $f(n)$ of the recurrence (1.1) is equal to the minimum cost of a derivation with capacity $n$. This is easily proved by the inductive argument sketched in the opening paragraph of this section.

Furthermore, we can extend this reinterpretation to the recurrences (1.7) and (1.12) simply by restricting or extending the set of allowable loads. Specifically, if we define a "$d$-derivation" for $d > 1$ to be a derivation in which all the loads are integral powers of $d$, then $f_d(n)$ is the minimum possible cost of a $d$-derivation with capacity $n$. Similarly, if we define a "0-derivation" to be like a derivation, except that the loads may be any reals exceeding 1, then $f_0(x)$ is the minimum possible cost of a 0-derivation with capacity $x$.

## 4. Integral powers of *d*.

In this section we shall analyze the recurrence (1.7), starting with the case $d \geq 5$; later we shall also consider $2 \leq d \leq 4$. The case $d \geq 5$ could actually be solved by reduction to a recurrence dealt with by Fredman and Knuth [FK], but we shall use a slightly different analysis in order to prepare for other cases treated later.

THEOREM 4.1. *For $d \geq 5$, the solution $f_d$ to the recurrence* (1.7) *satisfies*

$$f_d(n) \sim C_d (\log n)^{1+1\gamma}, \tag{4.1}$$

*where*

$$C_d = \frac{4d\gamma}{\gamma+1}\left(\frac{2^{-\gamma}\log 2^{\gamma} + 3^{-\gamma}\log 3^{\gamma}}{\log^{\gamma+1} d}\right)^{1/\gamma}. \tag{4.2}$$

As observed in §3, $f_d(n)$ is the minimum possible cost of a $d$-derivation of capacity $n$. When $d \geq 5$, our problem is simplified by the following observation: for every $n = d^k$, there exists a minimum-cost $d$-derivation of capacity $n$ in which the load of every leaf is $d$. To see this suppose that every minimal-cost $d$-derivation with capacity $n$ has a leaf with load at least $d^l$, where $l \geq 2$. Let $D$ be a $d$-derivation with capacity

$n$ and the minimum possible number of loads equal to $d^l$, and let $v$ be a leaf in $T_D$ with load equal to $l_D(v) = d^l$. Consider the derivation $D'$ obtained from $D$ by making $v$ an internal vertex with leaves as children. If we let $l_{D'}(v2) = d^{l-1}$ and $l_{D'}(v3) = d$, the capacity of $D'$ is the same as that of $D$. Furthermore, the cost of $D'$ is no greater than that of $D$, since the contribution $d^l W(v)$ to $D$ has been replaced by the contribution $(2d^{l-1} + 3d)W(v)$ to $D'$, and $2d^{l-1} + 3d \leq d^l$ when $d \geq 5$. This contradicts the assumption that $D$ has the minimum possible number of loads equal to $d^l$ and completes the proof of the observation.

We shall refer to the number of leaves in a tree as the "scale" of the tree and the sum of the weights of its leaves as its "total weight." When the load of every leaf is $d$, the capacity of a $d$-derivation is just $d^k$, where $k$ is the scale of its tree, and the cost of a $d$-derivation is just $d$ times the total weight of its tree. Thus a minimum-cost $d$-derivation is one based on a tree that, among those with a given scale, has the smallest possible total weight. This yields

$$f_d(d^k) = d\Psi(k), \tag{4.3}$$

where $\Psi(k)$ denotes the minimum possible total weight of leaves in a tree with $k$ leaves.

If $T$ is a tree, we shall call its set of internal vertices its "kernel" and denote it by $K(T)$. The kernel of a tree is closed under taking prefixes. Conversely, any set $K$ closed under taking prefixes is the kernel of a tree $T(K)$, obtained from $K$ by adjoining as leaves those vertices that are children of vertices in $K$ but do not themselves appear in $K$. Thus there is a one-to-one correspondence between trees and their kernels.

For any tree $T$, the set $T \setminus K(T)$ is the set of leaves of $T$. It will be called the "frontier" of $T$ and be denoted by $F(T)$.

If $v$ is any vertex, we have $W(v2) + W(v3) = 5W(v)$. Summing this identity over all $v \in K(T)$, we obtain

$$\sum_{u \in F(T)} W(u) = 1 + 4 \sum_{u \in K(T)} W(u),$$

since a leaf $u \in F(T)$ appears once as $v2$ or $v3$, the root $\lambda$ appears once as $v$, and each other internal vertex $u \in K(T)$ appears once as $v2$ or $v3$ and once as $v$. Thus, among trees of a given scale, those with the minimum total weight of their leaves are also those with the minimum total weight of their internal vertices. This yields

$$\Psi(k) = 1 + 4\Phi(k-1), \tag{4.4}$$

where $\Phi(k-1)$ denotes the minimum possible total weight of internal vertices in a tree with $k-1$ internal vertices.

We shall say that a tree is a "threshold tree" if the weight of every internal vertex is less than or equal to the weight of every leaf. If from the set of vertices we choose $k-1$ with the smallest weights, the resulting set of vertices is closed under taking prefixes, since the parent of a vertex $v$ has a strictly smaller weight than $v$. Such a set thus constitutes the set of internal vertices of a threshold tree with $k$ leaves. Thus there exist threshold trees of every scale. Furthermore, among trees of a given scale, threshold trees have minimum total weight of their internal vertices (since this is how their internal vertices were chosen) and, thus, have minimum total weight of their leaves. This yields

$$f_d(d^k) = d\big(1 + 4\Phi(k-1)\big), \tag{4.5}$$

where $\Phi(k-1)$ can now be interpreted as the sum of the weights of the $k-1$ smallest-weight vertices in the infinite tree. Our problem is now to determine the asymptotic behavior of $\Phi$.

Let $h(x)$ denote the number of vertices of the infinite tree having weight at most $x$. This function satisfies the asymptotic formula

$$h(x) \sim \frac{x^\gamma}{H}, \tag{4.6}$$

where

$$H = 2^{-\gamma}\log 2^\gamma + 3^{-\gamma}\log 3^\gamma. \tag{4.7}$$

(Information theorists will recognize $H$ as the entropy per independent flip of a biased coin that falls heads with probability $2^{-\gamma}$ and tails with probability $3^{-\gamma}$.) Formula (4.6) was proved by Fredman and Knuth [FK], who used an analytic argument; an elementary proof (in the technical sense) can be found in Pippenger [P2]. (This formula is the only point at which the present paper is not self-contained.)

Let $W_j$ denote the weight of the $j$th vertex of the infinite tree (when the vertices are arranged in nondecreasing order by weight). Inverting (4.6) by raising each side to the power $1/\gamma$, we see that

$$W_j \sim H^{1/\gamma} j^{1/\gamma}. \tag{4.8}$$

Summing over $j$ we obtain

$$\begin{aligned}\Phi(k) &= \sum_{1\le j\le k} W_j \\ &\sim \frac{\gamma}{\gamma+1} H^{1/\gamma} k^{1+1/\gamma}.\end{aligned} \tag{4.9}$$

Combining (4.9) with (4.5) and using $k = \log_d n$ yields Theorem 4.1.

For $2 \le d \le 4$, the analysis given above breaks down: there may be no minimum-cost $d$-derivations in which all loads equal $d$. This is best illustrated by the case $d = 2$, which we treat now.

THEOREM 4.2. *We have*

$$f_2(n) \sim C_2(\log n)^{1+1/\gamma}, \tag{4.10}$$

*where*

$$C_2 = \frac{\gamma}{\gamma+1}\left(\frac{2^{-\gamma}\log 2^\gamma + 3^{-\gamma}\log 3^\gamma}{(4^{-\gamma}+12^{-\gamma})\log^{\gamma+1} 2}\right)^{1/\gamma}. \tag{4.11}$$

We begin with a simple observation that we shall call the "ordering principle." Suppose that we fix a tree $T$ and a suite (that is, multiset) $S$ of loads and ask which loads should be assigned to which leaves in order to minimize the resulting cost. If $A_1 \le \cdots \le A_k$ are the weights in nondecreasing order and $B_1 \ge \cdots \ge B_k$ are the loads in non-increasing order, then the minimum possible cost is $\sum_{1\le j\le k} A_j B_j$ (this is simply Chebyshev's inequality). In particular, the smallest load should be assigned to the leaf with the largest weight, and vice versa.

Our next task is to determine what loads can appear on leaves of a minimum-cost 2-derivation; we claim that, excluding the trivial case $n = 2$, these are 4, 8, and 16.

Suppose there is an optimal 2-derivation $D$ in which some leaf has load 2. By the ordering principle, we may assume this leaf has the largest weight of any leaf in the tree; thus it is of the form $v3$ for some word $v$ (this is the point at which we must exclude the case $n = 2$), and $v2$ is also a leaf (for if it were the root of a subtree, all the leaves of this subtree would have weight larger than that of $v3$). Let $2^l$ be the load of the leaf $v2$ in $D$. Let $D'$ be the 2-derivation obtained from $D$ by making $v$ a leaf with load $2^{l+1}$. Then $D'$ has the same capacity as $D$. Furthermore, $D'$ has smaller cost than $D$, since the contribution $2^{l+1}W(v)$ of $v$ to $C(D')$ is less than the contribution $(2 \cdot 2^l + 3 \cdot 2)W(v)$ of $v2$ and $v3$ to $D$. This contradicts the assumption that an optimal 2-derivation can contain a leaf with load 2.

Now suppose that there is an optimal 2-derivation $D$ in which some leaf $v$ has load $2^l$, where $l \geq 5$. Let $D'$ be the 2-derivation obtained from $D$ by making $v$ an internal vertex, with leaves as children. If we let $l_{D'}(v2) = 2^{l-2}$ and $l_{D'}(v3) = 4$, then the capacity of $D'$ is the same as that of $D$. Furthermore, the cost of $D'$ is less than that of $D$, since the contribution $(2 \cdot 2^{l-2} + 3 \cdot 4)W(v)$ of $v2$ and $v3$ in $D'$ is less than the contribution $2^l W(v)$ of $v$ to $D$. This contradicts the assumption that an optimal 2-derivation can contain a leaf with load $2^l$, where $l \geq 5$, and completes the proof of the claim that optimal 2-derivations contain only 4, 8, and 16 as loads.

Next we claim that in an optimal 2-derivation, no leaf of the form $v2$ (that is, no "left leaf") can have load 4. Suppose that $D$ is an optimal 2-derivation in which $l_D(v2) = 4$. If the subtree rooted at $v3$ has capacity greater than 4, then we may obtain a 2-derivation with the same capacity as, but lower cost than, $D$ by exchanging the subtrees rooted at $v2$ and $v3$. On the other hand, if $v3$ is a leaf with load 4, we may obtain a 2-derivation with the same capacity as, but lower cost than, $D$ by making $v$ a leaf with load 16. In either case we obtain a contradiction, proving that no left leaf can have load 4.

In what follows we shall confine our attention to 2-derivations in which all loads are 4, 8, and 16 and in which no left leaf has load 4; we shall call these "admissible" 2-derivations. Define $D_0$ to be the admissible 2-derivation in which the root $\lambda$ is the only leaf, with load 8. Now consider three operations, which we shall call "promotions," that transform admissible 2-derivations into other admissible 2-derivations. The promotion $(v, 4)$ will be applicable to any admissible 2-derivation $D$ in which $v$ is a leaf with $l_D(v) = 4$; the result of applying $(v, 4)$ to $D$ is the admissible 2-derivation obtained from $D$ by increasing the load of $v$ to 8. The promotion $(v, 8)$ will be applicable to any admissible 2-derivation $D$ in which $v$ is a leaf with $l_D(v) = 8$; the result of applying $(v, 8)$ to $D$ is the admissible 2-derivation obtained from $D$ by increasing the load of $v$ to 16. The promotion $(v, 16)$ will be applicable to any admissible 2-derivation $D$ in which $v$ is a leaf with $l_D(v) = 16$; the result of applying $(v, 16)$ to $D$ is the admissible 2-derivation obtained from $D$ by making $v$ an internal vertex with leaves as children, assigning 8 as the load of $v2$ and 4 as the load of $v3$.

Any admissible 2-derivation $D$ with capacity at least 8 can be obtained by starting with $D_0$ and applying a sequence of promotions; this is easily proved by induction on the capacity of $D$. [The basis is capacity 8. The inductive step breaks into three cases: if $D$ has a leaf $v$ with load 16, then $D$ can be obtained by applying promotion $(v, 8)$ to an admissible 2-derivation with one-half the capacity (which can by the inductive hypothesis be obtained from $D_0$ by promotions); if $D$ has a right leaf $v$ with load 8, then $D$ can be obtained by applying promotion $(v, 4)$ to an admissible 2-derivation with one-half the capacity; and otherwise $D$, if it is not the basis, must contain an internal vertex $v$ with leaves as children, with 8 as load of $v2$ and 4 as load of $v3$, so that $D$ can be obtained by applying promotion $(v, 16)$ to an admissible 2-derivation with one-half the capacity.]

Any promotion doubles the capacity of the admissible 2-derivation to which it is applied. We shall assign a "cost" $C(P)$ to each promotion $P$ as follows: the cost of the promotion $(v, 4)$ is $4W(v)$, the cost of $(v, 8)$ is $8W(v)$, and the cost of $(v, 16)$ is $12W(v)$. Then if application of promotion $P$ to admissible 2-derivation $D$ yields $D'$, we have $C(D') = C(D) + C(P)$.

Among promotions, some are prerequisite to others: the promotion $(v, 8)$ is prerequisite to $(v, 16)$, the promotion $(v, 16)$ is prerequisite to both $(v2, 8)$ and $(v3, 4)$, and the promotion $(v3, 4)$ is prerequisite to $(v3, 8)$. In every case, however, if $P$ is prerequisite to $Q$, then the cost of $P$ is at most the cost of $Q$. In particular, we can order all possible promotions in a sequence $P_1, P_2, \ldots, P_j, \ldots$ in such a way that: (1) the costs are nondecreasing, and (2) each promotion is preceded by all of its prerequisites. It follows that the result of applying $P_1, \ldots, P_{k-3}$ in order to $D_0$ is a minimum-cost 2-derivation with capacity $2^k$. This yields

$$f_2(2^k) = 8 + \sum_{1 \leq j \leq k-3} C(P_j). \tag{4.12}$$

Our problem is now to determine the asymptotic behavior of $C(P_j)$.

Let $g(x)$ denote the number of promotions with cost at most $x$. We can write

$$g(x) = g_4(x) + g_8(x) + g_{16}(x), \tag{4.13}$$

where $g_4(x)$, $g_8(x)$, and $g_{16}(x)$ denote the numbers of promotions of the form $(v, 4)$, $(v, 8)$, and $(v, 16)$, respectively, with cost at most $x$. Since the cost of $(v, 16)$ is $12W(v)$, we have

$$g_{16}(x) = h(x/12). \tag{4.14}$$

Since the cost of $(v, 8)$ is $8W(v)$, we have

$$g_8(x) = h(x/8). \tag{4.15}$$

There is a promotion $(v, 4)$ if and only if $v$ is of the form $u3$; since the cost of $(v, 4)$ is $4W(v) = 12W(u)$, we have

$$g_4(x) = h(x/12). \tag{4.16}$$

This yields

$$\begin{aligned} g(x) &= 2h(x/12) + h(x/8) \\ &\sim (2 \cdot 12^{-\gamma} + 8^{-\gamma}) \frac{x^\gamma}{H} \\ &\sim (12^{-\gamma} + 4^{-\gamma}) \frac{x^\gamma}{H}, \end{aligned} \tag{4.17}$$

where we have used the identity (1.5) to obtain the last line from its predecessor.

Inverting (4.17) by raising each side to the power $1/\gamma$, we see that

$$C(P_j) \sim \left( \frac{H}{12^{-\gamma} + 4^{-\gamma}} \right)^{1/\gamma} j^{1/\gamma}. \tag{4.18}$$

Substituting this formula in (4.12) and summing yields (4.10) and (4.11), completing the proof of Theorem 4.2.

It is worth observing that the promotion $(v, 16)$ that creates a load of 4 has the same cost as the promotion $(v3, 4)$ that destroys the load of 4. It follows that the sequence of optimal promotions can be arranged so that the resulting optimal 2-derivations each have at most one leaf with load 4. Thus we can arrange that "almost all" the loads in an optimal 2-derivation are either 8 or 16.

The cases $d = 3$ and $d = 4$ are similar to $d = 2$, and we shall only describe the key points in the analyses. For $d = 3$, we easily show that no optimal 3-derivation can have a load as large as 81 and need not have any load as large as 27 (since this can be replaced without increasing the cost by children with loads 9 and 3). Futhermore, no left leaf can have a load of 3. Thus we need only consider "admissible" 3-derivations in which all loads are either 3 or 9, and no left leaf has a load of 3. We can analyze these by introducing promotions as before. We then observe that the promotion that creates a load of 3 has the same cost as the promotion that destroys the load of 3. Thus we can arrange that optimal 3-derivations have at most one leaf with load 3. Since almost all the loads are then 9, we obtain the same asymptotic result as in the case $d = 9$: $C_3 = C_9$.

For $d = 4$, a similar analysis shows that all loads in an optimal 4-derivation must be either 4 or 16. Furthermore, no left leaf can have a load of 4. We can then continue the analysis using promotions, and the result is

$$C_4 = \frac{\gamma}{\gamma+1}\left(\frac{2^{-\gamma}\log 2^{\gamma} + 3^{-\gamma}\log 3^{\gamma}}{(28^{-\gamma}+36^{-\gamma})\log^{\gamma+1} 4}\right)^{1/\gamma}. \tag{4.19}$$

It is worthwhile observing that for $2 \le d \le 4$ the trees underlying optimal $d$-derivations are threshold trees, just as they were for $d \ge 5$; this is easily seen by considering the promotions that increase the number of leaves in the tree.

**5. Reals.** In this section we shall analyze the recurrence (1.12) obtained by eliminating the integrality constraint from (1.1).

THEOREM 5.1. *We have*

$$f_0(x) \sim C_0(\log x)^{1+1/\gamma}, \tag{5.1}$$

*where* $C_0 = 1.5586\ldots$ *is given by*

$$C_0 = 6e\,(2^{-\gamma}\log 2^{\gamma} + 3^{-\gamma}\log 3^{\gamma})^{1/\gamma}\left(\frac{\gamma}{\gamma+1}\right)^{1+1/\gamma}. \tag{5.2}$$

As indicated in §3, our quest is for optimal 0-derivations. To determine these, let us fix a tree $T$ and ask how the loads of its leaves should be assigned so as to minimize the cost, while achieving a prescribed capacity. (Later we shall determine how the tree $T$ should be chosen.)

We first claim that, among 0-derivations based on a prescribed tree $T$ and having a prescribed capacity $x$, a 0-derivation $D$ with minimum cost must be such that there exists a constant $c$ such that for all leaves $v \in F(T_D)$,

$$l_D(v)W(v) = c. \tag{5.3}$$

Thus the loads must vary as the reciprocal of the weights of their leaves. To see this, suppose to the contrary that $l_D(u)W(u) > l_D(v)W(v)$ for some leaves $u$ and $v$.

Set $c' = \sqrt{l_D(u)W(u)l_D(v)W(v)}$, and let $D'$ be the 0-derivation obtained from $D$ by changing the loads of $u$ and $v$ to $l_{D'}(u) = c'/W(u)$ and $l_{D'}(v) = c'/W(v)$. Then $D'$ has the same capacity as $D$, but lower cost (as follows from the inequality between geometric and arithmetic means). This contradiction proves the claim.

Next we shall ask which tree $T$, among those with $k$ leaves, should be used to construct an optimal 0-derivation. (Later we shall determine how $k$ should be chosen.)

If the capacity of $D$ is to be $x$, we must have

$$\prod_{v \in F(T_D)} l_D(v) = x. \tag{5.4}$$

Multiplying (5.3) over all $v \in F(T_D)$ yields

$$\begin{aligned} c^k &= \prod_{v \in F(T_D)} l_D(v)W(v) \\ &= x \prod_{v \in F(T_D)} W(v), \end{aligned}$$

and thus

$$c = x^{1/k} \left( \prod_{v \in F(T_D)} W(v) \right)^{1/k}. \tag{5.5}$$

Since each leaf contributes $c$ to the cost of $D$, we have

$$C(D) = kx^{1/k} \left( \prod_{v \in F(T_D)} W(v) \right)^{1/k}. \tag{5.6}$$

Thus the optimal tree $T$ is one that minimizes the geometric mean of the weights of the leaves and, therefore, given that the number of leaves is fixed, minimizes the product of the weights. And since the logarithm is an increasing function, it is equivalent to minimize the sum of the logarithms of the weights of the leaves.

If $v$ is any vertex, we have $\log W(v2) + \log W(v3) = \log 6 + 2 \log W(v)$. Summing this identity over all $v \in K(T)$, we obtain

$$\sum_{u \in F(T)} \log W(u) = \log 1 + (k-1)\log 6 + \sum_{u \in K(T)} \log W(u), \tag{5.7}$$

since each leaf $u \in F(T)$ appears once as $v2$ or $v3$, the root $\lambda$ appears once as $v$, and each other internal vertex $u \in K(T)$ appears once as $v2$ or $v3$ and once as $v$. When $k$ is fixed, the right-hand side of (5.7) is minimized by choosing the $k-1$ vertices $v$ with the smallest $W(v)$ to be the internal vertices in $K(T)$. Thus threshold trees, which emerged as optimal for $d$-derivations ($d > 1$), are also optimal for 0-derivations.

It remains to determine the optimal value of $k$ as a function of the capacity $x$. To do this we shall determine the asymptotic behavior of the geometric mean $U(k)$ of the weights of the leaves in a threshold tree with $k$ leaves. We shall show that

$$U(k) \sim 6e^{-1/\gamma}H^{1/\gamma}k^{1/\gamma}. \tag{5.8}$$

Using (5.7) we have

$$\begin{aligned}
U(k) &= \exp \frac{1}{k} \sum_{v \in F(T)} \log W(v) \\
&= \exp \frac{1}{k} \left( (k-1) \log 6 + \sum_{v \in K(T)} \log W(v) \right) \\
&= \exp \frac{1}{k} \left( (k-1) \log 6 + \sum_{1 \le j \le k-1} \log W_j \right).
\end{aligned} \tag{5.9}$$

From (4.8) we obtain

$$\log W_j = \frac{1}{\gamma} \log j + \frac{1}{\gamma} \log H + o(1). \tag{5.10}$$

Substituting (5.10) into (5.9) and estimating the sum by an integral yields (5.8).

From (5.8) we can complete the proof of Theorem 5.1 as follows. From (5.6) we have

$$\begin{aligned}
C(D) &= k x^{1/k} U(k) \\
&\sim 6 e^{-1/\gamma} H^{1/\gamma} k^{1+1/\gamma} x^{1/k}.
\end{aligned} \tag{5.11}$$

Choosing $k$ to minimize $k^{1+1/\gamma} x^{1/k}$ yields

$$k \sim \frac{\gamma + 1}{\gamma} \log x \tag{5.12}$$

and

$$k^{1+1/\gamma} x^{1/k} \sim \left( \frac{e\gamma}{\gamma + 1} \right)^{1+1/\gamma} (\log x)^{1+1/\gamma}. \tag{5.13}$$

Substituting (5.13) into (5.11) yields

$$C(D) \sim 6eH^{1/\gamma} \left( \frac{\gamma}{\gamma + 1} \right)^{1+1/\gamma} (\log x)^{1+1/\gamma}, \tag{5.14}$$

which completes the proof of Theorem 5.1.

## 6. Integers.

We arrive in this section at our main result, the solution of the recurrence (1.1).

THEOREM 6.1. *For every $\varepsilon > 0$, we have*

$$f(n) \le (C + \varepsilon)(\log n)^{1+1/\gamma} \tag{6.1}$$

*for infinitely many values of n but*

$$f(n) \le (C - \varepsilon)(\log n)^{1+1/\gamma} \tag{6.2}$$

*for only finitely many, where*
(6.3)

$$C = \frac{\gamma\,(2^{-\gamma}\log 2^{\gamma} + 3^{-\gamma}\log 3^{\gamma})^{1/\gamma}}{(\gamma+1)\left(15^{-\gamma}\log^{\gamma+1}\frac{5}{2} + 3^{-\gamma}\sum_{5\le k\le 7}\log^{\gamma+1}\frac{k+1}{k} + \sum_{8\le k\le 15}\log^{\gamma+1}\frac{k+1}{k}\right)^{1/\gamma}}.$$

Because $f(n)$ is large when $n$ is prime, we must focus attention on the lower envelope. We do this by defining

$$C' = \liminf_{n\to\infty} f(n)/(\log n)^{1+1/\gamma} \tag{6.4}$$

so that our task is to prove that $C' = C$.

Let us say that an integer $n$ is "good" if there is no larger integer $m > n$ such that $f(m) \le f(n)$. If $n$ is not good, then for some larger $m$ we have

$$f(m)/(\log m)^{1+1/\gamma} < f(n)/(\log n)^{1+1/\gamma}.$$

Thus the *limes inferior* in (6.4) remains unchanged if we confine attention to good $n$.

We begin as in §4 with an analysis of the possible load values; we shall not obtain the sharpest bounds here but merely aim to reduce the range that must be considered later. Let us consider a minimum-cost derivation $D$ for a good integer $n$; and let us further suppose that, among derivations of this minimum cost, $D$ has the largest possible number of leaves.

First, we claim that $D$ can have no load as small as 2. For if any leaf had load 2, this would certainly have to be the case for the leaf $v3$ of largest weight (by the ordering principle), and the sibling of $v3$ is another leaf $v2$ (else it would subtend a leaf of greater weight than $v3$). Suppose the load of $v2$ is $l$. Then the derivation obtained from $D$ by making $v$ a leaf with load $2l$ would have the same capacity as, but lower cost than, $D$.

Second, $D$ cannot have a leaf $v$ with load of 24: replacing $v$ by leaves $v2$ with load 6 and $v3$ with load 4 would leave the capacity and cost unchanged but increase the number of leaves.

Third, $D$ cannot have a leaf $v$ with load of 25: replacing $v$ by leaves $v2$ with load 5 and $v3$ with load 5 would leave the capacity and cost unchanged but increase the number of leaves.

Finally, $D$ cannot have a leaf $v$ with load $l$ as large as 26. To see this, it will suffice to show that we can find integers $i$ and $j$ such that $ij > l$ and $2i + 3j = l$, for then we could replace $v$ by leaves $v2$ with load $i$ and $v3$ with load $j$ and increase the capacity while leaving the cost unchanged; this contradicts the assumption that the capacity of $D$ is good.

If we plot the line $2i + 3j = l$ and the hyperbola $ij = l$ in the real $(i, j)$ plane, they intersect at two points with $i$-coordinates

$$i = \frac{l \pm \sqrt{l^2 - 24l}}{4}. \tag{6.5}$$

The difference between these $i$-coordinates is

$$\Delta i = \frac{\sqrt{l^2 - 24l}}{2}. \tag{6.6}$$

We will have $\Delta i > 3$ if $l^2 - 24l > 36$; this in turn holds when $l > 12 + 2\sqrt{45}$ and, thus, certainly when $l \geq 12 + 2\sqrt{49} = 26$.

The line $2i + 3j = l$ contains infinitely many lattice points (points with integral coordinates); the $i$-coordinates of successive such lattice points differ by 3, since adding 3 to $i$ and subtracting 2 from $j$ leaves the sum $2i + 3j$ unchanged. Thus there must be a lattice point whose $i$-coordinate lies in the interval whose endpoints are given by (6.5). For this point $(i, j)$ we have $2i + 3j = l$ and $ij > l$, as desired.

Thus every load on an optimal derivation with a good capacity and a maximal number of leaves is at least 3 and at most 23. We also claim that, in such a derivation $D$, if $v$ has leaves $v2$ with load $i$ and $v3$ with load $j$ for children, then $2i + 3j \geq 24$. For the maximum of $ij$ subject to the constraint $2i + 3j = l$ is $l^2/24$. Thus if $2i + 3j < 24$, the derivation obtained from $D$ by making $v$ a leaf with load $ij$ will have the same capacity as, but smaller cost than, $D$.

We next claim that the tree $T$ underlying the derivation $D$ is a threshold tree. Suppose to the contrary that $T$ contains an internal vertex $u$ and a leaf $v$ with $W(u) > W(v)$. We may assume that $u$ has leaves $u2$ and $u3$ as children (since if not we may transfer attention from $u$ to one of its children). Let $h$, $i$, and $j$ be the loads of the leaves $v$, $u2$, and $u3$, respectively. As we have seen above, we must have $h \leq 23$ and $2i + 3j \geq 24$. Thus we obtain

$$-h + 2i + 3j > 0. \tag{6.7}$$

Consider now the tree $T'$ obtained from $T$ by making $u$ a leaf and making $v$ an internal vertex with children $v2$ and $v3$ as leaves. Now let us create a derivation $D'$ from the tree $T'$ by assigning the loads $h$, $i$, and $j$ in some order to the leaves $u$, $v2$, and $v3$ and letting the loads of all other leaves be the same as in $D$. Then $D'$ has the same capacity as $D$. We shall show that there is some order of assignment that results in $D'$ having a smaller cost than $D$. The analysis breaks into three cases, depending on how $W(u)$ ranks among $2W(v) < 3W(v)$.

First, suppose that $3W(v) < W(u)$. Then by the ordering principle we should assign $h$, $i$, and $j$ to $v2$, $v3$, and $u$, respectively. These three loads contribute $hW(v) + i2W(u) + j3W(u)$ to $C(D)$ and $h2W(v) + i3W(v) + jW(u)$ to $C(D')$. If this does not decrease the cost, that is, if $C(D') - C(D) \geq 0$, then

$$hW(v) + i\big(3W(v) - 2W(u)\big) - j2W(u) \geq 0. \tag{6.8}$$

Multiplying (6.7) by $W(v)$ and adding the result to (6.8) yields

$$i\big(5W(v) - 2W(u)\big) + j\big(3W(v) - 2W(u)\big) > 0. \tag{6.9}$$

This is a contradiction, since $3W(v) < W(u)$ implies that the coefficients of $i$ and $j$ are each strictly negative.

Secondly, suppose that $2W(v) < W(u) \leq 3W(v)$. Then we assign $h$, $i$, and $j$ to $v2$, $u$, and $v3$, respectively. These three loads contribute $hW(v) + i2W(u) + j3W(u)$ to $C(D)$ and $h2W(v) + iW(u) + j3W(v)$ to $C(D')$. If this does not decrease the cost, then

$$hW(v) - iW(u) + j\big(3W(v) - 3W(u)\big) \geq 0. \tag{6.10}$$

Multiplying (6.7) by $W(v)$ and adding the result to (6.10) yields

$$i\big(2W(v) - W(u)\big) + j\big(6W(v) - 3W(u)\big) > 0. \tag{6.11}$$

This is a contradiction, since $2W(v) < W(u)$ implies that the coefficients of $i$ and $j$ are each strictly negative.

Finally, suppose that $W(v) < W(u) \leq 2W(v)$. Then we assign $h$, $i$, and $j$ to $u$, $v2$, and $v3$, respectively. These three loads contribute $hW(v) + i2W(u) + j3W(u)$ to $C(D)$ and $hW(u) + i2W(v) + j3W(v)$ to $C(D')$. If this does not decrease the cost, then

$$(6.12) \qquad -h\big(W(v) - W(u)\big) + 2i\big(W(v) - W(u)\big) + 3j\big(W(v) - W(u)\big) \geq 0.$$

But this contradicts (6.7), since $W(v) < W(u)$. Thus the assumption that an internal vertex $v$ has greater weight than a leaf $u$ leads to a contradiction, completing the proof that threshold trees are optimal.

Now that we know that optimal derivations are based on threshold trees and that their loads are at least 3 and at most 23, it remains to determine the number of leaves that should be assigned each of these loads (since, then, the ordering principle will tell us which loads to assign to which leaves). Let $T$ be a threshold tree, and let $y$ denote the largest weight of an internal vertex. We shall renormalize the weights of the leaves by setting $\eta_v = W(v)/y$ for each leaf $v \in F(T)$. Then we have $\eta_v \geq 1$, since $T$ is a threshold tree. Furthermore, we have $\eta_v \leq 3$, since the weight of a leaf is at most thrice the weight of its parent, which is an internal vertex. Finally, we have $\eta_v \leq 2$ unless $v$ is a "right" leaf (that is, a leaf of the form $u3$), since the weight of a left leaf is at most twice the weight of its parent, which is an internal vertex.

We shall show that, if we choose a leaf $v$ at random from a threshold tree with $k$ leaves, with all $k$ leaves being equally likely, the value of $\eta_v$ has a distribution that tends as $k \to \infty$ to a particular density function on the interval $1 \leq \eta \leq 3$, which is continuous except for a single jump at $\eta = 2$.

First, let us fix $\eta$ and $\varepsilon$ such that $2 < \eta < \eta + \varepsilon < 3$ and consider the number $E(y, \eta, \varepsilon)$ of leaves $v$ such that $\eta < \eta_v \leq \eta + \varepsilon$. Such leaves are right leaves (since $\eta_v > 2$) and are in one-to-one correspondence with internal vertices $u$ such that $\eta y/3 < W(u) \leq (\eta + \varepsilon)y/3$. Using (4.6), we obtain

$$(6.13) \qquad \begin{aligned} E(y,\eta,\varepsilon) &= h\big((\eta+\varepsilon)y/3\big) - h\big(\eta y/3\big) \\ &\sim \frac{\big((\eta+\varepsilon)y/3\big)^\gamma}{H} - \frac{\big(\eta y/3\big)^\gamma}{H} \\ &\sim \frac{y^\gamma}{H}\big(3^{-\gamma}\gamma\eta^{\gamma-1}\varepsilon + O(\varepsilon^2)\big). \end{aligned}$$

Again using (4.6), we have $k \sim y^\gamma/H$. Thus the distribution of leaves in the interval $2 < \eta < 3$ aymptotically follows the density function

$$(6.14) \qquad \phi(\eta) = 3^{-\gamma}\gamma\eta^{\gamma-1}.$$

We have normalized $\phi$ so that

$$(6.15) \qquad \int_2^3 \phi(\eta)\, d\eta = 1 - (2/3)^\gamma;$$

the reason for this will become clear shortly.

Next, let us fix $\eta$ and $\varepsilon$ such that $1 < \eta < \eta + \varepsilon < 2$ and consider the number $E(y, \eta, \varepsilon)$ of leaves $v$ such that $\eta < \eta_v \leq \eta + \varepsilon$. Such leaves may be either right leaves or left leaves. The right leaves are in one-to-one correspondence with internal vertices $u$ such that $\eta y/3 < W(u) \le (\eta+\varepsilon)y/3$, and the left leaves are in one-to-one correspondence with internal vertices $u$ such that $\eta y/2 < W(u) \le (\eta+\varepsilon)y/2$. Using (4.6) and (1.5), we obtain

$$\begin{aligned} E(y,\eta,\varepsilon) &= h\big((\eta+\varepsilon)y/3\big) - h\big(\eta y/3\big) \\ &\quad + h\big((\eta+\varepsilon)y/2\big) - h\big(\eta y/2\big) \\ &\sim \frac{\big((\eta+\varepsilon)y/3\big)^\gamma}{H} - \frac{\big(\eta y/3\big)^\gamma}{H}. \\ &\quad + \frac{\big((\eta+\varepsilon)y/2\big)^\gamma}{H} - \frac{\big(\eta y/2\big)^\gamma}{H} \\ &\sim \frac{y^\gamma}{H}\big(\gamma\eta^{\gamma-1}\varepsilon + O(\varepsilon^2)\big). \end{aligned} \tag{6.16}$$

Thus the distribution of leaves in the interval $1 < \eta < 2$ aymptotically follows the density function

$$\phi(\eta) = \gamma\eta^{\gamma-1}. \tag{6.17}$$

We have normalized $\phi$ so that

$$\int_1^2 \phi(\eta)\,d\eta = 2^\gamma - 1; \tag{6.18}$$

from (6.15), (6.18), and (1.5) we have $\int_1^3 \phi(\eta)\,d\eta = 1$, so $\phi$ can be viewed as a probability density function on the interval $1 < \eta < 3$.

For each good integer $n$, let $D(n)$ denote an optimal derivation with capacity $n$. Let $k(n)$ denote the number of leaves in $T_{D(n)}$. For $3 \le m \le 23$, let $k_m(n)$ denote the number of leaves $v$ of $T_{D(n)}$ such that $l_{D(n)}(v) = m$ and let $\mu_m(n) = k_m(n)/k(n)$ denote the fraction of such leaves. From the sequence of good integers, let us extract an infinite subsequence of "special" integers such that, as $n$ runs through the special integers: (1) $f(n)/(\log n)^{1+1/\gamma}$ tends to $C'$ (as defined in (6.4)), and (2) for each $m$ in the range $3 \le m \le 23$, $\mu_m(n)$ tends to a limit $\mu_m$. Condition (1) can be fulfilled by the definition of $C'$, and condition (2) because for each of the finitely many values of $m$, $\mu_m(n)$ varies in the compact interval $0 \le \mu_m(n) \le 1$. Henceforth we confine our attention to these special $n$. Of course, we have

$$\sum_{3\le m\le 23} \mu_m = 1. \tag{6.19}$$

For each $m$, define $\alpha_m$ and $\beta_m$ such that

$$\int_1^{\alpha_m} \phi(\eta)\,d\eta = \sum_{m<l} \mu_l \tag{6.20}$$

and

$$\int_{\beta_m}^3 \phi(\eta)\,d\eta = \sum_{l<m} \mu_l. \tag{6.21}$$

Let $M$ denote the set of $m$ such that $\mu_m > 0$. Then we have $1 \le \alpha_m \le \beta_m \le 3$, and $\alpha_m < \beta_m$ if and only if $m \in M$. The half-open intervals $(\alpha_m, \beta_m]$ for $m \in M$ form a partition of the interval $(1,3]$. Thus for $1 < \eta \leq 3$ we may define a non-increasing left-continuous step function $\psi$ on the interval $(1,3]$ by letting $\psi(\eta)$ be the unique value of $m$ such that $\alpha_m < \eta \leq \beta_m$.

For special $n$ we have

$$\log n = \sum_{v \in F(T_D(n))} \log l_{D(n)}(v) \tag{6.22}$$

and

$$f(n) = \sum_{v \in F(T_D(n))} l_{D(n)}(v) W(v). \tag{6.23}$$

The limiting distribution $\phi$ of the weights of leaves in threshold trees, the ordering principle, and the definition of $\psi$ allow us to express the asymptotic behavior of the sums in (6.22) and (6.23) using integrals as

$$\log n \sim k \int_1^3 \phi(\eta) \log \psi(\eta)\, d\eta \tag{6.24}$$

and

$$f(n) \sim ky \int_1^3 \phi(\eta) \eta \psi(\eta)\, d\eta. \tag{6.25}$$

Since $k \sim y^\gamma / H$ and $f(n)/(\log n)^{1+1/\gamma} \sim C'$, we conclude that

$$C' = \frac{H^{1/\gamma} P}{Q^{1+1/\gamma}}, \tag{6.26}$$

where

$$P = \int_1^3 \phi(\eta) \eta \psi(\eta)\, d\eta \tag{6.27}$$

and

$$Q = \int_1^3 \phi(\eta) \log \psi(\eta)\, d\eta. \tag{6.28}$$

Now if we let $\psi$ be any nonincreasing left-continuous step function defined on $(1,3]$ and taking values in $\{3, \ldots, 23\}$, then for each $m$ in the range $M$ of $\psi$ there $\psi(\eta)$ takes on the value $m$ for $\eta$ in an interval of the form $(\alpha_m, \beta_m]$. From any threshold tree $T_k$ with $k$ leaves and threshold $y$, we can obtain a derivation $D_k$ by assigning to each leaf $v \in F(T_k)$ the load $\psi\big(W(v)/y\big)$. Letting $k$ (and with it $y$) tend to infinity, we obtain a sequence of derivations with capacities

$$\log n_k \sim k \int_1^3 \phi(\eta) \log \psi(\eta)\, d\eta \tag{6.29}$$

and costs

$$C(D_k) \sim ky \int_1^3 \phi(\eta) \eta \psi(\eta)\, d\eta. \tag{6.30}$$

Since we must have $f(n_k) \leq C(D_k)$, we conclude that

$$C' = \min_{\psi} \Gamma(\psi), \tag{6.31}$$

where

$$\Gamma(\psi) = \frac{H^{1/\gamma} P(\psi)}{Q(\psi)^{1+1/\gamma}}, \tag{6.31$'$}$$

$$P(\psi) = \int_1^3 \phi(\eta)\eta\psi(\eta)\,d\eta, \tag{6.32}$$

and

$$Q(\psi) = \int_1^3 \phi(\eta)\log\psi(\eta)\,d\eta \tag{6.33}$$

and the minimum is taken over all nonincreasing left-continuous step functions $\psi$ defined on $(1,3]$ and taking values in $\{3,\ldots,23\}$. Thus we have reduced the determination of $C'$ to the solution of the variational problem (6.31).

First, we claim that the range $M$ of the function $\psi$ minimizing (6.31) must be an interval of consecutive integers. Suppose to the contrary that for some $h > i > j$ we have $1 < \beta_h = \alpha_i = \beta_i = \alpha_j < 3$. Let us denote this common value by $\delta$, and suppose for now that $\delta \neq 2$. Let us define a new function $\psi'$ by choosing $\varepsilon > 0$, changing $\beta_h = \alpha_i = \delta$ to $\beta'_h = \alpha'_i = \delta - \varepsilon(i-j)$ and $\beta_i = \alpha_j = \delta$ to $\beta'_i = \alpha'_j = \delta + \varepsilon(h-i)$. The effect of this change on $P$ is

$$\begin{aligned} P(\psi') - P(\psi) &= (i-h)\int_{\delta-\varepsilon(i-j)}^{\delta} \phi(\eta)\eta\,d\eta + (i-j)\int_{\delta}^{\delta+\varepsilon(h-i)} \phi(\eta)\eta\,d\eta \\ &= (i-h)\big(\varepsilon(i-j)\phi(\delta)\delta + O(\varepsilon^2)\big) \\ &\quad + (i-j)\big(\varepsilon(h-i)\phi(\delta)\delta + O(\varepsilon^2)\big) \\ &= O(\varepsilon^2) \end{aligned} \tag{6.34}$$

for $\varepsilon > 0$ sufficiently small. The effect on $Q$ is

$$\begin{aligned} Q(\psi') - Q(\psi) &= \left(\log\frac{i}{h}\right)\int_{\delta-\varepsilon(i-j)}^{\delta} \phi(\eta)\,d\eta + \left(\log\frac{i}{j}\right)\int_{\delta}^{\delta+\varepsilon(h-i)} \phi(\eta)\,d\eta \\ &= \left(\log\frac{i}{h}\right)\big(\varepsilon(i-j)\phi(\delta) + O(\varepsilon^2)\big) \\ &\quad + \left(\log\frac{i}{j}\right)\big(\varepsilon(h-i)\phi(\delta) + O(\varepsilon^2)\big) \\ &= \big[(h-j)\log i - (i-j)\log h - (h-i)\log j\big]\phi(\delta)\varepsilon + O(\varepsilon^2). \end{aligned} \tag{6.35}$$

for sufficiently small $\varepsilon > 0$. The quantity in square brackets in (6.35) is strictly positive, by the concavity of the logarithm. Thus the change from $\psi$ to $\psi'$ increases $Q$ to first-order in $\varepsilon$ but increases $P$ only to second-order in $\varepsilon$. It follows that by choosing $\varepsilon > 0$ sufficiently small, we obtain a contradiction to the assumption that $\psi$ minimizes (6.31). In the exceptional case that $\delta = 2$, the same argument works if we introduce a factor of $3^\gamma$ to compensate for the discontinuity of $\phi$: if we set $\beta'_h = \alpha'_i = \delta - \varepsilon(i-j)$ as before, but now set $\beta'_i = \alpha'_j = \delta + \varepsilon(h-i)3^\gamma$, we again obtain cancellation to first-order in $P$ but not in $Q$. Thus we conclude that $M$ is an interval of consecutive integers.

Next we claim that if $m+1$ and $m$ both belong to $M$ and if $\psi$ minimizes $\Gamma(\psi)$ (and therefore also $\log\Gamma(\psi)$), then we must have

$$\beta_{m+1} = \alpha_m = \varrho \log \frac{m+1}{m}, \tag{6.36}$$

where

$$\varrho = \frac{\gamma+1}{\gamma}\frac{P(\psi)}{Q(\psi)}. \tag{6.37}$$

Suppose that $\beta_{m+1} = \alpha_m$, and denote this common value by $\delta$. Suppose for now that $\delta \neq 2$. Let us define a new function $\psi'$ by choosing a small number $\vartheta$ (of either sign) and setting $\beta'_{m+1} = \alpha'_m = \delta + \vartheta$. The effect of this change on $P$ is

$$\begin{aligned} P(\psi') - P(\psi) &= \int_\delta^{\delta+\vartheta} \phi(\eta)\eta\, d\eta \\ &= \vartheta\phi(\delta)\delta + O(\vartheta^2), \end{aligned} \tag{6.38}$$

and the effect on $Q$ is

$$\begin{aligned} Q(\psi') - Q(\psi) &= \left(\log\frac{m+1}{m}\right)\int_\delta^{\delta+\vartheta} \phi(\eta)\, d\eta \\ &= \vartheta\left(\log\frac{m+1}{m}\right)\phi(\delta) + O(\vartheta^2). \end{aligned} \tag{6.39}$$

The effect on $\log\Gamma$ is

$$\log\Gamma(\psi') - \log\Gamma(\psi) = \frac{\vartheta\phi(\delta)\delta}{P(\psi)} - \frac{\gamma+1}{\gamma}\frac{\vartheta\left(\log\frac{m+1}{m}\right)\phi(\delta)}{Q(\psi)} + O(\vartheta^2). \tag{6.40}$$

Thus if (6.36) did not hold, we could choose a small value of $\vartheta$ (with appropriate sign) and make the right-hand side of (6.40) strictly negative. This contradicts the assumption that $\psi$ minimizes $\Gamma(\psi)$ and proves (6.36). In the exceptional case that $\delta = 2$, the same argument works if we interpret $\phi(\delta) = \phi(2)$ (which has not yet been defined) correctly. Specifically, if we wish to choose $\vartheta > 0$, we should set $\phi(2) = 3^{-\gamma}\gamma 2^{\gamma-1}$ (to make $\phi$ right-continuous at 2); and if we wish to choose $\vartheta < 0$, we should set $\phi(2) = \gamma 2^{\gamma-1}$ (to make $\phi$ left-continuous at 2).

Finally, we claim that if $\varrho\log((m+1)/m)$ falls in the open interval $(1,3)$, then $m+1$ and $m$ both belong to $M$. Suppose to the contrary that the largest element of $M$ is $q \leq m$. Let us define a new function $\psi'$ by choosing $\varepsilon > 0$, changing $\alpha_q = 1$ to $\alpha_q = 1+\varepsilon$, and setting $\alpha'_{m+1} = 1$ and $\beta'_{m+1} = 1+\varepsilon$. The change to $P$ is $(m+1-q)\varepsilon\phi(1) + O(\varepsilon^2)$, the change to $Q$ is $(\log((m+1)/q))\varepsilon\phi(1) + O(\varepsilon^2)$, and the change to $\log\Gamma$ is

$$\log\Gamma(\psi') - \log\Gamma(\psi) = \frac{\varepsilon\phi(1)}{P(\psi)} - \frac{\gamma+1}{\gamma}\frac{\varepsilon\left(\log\frac{m+1}{q}\right)\phi(1)}{Q(\psi)} + O(\varepsilon^2), \tag{6.40'}$$

which is strictly negative for $\varepsilon > 0$ sufficiently small because $\varrho \log((m+1)/q) \geq \varrho \log((m+1)/m) > 1$. This contradiction shows that $m+1$ belongs to $M$ when $\varrho \log((m+1)/m) > 1$; a similar argument (setting $\alpha'_m = 3 - \varepsilon$ and $\beta'_m = 3$) shows that $m$ belongs to $M$ when $\varrho \log((m+1)/m) < 3$.

At this point we have reduced the determination of the minimizing function $\psi$ to the determination of the single parameter $\varrho$. Although $\varrho$ is defined by (6.37), we do not yet know the values of $P(\psi)$ and $Q(\psi)$, and thus we seek a more explicit characterization of $\varrho$.

Assume for now that the set of values $\{\varrho \log((m+1)/m) : 3 \leq m, m+1 \leq 23\}$ is disjoint from the set $\{1, 2, 3\}$. Then from (6.36) we can write

$$\psi(\eta) = \left\lceil \frac{1}{\exp\frac{\eta}{\varrho} - 1} \right\rceil . \tag{6.41}$$

Thus if we define $r = \psi(1)$, $s = \psi(2)$, and $t = \psi(3)$, we have

$$r = \left\lceil \frac{1}{\exp\frac{1}{\varrho} - 1} \right\rceil , \tag{6.42}$$

$$s = \left\lceil \frac{1}{\exp\frac{2}{\varrho} - 1} \right\rceil , \tag{6.43}$$

$$t = \left\lceil \frac{1}{\exp\frac{3}{\varrho} - 1} \right\rceil . \tag{6.44}$$

Now we can evaluate the integrals in (6.32) and (6.33) by breaking the range of integration into intervals, over each of which $\psi$ is constant; the results are

$$\begin{aligned} P(\psi) &= r \int_1^{\varrho \log(r/(r-1))} \gamma \eta^\gamma \, d\eta + \sum_{r>q>s} q \int_{\varrho \log((q+1)/q)}^{\varrho \log(q/(q+1))} \gamma \eta^\gamma \, d\eta \\ &\quad + s \int_{\varrho \log((s+1)/s)}^{2} \gamma \eta^\gamma \, d\eta + s \int_2^{\varrho \log(s/(s-1))} 3^{-\gamma} \gamma \eta^\gamma \, d\eta \\ &\quad + \sum_{s>q>t} q \int_{\varrho \log((q+1)/q)}^{\varrho \log(q/(q+1))} 3^{-\gamma} \gamma \eta^\gamma \, d\eta + t \int_{\varrho \log((t+1)/t)}^{3} 3^{-\gamma} \gamma \eta^\gamma \, d\eta \\ &= P_0 + \varrho^{\gamma+1} P_{\gamma+1}, \end{aligned} \tag{6.45}$$

where

$$P_0 = \frac{\gamma+1}{\gamma}(3t + 2s - r) \tag{6.46}$$

and

$$P_{\gamma+1} = \frac{\gamma+1}{\gamma} \left( \sum_{r>q\geq s} \log^{\gamma+1} \frac{q+1}{q} + 3^{-\gamma} \sum_{s>q\geq t} \log^{\gamma+1} \frac{q+1}{q} \right) , \tag{6.47}$$

and

$$(6.48)\, Q(\psi) = (\log r)\int_{1}^{\varrho\log(r/(r-1))} \gamma\eta^{\gamma-1}\,d\eta + \sum_{r>q>s}(\log q)\int_{\varrho\log((q+1)/q)}^{\varrho\log(q/(q+1))}\gamma\eta^{\gamma-1}\,d\eta$$
$$+ (\log s)\int_{\varrho\log((s+1)/s)}^{2}\gamma\eta^{\gamma-1}\,d\eta + (\log s)\int_{2}^{\varrho\log(s/(s-1))}3^{-\gamma}\gamma\eta^{\gamma-1}\,d\eta$$
$$+ \sum_{s>q>t}(\log q)\int_{\varrho\log((q+1)/q)}^{\varrho\log(q/(q+1))}3^{-\gamma}\gamma\eta^{\gamma-1}\,d\eta$$
$$+ (\log t)\int_{\varrho\log((t+1)/t)}^{3}3^{-\gamma}\gamma\eta^{\gamma-1}\,d\eta$$
$$= Q_0 + \varrho^{\gamma}Q_{\gamma},$$

where

$$(6.49) \qquad Q_0 = \log\frac{st}{r}$$

and

$$(6.50) \qquad Q_{\gamma} = \sum_{r>q\geq s}\log^{\gamma+1}\frac{q+1}{q} + 3^{-\gamma}\sum_{s>q\geq t}\log^{\gamma+1}\frac{q+1}{q}.$$

Note that $P_{\gamma+1} = ((\gamma+1)/\gamma)Q_{\gamma}$. Combining (6.37) with (6.45)–(6.50), we conclude that

$$(6.51) \qquad \varrho = \frac{3t+2s-r}{\log\frac{st}{r}}.$$

We now observe that (6.42)–(6.44) and (6.51) have a unique solution, namely,

$$(6.52) \qquad r = 16, \qquad s = 8, \qquad t = 5,$$

and

$$(6.53) \qquad \varrho = \frac{15}{\log\frac{5}{2}}.$$

To verify this, it is convenient to define $\varrho_{p,q} = p/\log((q+1)/q)$, which is the value of $\varrho$ for which $\psi$ makes the step from $q+1$ to $q$ at $p$. Since $r \leq 23$, we must have $\varrho \leq \varrho_{1,23} = 23.4964\ldots$; and if $\varrho = \varrho_{1,23}$, then (6.43) and (6.44) yield $s = 12$ and $t = 8$. Since $t \geq 3$, we must have $\varrho \geq \varrho_{3,2} = 7.3989\ldots$; and if $\varrho = \varrho_{3,2}$, then (6.42) and (6.43) yield $r = 7$ and $s = 4$. Thus we have $r \in \{7,\ldots,23\}$, $s \in \{4,\ldots,12\}$, and $t \in \{3,\ldots,8\}$. The transitions among these possibilities occur when $\varrho$ takes on one of the values $\varrho_{1,7},\ldots,\varrho_{1,22},\varrho_{2,4},\ldots,\varrho_{2,11},\varrho_{3,3},\ldots,\varrho_{3,7}$. These 29 points, when sorted into increasing order, divide the interval $[\varrho_{3,2},\varrho_{1,23}]$ into 30 subintervals, and the values of $r$, $s$, and $t$ given by (6.42)–(6.44) are constant throughout each of these subintervals. Thus there is a unique value of $\varrho$ given by (6.51) for each of these subintervals. In only one case does this value of $\varrho$ fall into the subinterval: throughout the subinterval from $\varrho_{1,15} = 15.4949\ldots$ to $\varrho_{3,5} = 16.4544\ldots$, (6.42)–(6.44) give $r = 16$, $s = 8$, and $t = 5$, whence (6.51) gives $\varrho = 15/\log\frac{5}{2} = 16.3703\ldots$. (There are 30 cases to be considered here; the calculations could be done by hand with patience but were in fact done by a computer.) Thus we have established (6.52) and (6.53), on the assumption that the set of values $\{\varrho \log((m+1)/m) : 3 \leq m, m+1 \leq 23\}$ is disjoint from the set $\{1,2,3\}$ or equivalently that $\varrho$ is not one of the values $\varrho_{1,7}, \ldots, \varrho_{1,23}, \varrho_{2,4}, \ldots, \varrho_{2,11}, \varrho_{3,2}, \ldots, \varrho_{3,7}$. To lift this assumption, we need only verify that (6.37) does not hold if $\varrho$ takes on one of these values. (There are 31 cases to be considered here, and again the calculations were done by a computer.) Substituting (6.52) and (6.53) into (6.45)–(6.50) and (6.26) and simplifying yields $C' = C$ for $C$ given by (6.3).

The outcome of the final search for $r$, $s$, $t$, and $\varrho$ may seem fortuitous or obscure, but it has a simple explanation. The complications of this section are due to the fact that loads, and thus the function $\psi$, can take on only integral values. If we drop this constraint, recovering the problem of §5, we may describe the solution by saying that $\psi(\eta) = 6e/\eta$ is then the minimizing choice of $\psi$. This corresponds to dropping the ceiling brackets, replacing $\exp(\eta/\varrho)$ by the first two terms $1 + (\eta/\varrho)$ of its power series expansion and taking $\varrho = 6e = 16.3096\ldots$ in (6.41); and the values of $r$, $s$, and $t$ corresponding to this value of $\varrho$ according to (6.42)–(6.44) are $r = 16$, $s = 8$, and $t = 5$.

We can also interpret the individual terms in the denominator of (1.6) in terms of "promotions." Specifically, we can associate the term $\log^{\gamma+1}((q+1)/q)$ (for $8 \leq q \leq 15$) or $3^{-\gamma} \log^{\gamma+1}((q+1)/q)$ (for $5 \leq q \leq 7$) with the promotion of a load from $q$ to $q+1$; and we can associate the term $15^{-\gamma} \log \frac{5}{2}$ with the promotion of leaf with load 16 to a parent of leaves with loads 8 and 5. The expression for $C$ is thus analogous to those for $C_2$ and $C_4$, though the justification is much more elaborate.

Finally, we observe that for any fixed $\varepsilon > 0$, (1.3) is fulfilled for infinitely many $n$ in a geometric progression. To see this, observe that we may replace the optimal values of $\mu_m$ for $5 \leq m \leq 16$ by rational numbers $p_m/q$ without increasing the value of $\Gamma(\psi)$ to more than $C + \varepsilon$. Then if we consider a sequence $T_i$ of trees having $qi$ leaves and form from these a sequence $D_i$ of derivations in which there are $p_m i$ leaves with load $m$, the resulting derivations will have capacity $(\Pi_{5 \leq m \leq 16}\, m^{p_m})^i$ and cost satisfying (1.3). It should be noted that this observation does not contradict the results of §4, which dealt with the behavior of (1.1) restricted to geometric progressions: in the observation the capacity $n$ is confined to a geometric progression, but the divisor $m$ of $n$ in (1.1) ranges over all proper divisors; in §4, both $n$ and $m$ were confined to the geometric progression.

## REFERENCES

[BP] L. A. Bassalygo and M. S. Pinsker, *Complexity of an optimum nonblocking network without reconnections*, Problems Inform. Transmission, 9 (1974), pp. 64–66.

[Ca] D. G. Cantor, *On non-blocking switching networks*, Networks, 1 (1971), pp. 367–377.

[Cl] C. Clos, *A study of non-blocking switching networks*, Bell System Tech. J., 32 (1953), pp. 406–424.

[FK] M. L. Fredman and D. E. Knuth, *Recurrence relations based on minimization*, J. Math. Anal. Appl., 48 (1974), pp. 534–559.

[P1] N. Pippenger, *Telephone switching networks*, Proc. Sympos. Appl. Math. 26, American Mathematical Society, Providence, RI, 1982, pp. 101–133.

[P2] ———, *An elementary approach to some analytic asymptotics*, SIAM J. Math. Anal., 24 (1993), pp. 1361–1377.

[S] C. E. Shannon, *Memory requirements in a telephone exchange*, Bell System Tech. J., 29 (1950), pp. 343–349.

SIAM J. DISC. MATH.
Vol. 8, No. 3, pp. 347–358, August 1995

# AN INEQUALITY FOR PROBABILITY MOMENTS WITH APPLICATIONS *

EARL R. BARNES[†]

**Abstract.** We give an inequality relating the range of a discrete probability distribution to certain determinants formed from moments of the distribution. We show applications of this inequality to problems in geometry, statistics, linear algebra, and combinatorial optimization.

**Key words.** probabilities, inequalities, eigenvalues, moments

**AMS subject classifications.** 60E15, 65F15, 90C27

## 1. Introduction.

Let $x_1 < x_2 < \cdots < x_n$ be $n$ distinct numbers and let $p_1, \ldots, p_n$ be $n$ positive numbers satisfying $\sum_{j=1}^{n} p_j = 1$. Define

$$\mu_r = \sum_{j=1}^{n} p_j x_j^r, \qquad r = 0, 1, \ldots. \tag{1.1}$$

Following the convention in probability theory, we refer to $\mu_r$ as the $r$th moment of the numbers $x_1, \ldots, x_n$ with respect to the probabilities $p_1, \ldots, p_n$. In our applications, the $x$'s and $p$'s are unknown, but we know a few of the $\mu$'s. Our problem is to determine something about the $x$'s from our knowledge of the $\mu$'s. In particular, we would like to obtain bounds on $x_1$ and $x_n$, as well as the spread $x_n - x_1$, in terms of the $\mu$'s. Perhaps the best-known result along these lines is Chebychev's inequality. Let $X$ be a random variable that takes on the values $x_1, \ldots, x_n$ with probabilities $p_1, \ldots, p_n$, respectively. Chebychev's inequality states that for $t > 1$,

$$\text{prob}\{\mu_1 - t\sigma \le X \le \mu_1 + t\sigma\} \ge 1 - \frac{1}{t^2},$$

where $\sigma = \sqrt{\mu_2 - \mu_1^2}$. By letting $t$ approach 1 we conclude that at least one of the points $x_1, \ldots, x_n$ lies in the interval $\mu_1 - \sigma \le x \le \mu_1 + \sigma$.

As a sort of complement to this result, several authors, in particular Wolkowicz and Styan [14] and Brauer and Mewborn [4], have shown that not all of the $x$'s can satisfy the condition $\mu_1 - \sigma < x < \mu_1 + \sigma$. In fact, they have shown that

$$x_n - x_1 \ge 2\sigma. \tag{1.2}$$

See also [1], [2], [5], and [9].

It turns out that sharper bounds on the spread $x_n - x_1$ can be obtained when higher moments of the $x$'s are available. The purpose of this paper is to explain how this can be done. We will also obtain bounds for the extreme values $x_1$ and $x_n$.

## 2. Bounds.

In [2] we derived the inequality

$$(x_n - \mu_1)(\mu_1 - x_1) \ge \sigma^2, \tag{2.1}$$

* Received by the editors November 11, 1991; accepted for publication (in revised form) June 22, 1994. This research was supported by National Science Foundation grant DDM-9014823.

[†] School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0205.

which is a stronger statement than (1.2); for by the inequality between the geometric and arithmetic means of positive numbers, we have, from (2.1),

$$\sigma \leq \sqrt{x_n - \mu_1}\sqrt{\mu_1 - x_1} \leq \frac{(x_n - \mu_1) + (\mu_1 - x_1)}{2} = \frac{x_n - x_1}{2}.$$

Thus (2.1) implies (1.2). The main contribution of this paper is a strengthening of inequality (2.1) when higher moments of the $x$'s are available. This modification of (2.1) is stated in the following theorem.

THEOREM 2.1. *For $k \geq 2$, let $\mu_0, \ldots, \mu_{2k}$ be moments of the numbers $x_1 < x_2 < \cdots < x_n$ with respect to certain probabilities $p_1, \ldots, p_n$. Let $M, M_0$, and $M_1$ denote the matrices*

$$M = \begin{pmatrix} \mu_0 & \mu_1 & \cdots & \mu_k \\ \mu_1 & \mu_2 & \cdots & \mu_{k+1} \\ \vdots & & & \\ \mu_k & \mu_{k+1} & \cdots & \mu_{2k} \end{pmatrix}, \qquad M_0 = \begin{pmatrix} \mu_0 & \cdots & \mu_{k-1} \\ \mu_1 & \cdots & \mu_k \\ \vdots & & \\ \mu_{k-1} & \cdots & \mu_{2k-2} \end{pmatrix},$$

$$M_1 = \begin{pmatrix} \mu_1 & \cdots & \mu_k \\ \mu_2 & \cdots & \mu_{k+1} \\ \vdots & & \\ \mu_k & \cdots & \mu_{2k-1} \end{pmatrix},$$

*and assume that $M_0$ is nonsingular. Define moments $s_0, \ldots, s_{2k-2}$ by*

$$s_j = \frac{1}{k}\operatorname{Tr}(M_0^{-1}M_1)^j, \qquad j = 0, \ldots, 2k-2.$$

*Let $N$ and $N_0$ denote the moment matrices*

$$N = \begin{pmatrix} s_0 & s_1 & \cdots & s_{k-1} \\ s_1 & s_2 & \cdots & s_k \\ \vdots & & & \\ s_{k-1} & s_k & \cdots & s_{2k-2} \end{pmatrix},$$

$$N_0 = \begin{pmatrix} s_0 & s_1 & \cdots & s_{k-2} \\ s_1 & s_2 & \cdots & s_{k-3} \\ \vdots & & & \\ s_{k-2} & s_{k-1} & \cdots & s_{2k-4} \end{pmatrix}.$$

*Then $N$ is nonsingular, and*

$$(2.2) \qquad (x_n - \mu_1)(\mu_1 - x_1) \geq \sigma^2 + \frac{1}{k}\frac{\det M}{\det M_0} \cdot \frac{\det N_0}{\det N}.$$

Before we prove this theorem, we review some known results about moment matrices. These results are given in Chapters 2 and 4 of [8]. We summarize them here to make our paper self-contained.

Our assumption that $M_0$ is nonsingular implies that $k \leq n$. To see this, observe that for $k > n$, say $k = n+1$, we can write $M_0$ as the product of singular matrices as follows.

$$M_0 = \begin{pmatrix} \mu_0 & \cdots & \mu_{k-1} \\ \mu_1 & \cdots & \mu_k \\ \vdots & & \\ \mu_{k-1} & \cdots & \mu_{2k-2} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & \dots & 1 & 0 \\ x_1 & x_2 & \dots & x_n & 0 \\ \vdots & & & & \\ x_1^{k-1} & x_2^{k-1} & \cdots & x_n^{k-1} & 0 \end{pmatrix} \begin{pmatrix} p_1 & p_1x_1 & \dots & p_1x_1^{k-1} \\ p_2 & p_2x_2 & \dots & p_2x_2^{k-1} \\ \vdots & \vdots & & \vdots \\ p_n & p_nx_n & & p_nx_n^{k-1} \\ 0 & 0 & \dots & 0 \end{pmatrix}.$$

Thus if $M_0$ is to be nonsingular, we must have $k \leq n$. In general, we assume that $k$ is much smaller than $n$. The numerical example given in §5 is typical of what we have in mind.

Let $\Omega$ denote the set of measures $\sigma$ defined on $[x_1, x_n]$ and satisfying

$$\int_{x_1}^{x_n} t^r \, d\sigma(t) = \mu_r, \qquad r = 0, \dots, 2k-1.$$

$\Omega$ is not empty, since $\Sigma_{j=1}^n p_j x_j^r = \mu_r, r = 0, \dots, 2k$. For a fixed $\sigma \in \Omega$, let $\hat{\mu}_{2k} = \int_{x_1}^{x_n} t^{2k} d\sigma(t)$. Let $v(t)$ denote the $(k+1)$-dimensional vector with components $1, t, \dots, t^k$ and $\mu$ denote the column vector with components $\mu_k, \dots, \mu_{2k-1}$. We then have

$$\int_{x_1}^{x_n} v(t)v(t)^T d\sigma(t) = \begin{pmatrix} M_0 & \mu \\ \mu^T & \hat{\mu}_{2k} \end{pmatrix}.$$

Clearly this matrix is positive semidefinite since $v(t)v(t)^T \geq 0$ for each $t \in [x_1, x_n]$. This means that

$$\det \begin{pmatrix} M_0 & \mu \\ \mu^T & \hat{\mu}_{2k} \end{pmatrix} = (\hat{\mu}_{2k} - \mu^T M_0^{-1} \mu) \det M_0 \geq 0.$$

By assumption $\det M_0 > 0$ so $\hat{\mu}_{sk} - \mu^T M_0^{-1} \mu \geq 0$. This proves that

$$\int_{x_1}^{x_n} t^{2k} \, d\sigma(t) \geq \mu^T M_0^{-1} \mu$$

for any $\sigma \in \Omega$. In fact, it can be shown that the problem

$$\text{(2.3)} \qquad \min_\sigma \int_{x_1}^{x_n} t^{2k} \, d\sigma(t) \quad \text{subject to} \int_{x_1}^{x_n} t^r \, d\sigma(t) = \mu_r, \qquad r = 0, \dots, 2k-1$$

has minimum value $\mu_{2k}^* = \mu^T M_0^{-1} \mu$. Moreover, the measure that realizes this minimum concentrates mass at precisely $k$ points $\xi_1 < \cdots < \xi_k$ in $[x_1, x_n]$. This means that there are positive numbers $\sigma_1, \dots, \sigma_k$ satisfying

$$\text{(2.4)} \qquad \sigma_1 \xi_1^r + \sigma_2 \xi_2^r + \cdots + \sigma_k \xi_k^r = \mu_r, \qquad r = 0, 1, \dots, 2k-1.$$

This is the main result we need for the proof of Theorem 2.1.

*Proof of Theorem* 2.1. Let $v_j$ and $E_j$ denote the arrays

$$v_j = \begin{pmatrix} 1 \\ x_j \\ \vdots \\ x_j^{k-1} \end{pmatrix}, \quad E_j = v_j v_j^T, \quad j = 1, \dots, n,$$

and define

$$M_2 = \begin{pmatrix} \mu_2 & \cdots & \mu_{k+1} \\ \mu_3 & \cdots & \mu_{k+2} \\ \vdots & & \\ \mu_{k+1} & \cdots & \mu_{2k} \end{pmatrix}.$$

We then have, by definition,

$$M_0 = \sum_{j=1}^n p_j E_j, \quad M_1 = \sum_{j=1}^n p_j x_j E_j, \quad M_2 = \sum_{j=1}^n p_j x_j^2 E_j. \tag{2.5}$$

Clearly each $E_j$ is positive semidefinite, and since $x_1 \le x_j \le x_n$, for each $j$, the matrix $\sum_{j=1}^n p_j(x_n - x_j)(x_j - x_1)E_j$ is positive semidefinite. We write this as

$$\sum_{j=1}^n p_j\{(x_1 + x_n)x_j - x_1 x_n - x_j^2\}E_j = (x_1 + x_n)M_1 - x_1 x_n M_0 - M_2 \ge 0. \tag{2.6}$$

The matrices $M_0$ and $M_1$ involve only the moments $\mu_0, \dots, \mu_{2k-1}$, for which representation (2.4) is valid. We can therefore write

$$M_0 = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \xi_1 & \xi_2 & \cdots & \xi_k \\ \vdots & & & \\ \xi_1^{k-1} & \xi_2^{k-1} & \cdots & \xi_k^{k-1} \end{pmatrix} \begin{pmatrix} \sigma_1 & \sigma_1\xi_1 & \cdots & \sigma_1\xi_1^{k-1} \\ \sigma_2 & \sigma_2\xi_2 & \cdots & \sigma_2\xi_2^{k-1} \\ \vdots & & & \\ \sigma_k & \sigma_k\xi_2 & \cdots & \sigma_k\xi_k^{k-1} \end{pmatrix} = VSV^T,$$

where $S =$ diag$(\sigma_1, \dots, \sigma_k)$ and $V$ is the Vandermonde matrix

$$V = \begin{pmatrix} 1 & \cdots & 1 \\ \xi_1 & \cdots & \xi_k \\ \vdots & & \\ \xi_1^{k-1} & \cdots & \xi_k^{k-1} \end{pmatrix}.$$

$V$ is nonsingular, since the $\xi$'s are distinct.

We can also write $M_0 = (VS^{1/2})(VS^{1/2})^T$ and $M_1 = VS\Xi V^T$, where $\Xi =$ diag$(\xi_1, \dots, \xi_k)$. If we multiply inequality (2.6) on the left by $(VS^{1/2})^{-1}$ and on the right by $(VS^{1/2})^{-T}$, we obtain

$$(x_1 + x_n)\Xi - x_1 x_n I - S^{-1/2}V^{-1}M_2 V^{-T}S^{-1/2} \ge 0. \tag{2.7}$$

If we now observe that

$$\Xi^2 = (S^{-1/2}V^{-1}M_1 V^{-T}S^{-1/2})^2 = S^{-1/2}V^{-1}M_1 M_0^{-1} M_1 V^{-T}S^{-1/2},$$

we deduce from (2.7) that

$$S^{1/2}\{(x_1+x_n)\Xi - x_1x_nI - \Xi^2\}S^{1/2} \geq V^{-1}(M_2 - M_1M_0^{-1}M_1)V^{-T}. \tag{2.8}$$

Let $u_j$ denote the $j$th column of $M_1$ and $e_j$ denote the $j$th unit coordinate vector in $R^k$. Since $u_j$ is the $(j+1)$st column in $M_0$, we have $M_0^{-1}u_j = e_{j+1}, j = 1, \dots, k-1$. It follows that $u_i^TM_0^{-1}u_j = \mu_{i+j}$ for $2 \leq i+j < 2k$. This implies that

$$M_2 - M_1M_0^{-1}M_1 = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & \chi \end{pmatrix} = (e_ke_k^T)\chi,$$

where $\chi = \mu_{2k} - u_k^TM_0^{-1}u_k = (\det M)/(\det M_0)$. If we substitute this into (2.8) and take the trace on each side of this inequality, we obtain

$$\sum_{j=1}^{k} \sigma_j\{(x_1+x_n)\xi_j - x_1x_n - \xi_j^2\} \geq e_k^T(VV^T)^{-1}e_k\chi.$$

By using representation (2.4), we can write this inequality as

$$(x_1+x_n)\mu_1 - x_1x_n - \mu_1^2 \geq \mu_2 - \mu_1^2 + (e_k^T(VV^T)^{-1}e_k)\frac{\det M}{\det M_0}. \tag{2.9}$$

To complete the derivation of (2.2), we must find an expression for the matrix $VV^T$ in terms of $M_0$ and $M_1$.

In Chapter 4 of [8], it is shown that the numbers $\xi_1, \dots, \xi_k$ in representation (2.4) are the roots of the polynomial equation

$$\det\begin{pmatrix} \mu_0 & \mu_1 & \dots & \mu_{k-1} & 1 \\ \mu_1 & \mu_2 & \dots & \mu_k & t \\ \vdots & \vdots & & \vdots & \vdots \\ \mu_k & \mu_{k+1} & \dots & \mu_{2k-1} & t^k \end{pmatrix} = 0. \tag{2.10}$$

There seems to be no simple proof of (2.4). However, given that (2.4) is true, it is easy to see that the $\xi_j$'s are the roots of equation (2.10). This follows from the matrix factorization

$$\begin{pmatrix} \mu_0 & \mu_1 & \dots & \mu_{k-1} & 1 \\ \mu_1 & \mu_2 & \dots & \mu_k & t \\ \vdots & \vdots & & & \\ \mu_k & \mu_{k+1} & \dots & \mu_{2k-1} & t^k \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \xi_1 & \xi_2 & & t \\ \vdots & \vdots & & \vdots \\ \xi_1^k & \xi_2^k & & t^k \end{pmatrix} \begin{pmatrix} \sigma_1 & \sigma_1\xi_1 & \dots & \sigma_1\xi_1^{k-1} & 0 \\ \sigma_2 & \sigma_2\xi_2 & \dots & \sigma_2\xi_2^{k-1} & 0 \\ \vdots & & & & \\ \sigma_k & \sigma_k\xi_k & \dots & \sigma_k\xi_k^{k-1} & 0 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}.$$

Since the numbers $\sigma_1, \dots, \sigma_k$ are positive and the $\xi_j$'s are distinct, (2.10) holds if and only if $t$ is one of the $\xi_j$'s.

If we multiply each row in determinant (2.10) by $t$ and subtract it from the next row, we obtain the result

$$\det\begin{pmatrix} \mu_0 & \mu_1 & \dots & \mu_{k-1} & 1 \\ \mu_1 & \mu_2 & \dots & \mu_k & t \\ \vdots & \vdots & & \vdots & \vdots \\ \mu_k & \mu_{k+1} & \dots & \mu_{2k-1} & t^k \end{pmatrix} = \det(M_1 - tM_0).$$

It follows that the numbers $\xi_1, \dots, \xi_k$ are the eigenvalues of the matrix $M_0^{-1/2} M_1 M_0^{-1/2}$. We therefore have

$$\xi_1 + \cdots + \xi_k = \operatorname{Tr}(M_0^{-1/2} M_1 M_0^{-1/2}) = \operatorname{Tr}(M_0^{-1} M_1)$$

and, in general,

$$\xi_1^r + \cdots + \xi_k^r = \operatorname{Tr}(M_0^{-1/2} M_1 M_0^{-1/2})^r = \operatorname{Tr}(M_0^{-1} M_1)^r, \qquad r = 0, 1, \dots.$$

This shows that $VV^T = kN$, and if we substitute this into (2.9), we obtain

$$(2.11) \qquad (x_n - \mu_1)(\mu_1 - x_1) \geq \sigma^2 + \frac{1}{k}(e_k^T N^{-1} e_k) \frac{\det M}{\det M_0}.$$

By Cramer's rule, we have

$$e_k^T N^{-1} e_k = \frac{\det N_0}{\det N}.$$

Thus (2.11) agrees with (2.2) and the proof is complete.

Note that it is not necessary to compute the inverse $M_0^{-1}$ in order to determine the moments $s_0, s_1, \dots$. This follows from the observation that

$$M_0^{-1} M_1 = \begin{pmatrix} 0 & 0 & & \\ 1 & 0 & & \\ 0 & 1 & & \\ 0 & 0 & \dots & M_0^{-1} u_k \\ \vdots & \vdots & & \\ 0 & 0 & & \end{pmatrix}.$$

The column $M_0^{-1} u_k$ can be computed by solving the equation $M_0 v = u_k$ for $v = M_0^{-1} u_k$. The following obvious consequence of Theorem 2.1 sharpens inequality (1.2).

COROLLARY. *The numbers $x_1$ and $x_n$ in Theorem 2.1 satisfy*

$$(2.12) \qquad x_n - x_1 \geq 2 \left\{ \sigma^2 + \frac{1}{k} \frac{\det M}{\det M_0} \frac{\det N_0}{\det N} \right\}^{1/2}.$$

We close this section with some results on bounds for $x_1$ and $x_n$. It is well known that the numbers $\xi_1$ and $\xi_k$ in (2.4) satisfy

$$(2.13) \qquad x_1 \leq \xi_1 \quad \text{and} \quad x_n \geq \xi_k.$$

This result is an immediate consequence of (2.5); if $\xi$ is a solution of the equation $\det(M_1 - tM_0) = 0$, there is a nonzero vector $y \in R^k$ such that $\xi = y^T M_1 y / y^T M_0 y$. It follows from (2.5) that

$$\xi = \frac{\sum_{j=1} p_j x_j (v_j^T y)^2}{\sum_{i=1}^n p_i (v_i^T y)^2} = \sum_{j=1}^n \left\{ \frac{p_j (v_j^T y)^2}{\sum_{i=1}^n p_i (v_i^T y)^2} \right\} x_j.$$

This clearly shows that $x_1 \leq \xi \leq x_n$, and thus (2.13) holds.

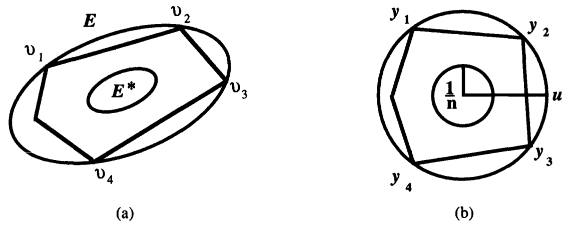


FIG. 1.

**3. The minimum covering ellipsoid for a polytope.** Let $P$ be a polytope in $R^n$ and $E$ denote the ellipsoid of least volume containing $P$. Let $c$ denote the center of $E$. Then there exists a positive definite matrix $Q$ such that

$$E = \{x | (x-c)^T Q (x-c) \le 1\}. \tag{3.1}$$

The volume of $E$ is proportional to $\det Q^{-1/2}$. Thus, if $v_1, \dots, v_N$ are the vertices of $P$, then $Q$ and $c$ solve the minimization problem

$$\text{minimize } \det M^{-1/2} \quad \text{subject to } (v_j - \gamma)^T M (v_j - \gamma) \le 1, \qquad j = 1, \dots, N, \tag{3.2}$$

where the minimization is taken over all positive definite matrices $M \in R^{n\times n}$ and over all vectors $\gamma \in R^n$.

In [7], John gives a characterization of the solution of (3.3). He shows that there exist nonnegative numbers $p_1, \dots, p_N$ satisfying $\sum_{j=1}^N p_j = 1$ such that

$$c = \sum_{j=1}^N p_j v_j, \qquad \frac{1}{n} Q^{-1} = \sum_{j=1}^N p_j (v_j - c)(v_j - c)^T, \tag{3.3}$$

and

$$p_j\{(v_j - c)^T Q (v_j - c) - 1\} = 0, \qquad j = 1, \dots, N.$$

John uses this characterization to prove the following theorem.

THEOREM 3.1. *Let the ellipsoid* (2.1) *be the ellipsoid of least volume containing a polytope $P \subset R^n$. Then the ellipsoid*

$$E^* = \left[ x | (x-c)^T Q (x-c) \le \left(\frac{1}{n}\right)^2 \right], \tag{3.4}$$

*obtained by shrinking $E$ by a factor $n$ about its center, is contained in $P$.*

*Proof.* Let $v_1, \dots, v_N$ denote the vertices of $P$. Some of these vertices lie on the boundary of $E$; see Fig. 1(a). For simplicity, assume these vertices are $v_1, \dots, v_k$. Since $p_j = 0$ if $(v_j - c)^T Q (v_j - c) < 1$, in (3.3) we have that $p_j = 0$ for $j = k+1, \dots, N$.

The linear transformation $y = Q^{1/2}(x - c)$ maps $E$ to the unit ball in $R^n$. Let $y_j = Q^{1/2}(v_j - c), j = 1, \dots, k$. To complete the proof of the theorem, it suffices to show that the convex hull of the points $\{y_1, \dots, y_k\}$ contains a ball of radius $1/n$ centered at 0. Under the map $x = Q^{-1}y + c$, this ball goes into the ellipsoid (3.4), and this ellipsoid is contained in the convex hull of the points $v^1, \dots, v^k$ and hence in $P$.

Let $u$ be a unit outward normal to a facet of the convex hull $\text{conv}\{y_1, \dots, y_k\}$; see Fig. 1(b). We show that the distance from 0 to this facet is at least $1/n$. This in turn shows that we can construct a ball of radius $1/n$, centered at the origin, inside $\text{conv}\{y_1, \dots, y_k\}$. The projection of $y_j$ on the direction $u$ is given by $u^T y_j$. We must show that $\max_j u^T y_j \ge 1/n$.

If we multiply the first equation in (3.3) by $Q^{1/2}$, we obtain

$$\sum_{j=1}^{k} p_j y_j = 0.$$

If we multiply the second equation on the right and left by $Q^{1/2}$, we obtain

$$\frac{1}{n} I = \sum p_j y_j y_j^T,$$

where $I$ denotes the $n \times n$ identity matrix. These last two equations imply that

$$\sum_{j=1}^{k} p_j (u^T y_j) = 0$$

and

$$\sum_{j=1}^{k} p_j u^T y_i y_j^T u = \sum_{j=1}^{k} p_j (u^T y_j)^2 = \frac{1}{n}.$$

Here we are able to compute the first two moments of the numbers $u^T y_1, \ldots, u^T y_k$ with respect to an unknown probability distribution. This is precisely the type of situation that Theorem 2.1 deals with.

Let $x_k = \max u^T y_j$ and $x_1 = \min u^T y_j, j = 1, \ldots, k$. By inequality (2.1), we have

$$x_k(-x_1) \geq \frac{1}{n},$$

and since $|u^T y_j| \leq \|u\| \|y_j\| = 1, j = 1, \ldots, k$, we have $-x_1 = |x_1| \leq 1$. Thus $x_k \geq 1/n$, and thus our proof is complete.

**4. Linear regression theory.** In a linear regression problem, an output variable $\eta$ is related to several input variables $\xi_1, \ldots, \xi_n$ by an equation of the form $\eta = \beta_0 + \beta_1 \xi_1 + \cdots + \beta_n \xi_n$, where the $\beta$'s are not known precisely. Estimates of the $\beta$'s are obtained by measuring several values $y_1, \ldots, y_m, m > n$, of the output variable corresponding to values $x_j = (x_{1j}, \ldots, x_{nj}), j = 1, \ldots, m$, of the input variables. We assume that the measurements are given by

$$y_j = \beta_0 + \sum_{i=1}^{n} \beta_i x_{ij} + \varepsilon_j, \qquad j = 1, \ldots, m,$$

where the $\varepsilon_j$'s are independent and identically distributed random variables with means 0 and variances $\sigma^2$.

Let $X$ denote the $m \times (n+1)$ matrix whose $j$th row is $(1, x_{1j}, x_{2j}, \ldots, x_{nj}), j = 1, \ldots, m$. The least squares estimate of $\beta = (\beta_0, \ldots, \beta_n)^T$ is given by

$$\hat{\beta} = (X^T X)^{-1} X^T y,$$

where $y = (y_1, \ldots, y_m)^T$ is the vector of measurements. Since $y$ is a random vector, $\hat{\beta}$ is a random vector. A new set of experiments yields a new estimate $\hat{\beta}$. The covariance matrix of $\hat{\beta}$ is given by

$$E(\hat{\beta} - \beta)(\hat{\beta} - \beta)^T = \sigma^2 (X^T X)^{-1}.$$

In particular, the variance of a component $\hat{\beta}_j$ of $\hat{\beta}$ is given by $\sigma^2 C_{jj}$, where $C_{jj}$ is the $(j+1)$th diagonal term of $(X^TX)^{-1}, j = 0, 1, \ldots, n$. The first inequality in (1.2) shows that the range of values for $\hat{\beta}_j$ is at least $2\sigma\sqrt{C_{jj}}, j = 0, 1, \ldots, n$. One would prefer these ranges to be small. In fact, in some studies on the design of experiments, one tries to select the measurements $x_j$ to

$$\text{minimize } \mathrm{Trace}(X^TX)^{-1} = \sum_{j=1}^{n+1} C_{jj}.$$

See, for example, [6].

*Remark.* Theorem 2.1 was stated for discrete probability distributions. The present application is to a distribution that is possibly continuous. It is easy to show that the theorem remains valid when the moments $\mu_r$ are taken from a continuous distribution.

**5. The spread of eigenvalues.** Let $\lambda_1 \le \lambda_2 \le \cdots \lambda_n$ denote the eigenvalues of a real symmetric matrix $A = (a_{ij})$. Several authors [2]–[4], [10]–[14] have been interested in estimating the spread $\lambda_n - \lambda_1$ of these eigenvalues. Theorem 2.1 can be used to obtain lower bounds on this spread as follows.

Let $u_1, \ldots, u_n$ denote a set of orthonormal eigenvectors of $A$ corresponding to the eigenvalues $\lambda_1, \ldots, \lambda_n$. Then

$$A^r = \lambda_1^r u_1 u_1^T + \lambda_2^r u_2 u_2^T + \cdots + \lambda_n^r u_n u_n^T, \qquad r = 0, 1, \ldots. \tag{5.1}$$

Let $a_{ij}^{(r)}$ denote the $ij$th element of $A^r$ and $u_{ij}$ denote the $i$th element of $u_j, i, j = 1, \ldots, n$. From (5.1), we see that

$$a_{ii}^{(r)} = \sum_{j=1}^{n} u_{ij}^2 \lambda_j^r, i = 1, \ldots, n.$$

Since the vectors $u_j$ are orthonormal, we have $\sum_{j=1}^n u_{ij}^2 = 1, i = 1, \ldots, n$. The diagonal terms of $A^r$ are thus moments of the eigenvalues of $A$.

For each diagonal position $(i, i)$, we can compute the moment matrices

$$M = \begin{pmatrix} 1 & a_{ii} & \cdots & a_{ii}^{(k)} \\ a_{ii} & a_{ii}^{(2)} & \cdots & a_{ii}^{(k+1)} \\ \vdots & & & \\ a_{ii}^{(k)} & a_{ii}^{(k+1)} & \cdots & a_{ii}^{(2k)} \end{pmatrix}, \qquad M_0 = \begin{pmatrix} 1 & \cdots & a_{ii}^{(k-1)} \\ a_{ii} & \cdots & a_{ii}^{(k)} \\ \vdots & & \\ a_{ii}^{(k-1)} & \cdots & a_{ii}^{(2k-2)} \end{pmatrix},$$

$$M_1 = \begin{pmatrix} a_{ii} & \cdots & a_{ii}^{(k)} \\ a_{ii}^2 & \cdots & a_{ii}^{(k+1)} \\ \vdots & & \\ a_{ii}^{(k)} & \cdots & a_{ii}^{(2k-1)} \end{pmatrix}.$$

and apply (2.12) to obtain a lower bound on the spread $\lambda_n - \lambda_1$ of the eigenvalues of $A$. We can also use (2.13) to obtain bounds on $\lambda_1$ and $\lambda_n$.

*Example.* Consider the $n \times n (n \geq 4)$ circulant matrix

$$A = \begin{pmatrix} 2 & 1 & 0 & \dots & 0 & 1 \\ 1 & 2 & 1 & & & 0 \\ 0 & 1 & 2 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & & 1 \\ 1 & \dots & 0 & 1 & & 2 \end{pmatrix}.$$

It is easy to show that $A^r$ is a circulant matrix with diagonal entries $\binom{2r}{r}, r = 1, 2, 3, 4$. We therefore have moments

$$\mu_r = \binom{2r}{r}, \qquad r = 1, 2, 3, 4$$

for the eigenvalues of $A$. For the purpose of demonstrating (2.12), we set $k = 2$. We then have

$$\mu_1 = 2, \quad \mu_2 = 6, \quad \mu_3 = 20, \quad \mu_4 = 70,$$

so

$$M = \begin{pmatrix} 1 & 2 & 6 \\ 2 & 6 & 20 \\ 6 & 20 & 70 \end{pmatrix}, \quad M_0 = \begin{pmatrix} 1 & 2 \\ 2 & 6 \end{pmatrix}, \quad M_1 = \begin{pmatrix} 2 & 6 \\ 6 & 20 \end{pmatrix},$$

$$M_0^{-1} M_1 = \begin{pmatrix} 0 & -2 \\ 1 & 4 \end{pmatrix}, \quad \text{and} \quad (M_0^{-1} M_1)^2 = \begin{pmatrix} -2 & -8 \\ 4 & 14 \end{pmatrix}.$$

It follows that $s_1 = 2, s_2 = 6$, and

$$N = \begin{pmatrix} 1 & 2 \\ 2 & 6 \end{pmatrix}, \qquad N_0 = 1.$$

Therefore

$$\sigma^2 + \frac{1}{2} \frac{\det M}{\det M_0} \frac{\det N_0}{\det N} = 2 + \frac{1}{2} \cdot \frac{4}{2} \cdot \frac{1}{2} = 2.5.$$

It follows from (2.12) that

$$\lambda_n - \lambda_1 \geq 2\sqrt{2.5} = 3.1622 \tag{5.2}$$

for any $n$.

Since $A$ is a circulant matrix, we know that its eigenvalues are given by

$$\lambda_j = 2 + \omega_j + \omega_j^{n-1}, \qquad j = 1, \dots, n,$$

where $\omega_1, \dots, \omega_n$ are the $n$th roots of unity. That is, $(\omega_j)^n = 1, j = 1, \dots, n$. When $n$ is even, we have the eigenvalues $\lambda_1 = 0$ and $\lambda_n = 4$, corresponding to $\omega_j = -1$ and $+1$, respectively. Thus $\lambda_n - \lambda_1 = 4$ for $n$ even. When $n$ is odd, we have $\lambda_n = 4$ and $\lambda_1 > 0$. (5.2) thus gives a better estimate of the spread for $n$ odd. For example, for $n = 5$, we have $\lambda_1 = 2 + 2\cos\frac{4\pi}{5} = .3819$ and $\lambda_n - \lambda_1 = 3.618$.

(2.12) tells us something about the spread of the eigenvalues of $A$ but gives no information about the precise locations of $\lambda_1$ and $\lambda_n$. For this information, we invoke (2.13). The eigenvalues of the matrix

$$M_0^{-1} M_1 = \begin{pmatrix} 0 & -2 \\ 1 & 4 \end{pmatrix}$$

satisfy $\lambda^2 - 4\lambda + 2 = 0$. We therefore have

$$\xi_1 = 2 - \sqrt{2} \quad \text{and} \quad \xi_2 = 2 + \sqrt{2}.$$

Thus, by (2.13), we have $\lambda_1 \leq 2 - \sqrt{2}$ and $\lambda_n \geq 2 + \sqrt{2}$.

It can be shown that the inequalities (2.13) become sharper as $k$ increases. In fact, our discussion of the polynomials (2.10) shows that for $k = n$, the moments $\mu_0, \mu_1, \ldots, \mu_{2k-1}$ determine the numbers $x_1, \ldots, x_n$ exactly. Thus, in the present example, when $k = n$ we have $\xi_j = \lambda_j, j = 1, \ldots, n$. On the other hand, the right-hand side of (2.12) may not increase as $k$ increases. This means that, in some cases, the difference $\xi_k - \xi_1$ may give a better estimate of the spread than (2.12).

**6. Combinatorial optimization problems.** Consider a traveling salesman problem on $n$ cities. For simplicity, we denote the cities by $1, 2, \ldots, n$. Let $a_{ij}$ denote the distance between cities $i$ and $j$ if $i \neq j$ and set $a_{ii} = 0, i = 1, \ldots, n$. The traveling salesman problem requires that we find a permutation $i_1, i_2, \ldots, i_n$ of the numbers $1, 2, \ldots, n$ such that the sum

$$a_{i_1,i_2} + a_{i_2,i_3} + \cdots + a_{i_{n-1},i_n} + a_{i_n,i_1} \tag{6.1}$$

is as small as possible. This problem is among the most difficult in integer programming. However, certain aspects of it are surprisingly easy. For example, it is easy to show that the average of the terms in (6.1) over all permutations $i_1, i_2, \ldots, i_n$ is simply

$$\mu_1 = \frac{\sum_{i=1}^n \sum_{j=1}^n a_{ij}}{n-1}$$

and that the average of the squares of the terms in (6.1) is

$$\mu_2 = \frac{1}{n-1} \sum_{i=1}^n \sum_{j=1}^n a_{ij}^2 + \frac{1}{(n-1)(n-2)}$$
$$\cdot \left\{ \left( \sum_{i=1}^n \sum_{j=1}^n a_{ij} \right)^2 - \sum_{i=1}^n (AA)_{ii} - \sum_{i \neq j} (AA^T)_{ij} - \sum_{i=1}^n \sum_{j=1}^n (A^T A)_{ij} \right\}.$$

Here $M_{ij}$ denotes the number in position $ij$ of the matrix $M$.

It is possible to compute higher moments of the tour lengths (6.1), and based on preliminary results, it appears that these can be used to obtain good lower bounds on the difference between the longest and shortest tour lengths in (6.1). (2.13) can be used to obtain upper bounds in this length of the shortest tour, which may prove useful in branch and bound codes.

## REFERENCES

[1] E. R. BARNES, *Circular discs containing eigenvalues of normal matrices*, Linear Algebra Appl., 114/115 (1989), pp. 501–521.

[2] E. R. BARNES AND A. J. HOFFMAN, *On bounds for eigenvalues of real symmetric matrices*, Linear Algebra Appl., 40 (1981), pp. 217–223.

[3] ———, *Bounds for the spectrum of normal matrices*, Linear Algebra Appl., 201 (1994), pp. 79–90.

[4] A. Brauer and A. C. Mewborn, *The greatest distance between two characteristic roots of a matrix*, Duke Math. J., 26 (1959), pp. 653–661.

[5] N. G. de Bruijn, *Remarks on Hermitian matrices*, Linear Algebra Appl., 32 (1980), pp. 201–208.

[6] G. Elfving, *Optimal allocation in linear regression theory*, Annual Mathematical Statistics, 23 (1952), pp. 255–262.

[7] F. John, *Extremal problems with inequalities as subsiding conditions*, in Studies and Essays, Interscience, New York, 1948, pp. 187–204.

[8] S. Karlin and W. J. Studden, *Tchebycheff Systems: With Applications in Analysis and Statistics*, Interscience, New York, 1966.

[9] A. W. Marshall and I. Olkin, *Inclusion theorems for eigenvalues from probability inequalities*, Numer. Math., 6 (1964), pp. 98–102.

[10] L. Mirsky, *The spread of a matrix*, Mathematika, 3 (1956), pp. 127–130.

[11] ———, *Inequalities for normal and Hermitian matrices*, Duke Math. J., 24 (1957), pp. 591–598.

[12] D. S. Scott, *On the accuracy of the Gershgorin circle theorem for bounding the spread of a real symmetric matrix*, Linear Algebra Appl., 65 (1985), pp. 147–155.

[13] R. A. Smith and L. Mirsky, *The areal spread of matrices*, Linear Algebra Appl., 2 (1969), pp. 127–129.

[14] H. Wolkowicz and G. P. H. Styan, *Bounds for eigenvalues using traces*, Linear Algebra Appl., 29 (1980), pp. 471–506.

SIAM J. DISC. MATH.
Vol. 8, No. 3, pp. 359–387, August 1995

# TREE SPANNERS *

LEIZHEN CAI† AND DEREK G. CORNEIL‡

**Abstract.** A tree $t$-spanner $T$ of a graph $G$ is a spanning tree in which the distance between every pair of vertices is at most $t$ times their distance in $G$. This notion is motivated by applications in communication networks, distributed systems, and network design.

This paper studies graph-theoretic, algorithmic, and complexity issues about tree spanners. It is shown that a tree 1-spanner, if it exists, in a weighted graph with $m$ edges and $n$ vertices is a minimum spanning tree and can be found in $O(m \log \beta(m,n))$ time, where $\beta(m,n) = \min\{i \mid \log^{(i)} n \leq m/n\}$. On the other hand, for any fixed $t > 1$, the problem of determining the existence of a tree $t$-spanner in a weighted graph is proven to be NP-complete. For unweighted graphs, it is shown that constructing a tree 2-spanner takes linear time, whereas determining the existence of a tree $t$-spanner is NP-complete for any fixed $t \geq 4$. A theorem that captures the structure of tree 2-spanners is presented for unweighted graphs. For digraphs, an $O((m+n)\alpha(m,n))$ algorithm is provided for finding a tree $t$-spanner with $t$ as small as possible, where $\alpha(m,n)$ is a functional inverse of Ackerman's function. The results for tree spanners on undirected graphs are extended to "quasi-tree spanners" on digraphs. Furthermore, linear-time algorithms are derived for verifying tree spanners and quasi-tree spanners.

**Key words.** graph algorithm, NP-complete, tree spanner, spanning tree, distance

**AMS subject classifications.** 05C05, 05C12, 05C85, 68Q25, 68R10

## 1. Introduction.

### 1.1. Motivation.

A *t-spanner* of a graph $G$ is a spanning subgraph $H$ in which the distance between every pair of vertices is at most $t$ times their distance in $G$. This notion was introduced in 1987 by Peleg and Ullman [27], who showed that spanners can be used to construct synchronizers for transforming synchronous algorithms into asynchronous algorithms. A similar notion appeared in 1986 when Chew [16] studied approximations of complete Euclidean graphs by their planar subgraphs.

The key idea behind the notion of spanners is the approximation of pairwise vertex-to-vertex distances in the original graph by spanning subgraphs. The quality of the distance approximation by a $t$-spanner is measured by the parameter $t \geq 1$, which is referred to as the *stretch factor* of the $t$-spanner. This distance approximation property makes spanners quite useful in areas such as communication networks, distributed systems, motion planning, network design, and parallel machine architectures [5], [3], [6], [16], [27]–[29], [25]. For example, a sparse spanner (a spanner with few edges) of small stretch factor can be used to plan efficient routing schemes in a communication network while maintaining succinct routing tables [28]. Such a spanner can also be used as a substitute for its original network to reduce the construction cost of the network while keeping similar communication costs [29], [23]–[25]. In motion planning, when the input of a simple polygon is inaccurate, a special spanner of the visibility

* Received by the editors September 24, 1992; accepted for publication (in revised form) August 30, 1994. This research was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

† Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada. Current address: Department of Computer Science, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong (lcai@cs.cuhk.hk).

‡ Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada. (dgc@cs.toronto.edu).

graph of the input polygon, called the visibility skeleton, can be used to plan collision-free paths inside the real polygon [14].

In most applications, the sparseness of a spanner is the main concern; the sparsest $t$-spanner in a connected graph is a tree $t$-spanner, that is, a $t$-spanner that is a tree. Therefore, as far as sparseness is concerned, tree $t$-spanners are the best possible $t$-spanners. Furthermore, tree spanners have other interesting applications besides those mentioned for general graph spanners. Tree spanners of small stretch factors can be used to perform multisource broadcasts in a network [5], which can greatly simplify the message routing at the cost of only small delays in message delivery. The existence of a tree 2-spanner in a 2-connected network guarantees that the communication between operative sites will not be affected by any isolated failure of communication sites and lines [10]. There are also some surprising connections between tree 2-spanners and cycles in graphs. Certain cycle-extremal weighted graphs can be represented as a weighted union of tree 2-spanners ([8], where they were called tritrees), and graphs that contain tree 2-spanners ([7], where they were called trigraphs) appear to be the only graphs that require a large number of cycles to cover the edges of the graph exactly twice.

In this paper we consider graph-theoretic, algorithmic, and complexity issues about tree spanners. We study tree spanners in weighted, unweighted, and directed graphs, as well as "quasi-tree" spanners in directed graphs. By exploring graph-theoretic characterizations, we obtain several efficient algorithms for finding tree and quasi-tree $t$-spanners for some values of $t$. On the other hand, we show the intractability of determining the existence of tree and quasi-tree $t$-spanners for almost all other values of $t$. Furthermore, we present linear-time algorithms for verifying tree and quasi-tree $t$-spanners for all values of $t$.

**1.2. Notation and definitions.** We use the terminology of Bondy and Murty [9]. Graphs in this paper can be either weighted or unweighted, directed or undirected; they are connected graphs without loops, multiedges, and multiarcs. For any graph $G, V(G)$ denotes the vertex set of $G$; if $G$ is undirected then $E(G)$ denotes the edge set of $G$, and if $G$ is directed then $A(G)$ denotes the arc set of $G$. For a subset $V'$ of vertices of $G, G[V']$ denotes the induced subgraph of $G$ on $V'$; for a subset $E'$ of edges of $G, G[E']$ denotes the edge-induced subgraph of $G$ on $E'$. The induced subgraph $G[V(G) \setminus V']$ is denoted by $G - V'$, and the edge-induced subgraph $G[E(G) \setminus E']$ is denoted by $G - E'$. For any subgraph $H$ of $G, G - H$ denotes the subgraph obtained from $G$ by deleting edges (or arcs) of $H$ from $G$. Throughout this paper, unless specified otherwise, $m$ denotes the number of edges (or arcs) of $G$ and $n$ denotes the number of vertices of $G$. For any real number $x$, $\lfloor x \rfloor$ denotes the largest integer $\leq x$ and $\lceil x \rceil$ denotes the least integer $\geq x$.

We shall assume that the weight $w(e)$ of an edge (or arc) $e$ is a positive real number and regard an unweighted graph as a weighted graph where each edge (or arc) has unit weight. Given a subgraph $H$ of $G, w(H)$ denotes the *weight* of $H$, i.e., the sum of the weights of all edges in $H$; when $H$ is a (directed) path, $w(H)$ is the *length* of $H$. For any two vertices $x$ and $y$ of $G$, a path from $x$ to $y$ is an *$(x, y)$-path* and an $(x, y)$-path of minimum length is a *shortest $(x, y)$-path.* We use $d_G(x, y)$ to denote the *weighted distance* in $G$ from $x$ to $y$, i.e., the length of a shortest $(x, y)$-path in $G$. Note that $d_G(x, y) = \infty$ if there is no $(x, y)$-path in $G$ and $d_G(x, y) = d_G(y, x)$ if $G$ is undirected.

For any real number $t \geq 1$, a spanning subgraph $H$ of $G$ is a *$t$-spanner* if $d_H(x, y) \leq t \cdot d_G(x, y)$ for every pair of vertices $x$ and $y$ of $G$. The parameter $t$ is called the *stretch* *factor* of $H$. The *stretch index* of a spanner $H$ is the minimum number $t$ for which $H$ is a $t$-spanner. A $t$-spanner $H$ is a *minimal t-spanner* if no subgraph of $H$ is a $t$-spanner of $G$, a *minimum t-spanner* if it has the least number of edges among all $t$-spanners of $G$, and an *optimal t-spanner* if $H$ has the least weight among all $t$-spanners of $G$.

For an undirected graph $G$, a spanning subgraph $T$ of $G$ is a *tree t-spanner* if $T$ is both a $t$-spanner and a tree; in this case $G$ is *tree t-spanner admissible.* A spanning subgraph $T$ of $G$ is a *tree spanner* if it is a tree $t$-spanner for some $t \geq 1$ and a *minimum tree spanner* if it has the smallest stretch factor among all tree spanners of $G$. Thus a spanning tree of $G$ is always a tree spanner.

For a directed graph (digraph) $G = (V, A; w)$, we use $\tilde{G} = (V, E; \tilde{w})$ to denote its underlying undirected graph, i.e., $xy \in E$ iff either $(x, y) \in A$ or $(y, x) \in A$ or both, and

$$\tilde{w}(x,y) = \begin{cases} w((x,y)) & \text{if } (x,y) \in A, (y,x) \notin A, \\ w((y,x)) & \text{if } (y,x) \in A, (x,y) \notin A, \\ \min\{w((x,y)), w((y,x))\} & \text{if } (x,y), (y,x) \in A. \end{cases}$$

A vertex $x$ *reaches* vertex $y(y$ is *reachable* from $x)$ in $G$ if there is a directed $(x, y)$-path in $G$. It follows that $d_G(x, y) = \infty$ if $y$ is not reachable from $x$ in $G$.) A *spanning tree* of $G$ is a spanning subgraph $T$ that contains no directed cycle and such that $\tilde{T}$ is a tree. Then, as with undirected graphs, a *tree t-spanner* of a digraph is a spanning tree that is a $t$-spanner. A *quasi tree* of $G$ is a spanning subgraph $T$ such that $\tilde{T}$ is a tree; $T$ is a *quasi-tree t-spanner* if it is a $t$-spanner of $G$. Note that a quasi tree may contain a cycle consisting of two arcs $(x, y)$ and $(y, x)$. Other terms on tree spanners of undirected graphs are naturally extended to tree spanners and quasi-tree spanners of digraphs. However, a spanning tree (quasi tree) of a digraph is not necessarily a tree (quasi tree) spanner.

A few more definitions are in order for undirected graphs. (For simplicity, we will use these definitions for digraphs as well; it is understood that whenever we do so, we either refer to the underlying graphs or mean that the underlying graphs have the property.) For a connected graph, a *k-cut* is a set of $k$ vertices whose deletion disconnects the graph. A graph $G$ is *nonseparable* if it has no 1-cut and *triconnected* if it has no $k$-cut for $k \leq 2$. A *block* of a graph is a maximal nonseparable subgraph, and a *triconnected component* of a graph is a maximal triconnected subgraph. A vertex is *universal* if it is adjacent to all other vertices of the graph. An edge $e$ is a *binding edge* if its two ends form a minimal cut set. Two disjoint subgraphs $S$ and $S'$ of $G$ are *fully joined* if every vertex in $S$ is adjacent to every vertex in $S'$. A *star* is any complete bipartite graph $K_{1,n}$ with $n \geq 1$.

Finally, by the *tree t-spanner problem,* we usually mean the problem of finding a tree $t$-spanner in a graph, but it may refer to the problem of determining whether a graph contains a tree $t$-spanner when we talk about NP-completeness. Its meaning should be clear from the context. The meanings of other spanner problems are similarly defined.

**1.3. Observations.** We gather here some fundamental results on spanners in a graph. For simplicity, we will state our results only in terms of undirected graphs. These results also hold for digraphs and will be used in our discussions throughout the paper.

First, because edge weights are assumed to be positive, each of the following statements gives an equivalent definition of a $t$-spanner in a weighted graph.

THEOREM 1.1. *Let $H$ be a spanning subgraph of a weighted graph $G = (V, E; w)$. The following statements are then equivalent:*

(1) *$H$ is a $t$-spanner of $G$ (i.e., $d_H(x,y) \le t \cdot d_G(x,y)$ for every pair $x,y \in V$).*
(2) *For every edge $xy \in E, d_H(x,y) \le t \cdot d_G(x,y)$.*
(3) *For every edge $xy \in E \setminus E(H), d_H(x,y) \le t \cdot d_G(x,y)$.*
(4) *For every edge $xy \in E, d_H(x,y) \le t \cdot w(xy)$.*
(5) *For every edge $xy \in E \setminus E(H), d_H(x,y) \le t \cdot w(xy)$.*

*Proof.* The implications (1) $\Rightarrow$ (2), (2) $\Rightarrow$ (3), and (4) $\Rightarrow$ (5) are trivial. To see that (3) $\Rightarrow$ (4), we need only note that, for any edge $xy \in E$, we have $d_G(x,y) \le w(xy)$ and that $d_H(x,y) \le w(xy) \le t \cdot w(xy)$ if $xy \in E(H)$, since $t \ge 1$ and $w(xy) > 0$.

We now show that (5) $\Rightarrow$ (1). It suffices to show that $d_H(x,y) \le t \cdot d_G(x,y)$ for two arbitrary vertices $x,y$ of $G$. Let $P$ be a shortest $(x,y)$-path in $G$. Then for each edge $uv$ on $P$, if $uv \in E(H)$, then $d_H(u,v) \le w(uv) \le t \cdot w(uv)$, since $t \ge 1$ and $w(uv) > 0$; otherwise, $d_H(u,v) \le t \cdot w(uv)$ by statement (5). Therefore,

$$d_H(x,y) \le \sum_{uv \in P} d_H(u,v) \le t \sum_{e \in P} w(e).$$

Since $d_G(x,y) = \sum_{e \in P} w(e)$ by the choice of $P$, we obtain

$$d_H(x,y) \le t \cdot d_G(x,y)$$

This completes the proof. □

Quite often we will use statement (5) in the above theorem as the definition of a $t$-spanner, since it is easy to handle in most cases. Based on the above theorem, we can easily observe the following facts.

*Observation* 1.2. Let $F$ be a $t$-spanner of $G$ and $H$ be a $k$-spanner of $F$. Then $H$ is a $kt$-spanner of $G$.

*Observation* 1.3. For any $k > 1, H = (V, E'; w)$ is a $t$-spanner of $G = (V, E; w)$ iff $H' = (V, E'; w')$ is a $t$-spanner of $G' = (V, E; w')$, where $w'(e) = k \cdot w(e)$ for every $e \in E$.

It is easy to see that we can consider each block separately in dealing with most spanners, such as minimal, minimum, and optimal $t$-spanners. In particular, we can restrict our attention to nonseparable graphs when we deal with tree spanners.

*Observation* 1.4. Let $T$ be a spanning tree of a graph $G$. Then $T$ is a tree $t$-spanner of $G$ iff for every block $H$ of $G, T \cap H$ is a tree $t$-spanner of $H$.

Finally, for an unweighted graph $G$, the distance between any two vertices in $G$ is always an integer. Therefore, in light of statement (4) of Theorem 1.1, we need only consider $t$-spanners for integral $t$.

*Observation* 1.5. Let $H$ be a spanning subgraph of an unweighted graph $G$. Then $H$ is a $t$-spanner iff $H$ is a $\lfloor t \rfloor$-spanner.

**1.4. Outline of the paper.** We begin by discussing the verification of tree spanners and quasi-tree spanners in §2. We present $O(m)$ time algorithms for verifying tree $t$-spanners in graphs and in digraphs as well as quasi-tree $t$-spanners in digraphs.

In §3, we consider tree spanners in weighted graphs. We show that a tree 1-spanner, if it exists, is a minimum spanning tree and can be found in $O(m \log \beta(m,n))$ time, where $\beta(m,n) = \min\{i | \log^{(i)} n \le m/n\}$. On the other hand, we prove that, for any fixed $t > 1$, the problem of finding a tree $t$-spanner in a weighted graph is intractable.

In §4, we investigate tree spanners in unweighted graphs. We show that a tree 2-spanner can be constructed in linear time and that the tree $t$-spanner problem is

NP-complete for any fixed integer $t \geq 4$. We also present a skeleton tree theorem, which captures the structure of tree 2-spanners.

We deal with tree spanners of digraphs in §5. We present an $O((m+n)\alpha(m+n,n))$ algorithm for finding a minimum tree spanner in a digraph, where $\alpha(m,n)$ is a functional inverse of Ackerman's function. For general digraphs, we extend the results of §§3 and 4 to quasi-tree spanners.

We conclude the paper with a short summary and some open problems in §6.

**2. Verifying a tree *t*-spanner.** Given a graph $G$, a spanning tree $T$, and a positive number $t$, we wish to verify whether $T$ is a tree $t$-spanner of $G$. We may also wish to know if $T$ is a tree spanner and, if it is, determine its stretch index, i.e., the smallest $t$ for which $T$ is a $t$-spanner. Similar problems can also be explored for quasi-tree spanners. These problems will come forth naturally in later sections, and, for convenience, we will refer to these problems as *tree spanner verification problems.*

In this section, we will provide linear-time algorithms for the above verification problems. The main results of this section are summarized in the following theorem, which will be used in later sections.

THEOREM 2.1. *Let $D$ and $G$ be directed and undirected weighted graphs, respectively. Let $S$ and $T$ be spanning trees of $D$ and $G$, respectively. Let $Q$ be a quasi tree of $D$. Then the following problems can be solved in $O(m)$ time:*

(a) *Determine the stretch index of $T$.*

(b) *Is $S$ a tree spanner? If it is, determine its stretch index.*

(c) *Is $Q$ a quasi-tree spanner? If it is, determine its stretch index.*

**2.1. A verification algorithm paradigm.** We first describe an algorithm paradigm for tree spanner verification problems. Clearly, statement (5) of Theorem 1.1 provides us with a simple method for solving these problems. By taking this approach, we need to compute the distances in $T$ of all $m-n+1$ vertex pairs defined by nontree edges. Thus the cost of distance computation dominates the running time of verification algorithms based on this approach. If we compute the distance of each vertex pair directly and independently, it may take $O(mn)$ time to compute these distances, since each distance may take $O(n)$ time to compute. We can reduce the cost to $O(n^2)$ by computing all pairwise distances in $T$ together. Unfortunately, this is not satisfactory for sparse graphs. To speed up the verification, we need a better way to compute the distances of these $m-n+1$ vertex pairs.

For simplicity, we will describe an algorithm for verifying a tree $t$-spanner in an undirected graph. The algorithm is easily extended to other verification problems. Several definitions are in order. A *rooted tree* $T$ is a tree with a distinguished vertex $r$, called the *root.* For any two vertices $x$ and $y$ in $T$, if $x$ is on the path from $r$ to $y$, then $x$ is an *ancestor* of $y$. The *least common ancestor* of $x$ and $y$, denoted by $\mathrm{LCA}(x,y)$, is the common ancestor $z$ of $x$ and $y$ such that for any common ancestor $z'$ of $x$ and $y$, $z'$ is an ancestor of $z$. We will take advantage of the structure of a tree to compute distances more efficiently. To achieve this, we arbitrarily choose a vertex $r$ to be the root of $T$ and then label vertices of $T$ in such a way that distance $d_T(x,y)$ of any vertex pair $(x,y)$ can be quickly computed from the labels of $x, y$, and $\mathrm{LCA}(x,y)$. Notice that $d_T(x,y) = d_T(x, \mathrm{LCA}(x,y)) + d_T(\mathrm{LCA}(x,y),y)$.

```
ALGORITHM VERIFICATION$(G, T, t)$ {Verify if $T$ is a tree $t$-spanner of $G$.}
Input: A graph $G$, a spanning tree $T$ and a positive number $t$;
Output: "Yes" if $T$ is a tree $t$-spanner; "No" otherwise.

begin
1.    Arbitrarily choose a vertex $r$ as the root of $T$;
2.    Compute a label label$(x)$ for each vertex $x$ of $T$;
3.    Compute LCA$(x, y)$ for every nontree edge $xy$ of $G$;
4.    for each nontree edge $xy$ do
      begin
4.1.      Compute $d_T(x, y)$ by using the labels of $x, y$ and LCA$(x, y)$;
4.2.      if $d_T(x, y) > t \cdot w(xy)$ then output "No" EXIT;
      end;
5.    output "Yes";
end.
```

By statement (5) of Theorem 1.1, we note that the stretch index of $T$ equals

$$\max\{1, d_T(x, y)/w(xy) | xy \in E(G) \setminus E(T)\}.$$

The above algorithm can thus be modified (line (4.2)) to compute the stretch index of $T$ as well. To apply this algorithm to a digraph $D$, we take the underlying tree $\tilde{T}$ of $D$'s spanning tree (quasi tree) $T$ to define a rooted tree and carry out the computation of the algorithm with respect to this rooted tree. In this case, when we notice that $x$ reaches $y$ in $T$ iff $d_T(x, y)$ is finite, we can use the algorithm to check the reachability from $x$ to $y$ in $T$ as well.

Regarding the complexity of the algorithm, we see that line (1) is trivial. To carry out the computation of line (3), we use a linear-time least common ancestor algorithm of Harel and Tarjan [21]. Clearly, line (4.2) takes $O(1)$ time. In the next two subsections, we will discuss efficient implementations of line (2) and line (4.1) for undirected graphs and digraphs so as to obtain the results in Theorem 2.1.

**2.2. Undirected case.** Let $G$ be an undirected weighted graph and $T$ be a spanning tree of $G$. Arbitrarily choose a vertex $r$ in $T$ as the root of $T$. For each vertex $x$ in $T$, label $x$ by the root-to-vertex distance of $x$, i.e., label$(x) = d_T(r, x)$. See Fig. 1 for an example.

We show that for any two vertices $x$ and $y$, their distance $d_T(x, y)$ in $T$ can be computed in constant time from label$(x)$, label$(y)$, and label(LCA$(x, y)$). Notice that

$$\text{label}(x) = d_T(r, x) = d_T(r,\ \text{LCA}(x, y)) + d_T(\text{LCA}(x, y), x)$$

and

$$\text{label}(y) = d_T(r, y) = d_T(r,\ \text{LCA}(x, y)) + d_T(\text{LCA}(x, y), y).$$

We obtain

$$d_T(x, y) = \ \text{label}(x) + \ \text{label}(y) - 2 \cdot \text{label}(\text{LCA}(x, y)).$$

Therefore $d_T(x, y)$ can be determined in $O(1)$ time.

It is easy to see that by either a depth-first or a breadth-first search from the root $r$, we can obtain the labels for all vertices of $T$. Thus line (2) of algorithm VERIFICATION can be carried out in $O(n)$ time. Furthermore, line (4.1) can be done in $O(1)$ time; thus step (4) takes $O(m - n)$ time. Therefore, the overall time

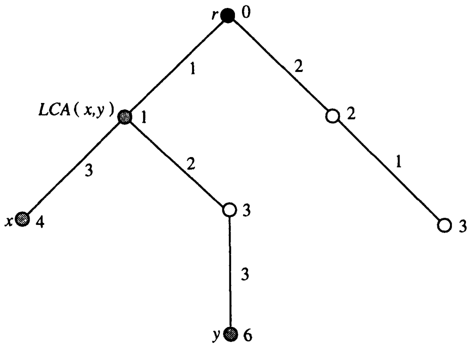


FIG. 1. *Labeling the vertices of T by their root-to-vertex distances.*

of the algorithm is linear. Thus verifying a tree $t$-spanner takes linear time. Once $d_T(x,y)$ is obtained for every nontree edge $xy$, we can easily determine the stretch index of $T$ in linear time, thereby establishing Theorem 2.1(a).

**2.3. Directed case.** Let $D$ be a weighted digraph and $S$ be a spanning tree of $D$. Then by statement (5) of Theorem 1.1, it is easy to see that $S$ is a tree spanner of $D$ iff $x$ reaches $y$ in $S$ for any nontree arc $(x,y)$ of $D$. Therefore, in order to verify that $S$ is a tree spanner of $D$, we need to verify that $S$ preserves reachability for each nontree arc of $D$. We apply VERIFICATION to $D$ and $S$ together with the underlying tree $\tilde{S}$ of $S$.

Arbitrarily choose a vertex $r$ in $\tilde{S}$ as the root of $\tilde{S}$. An edge $xy$ of $\tilde{S}$, where $x$ is an ancestor of $y$, is a *forward edge* if $(x,y)$ is an arc of $S$ and a *backward edge* if $(y,x)$ is an arc of $S$. For an arbitrary vertex $x$ in $\tilde{S}$, let $P(x)$ be the unique $(r,x)$-path in $\tilde{S}$. Label $x$ by a triple $(b(x), f(x), l(x))$, where

$b(x)$ is the number of backward edges on $P(x)$,

$f(x)$ is the number of forward edges on $P(x)$, and

$l(x)$ is the total weight of forward edges on $P(x)$ minus the total weight of backward edges on $P(x)$.

The first two components in the triple are used for verifying reachability; the third one is used for computing distances. It is easy to see that all vertex labels can be computed in $O(n)$ time by either a depth-first or breadth-first search of $\tilde{S}$ from the root $r$. See Fig. 2 for an example. Backward and forward edges are indicated by upward and downward arrows, respectively.

For any two vertices $x$ and $y$ of $S$, it is easy to see that $x$ reaches $y$ in $S$ iff

$$f(x) = f(\text{LCA}(x,y)) \text{ and } b(y) = b(\text{LCA}(x,y)).$$

Since these two conditions can be easily checked in $O(1)$ time, the overall cost of verifying a tree spanner is linear. Furthermore, it is not difficult to see that if $x$ reaches $y$ in $S$, then

$$d_S(x,y) = l(y) - l(x).$$

Thus $d_S(x,y)$ can be computed in $O(1)$ time. Therefore, it takes linear time to compute the stretch index of a tree spanner of $D$. This also implies that verifying a tree $t$-spanner of $D$ takes linear time. Hence, we have Theorem 2.1(b).

We now turn our attention to quasi-tree spanners. Let $Q$ be a quasi tree of $D$. Like the situation for tree spanners in digraphs, in order to verify that $Q$ is a quasi-tree

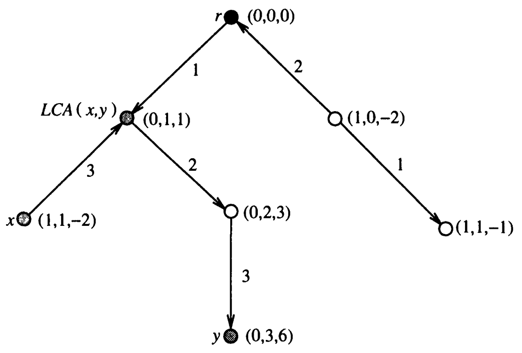


FIG. 2. *Labeling the vertices of $\tilde{S}$ by triples.*

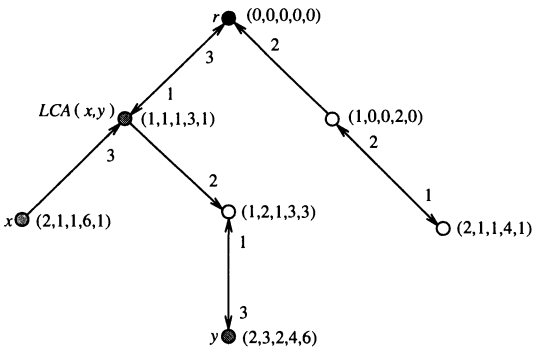


FIG. 3. *Labeling the vertices of $\tilde{Q}$ by quintuples.*

spanner of $D$, we need to verify that $Q$ preserves reachability for each arc of $D$ that is not in $Q$. We apply VERIFICATION to $D$ and $Q$ together with the underlying tree $\tilde{Q}$ of $Q$.

Arbitrarily choose a vertex $r$ in $\tilde{Q}$ as the root of $\tilde{Q}$. An edge $xy$ of $\tilde{Q}$ is a *double edge* if both $(x,y)$ and $(y,x)$ are arcs in $Q$. Note that a double edge is also a forward edge and a backward edge. For an arbitrary vertex $x$ in $\tilde{Q}$, let $P(x)$ be the unique $(r,x)$-path in $\tilde{Q}$. Label $x$ by a quintuple $(b(x), f(x), d(x), lb(x), lf(x))$, where

$b(x)$ is the number of backward edges on $P(x)$,

$f(x)$ is the number of forward edges on $P(x)$,

$d(x)$ is the number of double edges on $P(x)$,

$lb(x)$ is the total weight of arcs of $Q$ corresponding to backward edges on $P(x)$, and

$lf(x)$ is the total weight of arcs of $Q$ corresponding to forward edges on $P(x)$. The first three components in the quintuple are used for verifying reachability; the last two are used for computing distances. Again all vertex labels can be computed in $O(n)$ time by either a depth-first or breadth-first search of $\tilde{Q}$ from the root $r$. See Fig. 3 for an example. Each double edge is shown with both an upward and downward arrow; the numbers beside the arrows indicate the weights of the corresponding arcs of $Q$.

For any two vertices $x$ and $y$, it is easy to see that $x$ reaches $y$ iff

$$f(x) - f(\mathrm{LCA}(x,y)) = d(x) - d(\mathrm{LCA}(x,y))$$

and

$$b(y) - b(\mathrm{LCA}(x, y)) = d(y) - d(\mathrm{LCA}(x, y)).$$

We can check these two conditions in $O(1)$ time and thus verify a quasi-tree spanner in linear time. Furthermore, if $x$ reaches $y$, then we have

$$d_Q(x, y) = lb(x) - lb(\mathrm{LCA}(x, y)) + lf(y) - lf(\mathrm{LCA}(x, y)).$$

Thus $d_Q(x, y)$ can be computed in $O(1)$ time. Therefore, it takes linear time to compute the stretch index of a quasi-tree spanner of $D$, which implies that verifying a quasi-tree $t$-spanner takes linear time. This establishes Theorem 2.1(c), and thus completes the proof of Theorem 2.1.

## 3. Tree spanners in weighted graphs.

In this section, we consider the complexity of tree spanner problems on weighted graphs. By statement (5) of Theorem 1.1, a spanning subgraph $H$ of a weighted graph $G = (V, E; w)$ is a $t$-spanner iff for every edge $xy \in E \setminus E(H)$, we have $d_H(x, y) \leq t \cdot w(xy)$. We present an $O(m \log \beta(m, n))$ algorithm for finding a tree 1-spanner in a weighted graph; on the other hand, we show that for any fixed $t > 1$, the tree $t$-spanner problem is NP-complete on weighted graphs. This completely settles the issue of complexity of tree spanner problems for weighted graphs. Henceforth in this section, by a graph we mean a weighted graph.

### 3.1. Finding a tree 1-spanner.

Let $G = (V, E; w)$ be a weighted graph, and let $H$ be a 1-spanner of $G$. Since $H$ is a subgraph of $G$, it is clear that $d_H(x, y) \geq d_G(x, y)$ for any two vertices $x, y \in V$. Therefore, $d_H(x, y) = d_G(x, y)$ for any $x, y \in V$, i.e., $H$ preserves pairwise distances in $G$.

The distance-preserving property of a 1-spanner is useful in many applications. For example, a 1-spanner of a communication network can be used as a substitute for the original network without introducing any extra delay in communication. It is also closely related to the metric realization problem [2], [31], [20] (to construct a graph with a minimum total weight that realizes an $n$-by-$n$ symmetric distance matrix $M = (m_{i,j})$). To see this, we construct a complete graph $G(M)$ on $n$ vertices such that $w(ij) = m_{i,j}$ for each edge $ij$ of $G(M)$; then the optimal 1-spanner of $G(M)$ gives an optimal realization of $G$ if we allow only $n$ vertices. Regarding tree 1-spanners, we see that a tree 1-spanner is a distance-preserving spanning tree. Therefore, using a tree 1-spanner of a network to perform broadcast in the network guarantees the minimum delay. Furthermore, a tree 1-spanner can also be used as a compact encoding of the distance information of $G$.

*Remark.* Because of the connection between 1-spanners and metric realizations, some results in this subsection regarding minimal 1-spanners have appeared in the literature on metric realizations. In particular, Corollary 3.3 has been previously obtained by Hakimi and Yau [20].

We shall first explore the properties of 1-spanners of $G$. These properties lead us to polynomial algorithms for constructing a minimum or an optimal 1-spanner in $G$, and these algorithms can be used to find a tree 1-spanner in $G$. We then establish a relationship between a tree 1-spanner and a minimum spanning tree and use this relationship to derive a more efficient algorithm for finding a tree 1-spanner.

LEMMA 3.1. *Let $H$ be a 1-spanner of a weighted graph $G$. Then $H$ is minimal iff $d_{H-xy}(x, y) > w(xy)$ for every edge $xy$ of $H$.*

*Proof.* If there is an edge $xy$ of $H$ such that $d_{H-xy}(x, y) \leq w(xy)$, then $H - xy$ is a 1-spanner of $H$ by statement (5) of Theorem 1.1, and thus $H - xy$ is a 1-spanner of $G$ by Observation 1.2. Hence, $H$ is not minimal. Conversely, if $H$ is not minimal, then there is an edge $uv$ of $H$ such that $H - uv$ is a 1-spanner of $G$. This implies that $d_{H-uv}(u, v) \leq w(uv)$. □

We are now ready to present necessary and sufficient conditions for an edge of $G$ to be in a minimal 1-spanner. Bear in mind that edge weights of $G$ are positive.

THEOREM 3.2. *Let $H$ be a minimal 1-spanner of a weighted graph $G$, and let $xy$ be an edge of $G$. Then the following statements are equivalent:*

(1) *Edge $xy$ belongs to $H$.*

(2) *For every vertex $z \in V \setminus \{x, y\}, d_G(x, z) + d_G(z, y) > w(xy)$.*

(3) *Distance $d_{G-xy}(x, y) > w(xy)$.*

*Proof.* (1) $\Rightarrow$ (2). Let $z$ be an arbitrary vertex in $V \setminus \{x, y\}$. If $d_H(x, z) + d_H(z, y) \leq w(xy)$, then $d_H(x, z) < w(xy)$, since edge weights are positive, and thus any shortest $(x, z)$-path $P$ in $H$ avoids edge $xy$. Let $H' = H - xy$. Then $d_{H'}(x, z) = d_H(x, z)$, since $P$ is in $H'$. Similarly, $d_{H'}(z, y) = d_H(z, y)$.

By the definition of distance, we have

$$d_{H'}(x, y) \leq d_{H'}(x, z) + d_{H'}(z, y).$$

Therefore,

$$d_{H'}(x, y) \leq d_H(x, z) + d_H(z, y) \leq w(xy).$$

Then by Lemma 3.1, $H$ is not a minimal 1-spanner, which is a contradiction. Hence,

$$d_H(x, z) + d_H(z, y) > w(xy).$$

Since $H$ is a 1-spanner of $G$, we now have $d_H(x, z) = d_G(x, z)$ and $d_H(z, y) = d_G(z, y)$. Therefore, $d_G(x, z) + d_G(z, y) > w(xy)$.

(2) $\Rightarrow$ (3). Let $G' = G - xy$. By the definition of distance, we have

$$d_{G'}(x, y) = \min_{z \in V \setminus \{x, y\}} \{d_{G'}(x, z) + d_{G'}(z, y)\}$$

It follows from statement (2) that $d_{G'}(x, y) > w(xy)$.

(3) $\Rightarrow$ (1). Because edge weights are positive, statement (3) implies that $xy$ is the only $(x, y)$-path in $G$ with length $\leq w(xy)$. Since $H$ is a 1-spanner of $G$, we have $d_H(x, y) \leq w(xy)$. Therefore, $xy$ must appear in $H$. □

COROLLARY 3.3 (Hakimi and Yau [20]). *Every weighted graph $G$ has a unique minimal 1-spanner.*

*Proof.* By Theorem 3.2, each edge of a minimal 1-spanner of $G$ is uniquely determined. □

Since both a minimum and an optimal 1-spanner of $G$ are minimal 1-spanners, Corollary 3.3 implies the following result.

COROLLARY 3.4. *For any weighted graph $G$, the following statements are equivalent:*

(1) *$H$ is a minimal 1-spanner of $G$.*

(2) *$H$ is a minimum 1-spanner of $G$.*

(3) *$H$ is an optimal 1-spanner of $G$.*

If $G$ contains a tree 1-spanner $T$, then $T$ is also a minimal 1-spanner. Thus Corollary 3.3 also implies the uniqueness of a tree 1-spanner.

COROLLARY 3.5. *A weighted graph can contain at most one tree 1-spanner.*

In light of Theorem 3.2 and Corollary 3.4, we see that the minimum (or optimal) 1-spanner of a weighted graph can be constructed in polynomial time, since for each edge of $G$, we can use either statement (2) or statement (3) of Theorem 3.2 to decide if the edge belongs to the minimal 1-spanner $H$ of $G$. For a single edge, it is more efficient to use statement (3) to determine whether the edge is in $H$ if pairwise distances are not given. However, it seems that the algorithm using statement (2) is more efficient for constructing $H$, especially when pairwise distances are given; using the fastest known algorithm to compute pairwise distances [17], we can implement the algorithm in $O(mn + n^2 \log n)$ time.

THEOREM 3.6. *The minimum (or optimal) 1-spanner of a weighted graph can be found in $O(mn + n^2 \log n)$ time.*

Clearly, we can find the tree 1-spanner (if it exists) of $G$ in $O(mn + n^2 \log n)$ time by first computing the minimal 1-spanner of $G$ and then checking if it is a tree. However, this approach is not efficient. To obtain a more efficient algorithm for computing the tree 1-spanner, we will establish a relationship between the tree 1-spanner of $G$ and a minimum spanning tree of $G$.

THEOREM 3.7. *The tree 1-spanner of a weighted graph $G$ is a minimum spanning tree. Moreover, every tree 1-spanner admissible weighted graph contains a unique minimum spanning tree.*

*Proof.* Let $T$ be a minimum spanning tree of $G$. We first claim that $T$ is contained in any 1-spanner $H$ of $G$. To see this, let $xy$ be an arbitrary edge of $T$ and $P$ be a shortest $(x, y)$-path in $G - xy$. Then there is an edge $e$ on $P$ that is not in $T$. If $w(P) \leq w(xy)$, then $w(e) < w(xy)$, since edge weights are positive and $P$ contains at least two edges. This implies that $T + e - xy$ is a spanning tree whose weight is less than $w(T)$, contrary to $T$ being a minimum spanning tree. Therefore, $d_{G-xy}(x, y) = w(P) > w(xy)$, and, by Theorem 3.2, $xy$ is an edge of $H$.

Now let $T'$ be the tree 1-spanner of $G$. By the above claim, $T$ is a subgraph of $T'$. Since both $T$ and $T'$ are spanning trees of $G$, we have $T = T'$. The theorem follows immediately. □

In light of the above theorem, we have the following algorithm for constructing the tree 1-spanner of $G$: we first find a minimum spanning tree $T$ of $G$ and then verify whether $T$ is a 1-spanner of $G$. A minimum spanning tree can be found in $O(m \log \beta(m, n))$ time [18], where $\beta(m, n) = \min\{i \mid \log^{(i)} n \leq m/n\}$ and $\log^{(i)} n$ is defined by $\log^{(0)} n = n, \log^{(i)} n = \log \log^{(i-1)} n$ for $i \geq 1$. Since verification takes linear time by Theorem 2.1(a), we have the following result.

THEOREM 3.8. *The tree 1-spanner of a weighted graph can be found in $O(m \log \beta(m, n))$ time.*

*Remark.* The above algorithm can be applied to find tree 1-spanners in a weighted graph $G$ where zero weight is allowed. Let $G_0$ be the subgraph of $G$ induced by zero-weighted edges of $G$ and $Z_1, \ldots, Z_k$ be the connected components of $G_0$. We construct a new weighted graph $G'$ as follows: contract such $Z_i$ to a single vertex $z_i$, remove all loops, and, for all parallel edges (formed from the contraction) between two vertices, delete all but one with the lightest weight. Then $G'$ is a weighted graph with no zero weight on edges, and its tree 1-spanner can be found by the algorithm in this subsection. It is not hard to see that $G$ admits a tree 1-spanner iff $G'$ admits one. Actually, a tree 1-spanner of $G$ can be obtained from the tree 1-spanner of $G'$ by "replacing" each $z_i$ with a spanning tree of $Z_i$. However, $G$ may contain many tree 1-spanners when it has zero-weighted edges. In fact, the number of tree 1-spanners in $G$ equals the product of the number of spanning trees of $Z_i, 1 \leq i \leq k$. Figure 4 illustrates the above construction.

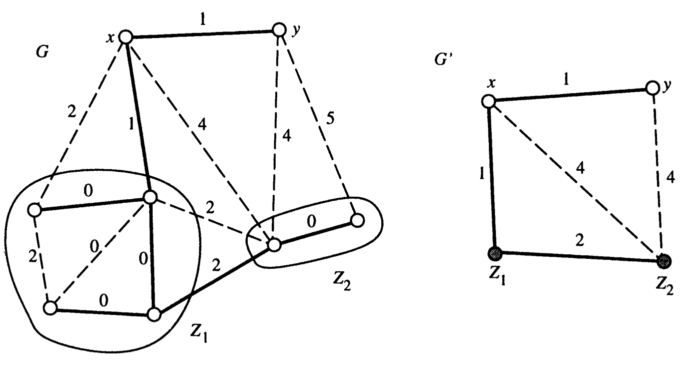


FIG. 4. *Graphs $G, G'$ and their tree 1-spanners (solid edges).*

TABLE 1
*Bridge length (number of edges) and edge weights of G (note $t = 1 + \varepsilon$).*

| | | Edge weights $w(e)$ | | |
|---|---|---|---|---|
| $t$ | Bridge length | Bridge edge | Literal edge | Clause edge |
| $1 < t < 2$ | 1 | 1 | $\lfloor 1/\varepsilon \rfloor$ | $\lceil (1 + 2\lceil 1/\varepsilon \rceil)/\varepsilon \rceil$ |
| $2 \le t < 4$ | $2\lfloor \varepsilon \rfloor$ | 1 | 2 | $\lceil (4 + 2\lfloor \varepsilon \rfloor)/\varepsilon \rceil$ |

**3.2. NP-completeness for $t > 1$.** We now consider the complexity of finding tree $t$-spanners ($t > 1$) in a weighted graph. It turns out that the tree $t$-spanner problem on weighted graphs is intractable for any fixed rational number $t > 1$. As a consequence, the minimum $t$-spanner problem on weighted graphs is intractable for any fixed rational number $t > 1$. Furthermore, we deduce that the optimal $t$-spanner problem on weighted graphs is also intractable for any fixed rational number $t > 1$. In this subsection, we assume that all edge weights are positive rational numbers and that $t > 1$ is a fixed rational number.

Recall that an instance $(U, C)$ of 3SAT (cf. [LO2] in [19]) consists of a set $U$ of $n$ distinct Boolean variables and a collection $C$ of $m$ 3-element clauses over $U$. For any variable $u \in U$, both $u$ and $\bar{u}$ are *literals*; for a truth assignment $\xi$, a literal $l$ is *true* if $\xi(l) = 1$ and *false* otherwise.

THEOREM 3.9. *For any fixed rational number $t > 1$, it is NP-complete to determine whether a weighted graph contains a tree $t$-spanner, even if all edge weights are positive integers.*

*Proof.* It is clear that the problem is in NP. To establish the NP-completeness of the problem, we present a polynomial transformation from 3SAT. Here we will only consider the case $1 < t < 4$; the proof for $t \ge 4$ is the same as that for unweighted graphs (see Theorem 4.10 of §4.3), where a more complicated construction is employed.

Let $t \in (1, 4)$ be a fixed rational number, and, for convenience, let $\varepsilon = t - 1$; then $0 < \varepsilon < 3$. For an arbitrary instance $(U, C)$ of 3SAT, we construct a weighted graph $G$ such that $C$ is satisfiable iff $G$ admits a tree $t$-spanner. Graph $G$ is constructed as follows:

1. Take a vertex $x$ and vertices $U' = \{u_1, \bar{u}_1, \ldots, u_n, \bar{u}_n\}$, where $n = |U|$, and construct a star $H$ centered at $x$ by joining $x$ to each vertex in $U'$. Each vertex in $U'$ is a *literal vertex* and each edge in $H$ is a *literal edge*.

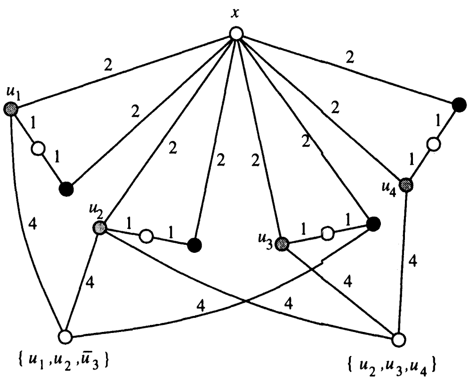


FIG. 5. *The graph $g$ for $t = 2.5$ and $C = \{\{u_1, u_2, \bar{u}_3\}, \{u_2, u_3, u_4\}\}$.*

2. Create a new vertex $c$ for each clause in $C$ and add an edge between $c$ and each of its three distinct literal vertices in $H$. These new vertices and edges are called *clause vertices* and *clause edges,* respectively.

3. Connect each pair $u_i, \bar{u}_i$ of literal vertices by a distinct path (whose length is specified in Table 1), called a *bridge,* to complete the construction of $G$. Each edge on the bridge is called a *bridge edge.*

4. For each edge $e$ of $G$, assign it weight $w(\varepsilon)$ according to Table 1.

Figure 5 shows an example of $G$. It is easy to see that $G$ can be constructed in polynomial time.

Now, for any tree $t$-spanner $T$ of $G$ (Remember that $1 < t < 4$ and $\varepsilon = t - 1$), we note the following two important properties:

P1. Every bridge is contained in $T$.

P2. For every pair $xu_i, x\bar{u}_i$ of literal edges, exactly one of them belongs to $T$.

To see property P1, let $ab$ be an arbitrary bridge edge and $P_{ab}$ be an $(a, b)$-path in $T$. Suppose that $ab$ is not in $T$. If $0 < \varepsilon < 1$, then $P_{ab}$ contains either two literal edges or at least two clause edges; otherwise $1 \leq \varepsilon \leq 3$ and $P_{ab}$ contains all other $2\lfloor\varepsilon\rfloor - 1$ bridge edges on the bridge containing $ab$ and either two literal edges or at least two clause edges. It is readily checked that we would then have $d_T(a, b) > t \cdot w(ab)$ in both cases, which contradicts $T$ being a $t$-spanner. Hence $ab$ belongs to $T$.

To see property P2, without loss of generality, we consider edge $xu_i$. If the $(x, u_i)$-path in $T$ contains neither edge $xu_i$ nor $x\bar{u}_i$, then it must contain a literal edge and at least two clause edges. It is easy to see that $d_T(x, u_i) > t \cdot w(xu_i)$ if $0 < \varepsilon < 1$; otherwise $1 \leq \varepsilon < 3$ and

$$d_T(x, u_i) \geq 2 + 2\left\lceil \frac{4 + 2\lfloor\varepsilon\rfloor}{\varepsilon} \right\rceil.$$

It can be shown that

$$\left\lceil \frac{4 + 2\lfloor\varepsilon\rfloor}{\varepsilon} \right\rceil > \varepsilon \quad \text{for} \quad 1 \leq \varepsilon < 3$$

(consider $\varepsilon \in [1, 2)$ and $[2, 3)$ separately). Thus it follows that $d_T(x, u_i) > t \cdot w(xu_i)$ for $1 < t < 4$, which contradicts $T$ being a tree $t$-spanner. So at least one of $xu_i, x\bar{u}_i$ is in $T$. Since $T$ is a tree, it can be deduced from property P1 that exactly one of them is in $T$.

We now prove that $C$ is satisfiable iff $G$ contains a tree $t$-spanner. Suppose that $C$ is satisfiable and let $\xi$ be a satisfying truth assignment for $C$. Call a literal vertex of $G$ a *true vertex* if its corresponding literal is a true literal under $\xi$. Construct a spanning tree $T$ of $G$ by taking all bridge edges, all literal edges incident with true vertices, and, for each clause, an arbitrary clause edge that is incident with a true vertex. Since $C$ is satisfied by $\xi$, it is clear that $T$ is a spanning tree and each clause vertex is a leaf.

To see that $T$ is a $t$-spanner, we note that, for any literal edge $xl$ not in $T$, the $(x,l)$-path in $T$ consists of a literal edge and a bridge and that, for any clause edge $cl'$ not in $T$, the $(c,l')$-path in $T$ consists of a clause edge, two literal edges, and at most one bridge. It is a routine matter to check that $d_T(x,l) \leq t \cdot w(xl)$ and $d_T(c,l') \leq t \cdot w(cl')$. Therefore, it follows from statement (5) of Theorem 1.1 that $T$ is a tree $t$-spanner of $G$.

Conversely, suppose that $T$ is a tree $t$-spanner of $G$. Then by property P2, exactly one of the two literal edges $xu_i, x\bar{u}_i$ is contained in $T$. Therefore we can define a truth assignment $\xi_T$ be setting $\xi_T(u_i) = 1$ if $xu_i \in E(T)$ and $\xi_T(u_i) = 0$ if $x\bar{u}_i \in E(T)$. It remains to be shown that $\xi_T$ satisfies $C$.

It is easy to see by property P1 that any two literal vertices are connected by a path in $T$ that avoids clause vertices. Thus each clause vertex is a leaf of $T$. Suppose that there is a clause vertex $c$ which contains only false literals under $\xi_T$. Then for a clause edge $cl$ not in $T$, the $(c,l)$-path in $T$ consists of a clause edge, two literal edges, and two bridges. If $0 < \varepsilon < 1$, then it is easy to check that $d_T(c,l) \geq t \cdot w(cl)$ (notice $1/\varepsilon > 1$). Otherwise $1 \leq \varepsilon < 3$, and again it is a routine matter to check that $d_T(c,l) \geq t \cdot w(cl)$ (notice $2\lfloor \varepsilon \rfloor > \varepsilon$); this contradicts $T$ being a $t$-spanner. Therefore, each clause in $C$ contains at least one true literal under $\xi_T$, and thus $C$ is satisfiable. The proof is complete. $\square$

Since a tree $t$-spanner has the least number of edges among all $t$-spanners, Theorem 3.9 implies that finding a minimum $t$-spanner in a weighted graph is intractable for any fixed rational number $t > 1$.

COROLLARY 3.10. *For any fixed rational number $t > 1$, it is NP-complete to determine, given a weighted graph $G$ and a positive integer $K$, whether $G$ contains a $t$-spanner with at most $K$ edges, even if all edge weights are positive integers.*

Furthermore, the tree $t$-spanner $T$ in the proof of Theorem 3.9 also achieves the minimum total weight (sum of weights of all edges in the spanning subgraph) over all $t$-spanners of $G$. Therefore, Theorem 3.9 also implies the following result.

COROLLARY 3.11. *For any fixed rational number $t > 1$, it is NP-complete to determine, given a weighted graph $G$ and a positive rational number $W$, whether $G$ contains a $t$-spanner of total weight at most $W$, even if all edge weights are positive integers.*

## 4. Tree spanners in unweighted graphs.

We now consider tree spanners in unweighted graphs, which can be considered as a special case of tree spanners in weighted graphs, i.e., tree spanners in unit-weighted graphs. Henceforth in this section, by a graph we always mean an unweighted graph. By statement (5) of Theorem 1.1, a spanning subgraph $H$ is a $t$-spanner of a graph $G = (V, E)$ iff for every edge $xy \in E \setminus E(H)$ we have $d_H(x,y) \leq t$.

In light of Observation 1.5, we need to consider only tree $t$-spanners for integral $t$. Clearly, a graph contains a tree 1-spanner iff it itself is a tree. We show that a tree 2-spanner in a graph can be found in linear time. We also study the structure of tree 2-spanners and give a characterization of tree 2-spanner-admissible graphs.

In particular, we present a structural theorem for tree 2-spanners in terms of the "skeleton tree" of a graph. This structural theorem is useful in dealing with various tree 2-spanner problems. On the other hand, we show that the tree $t$-spanner problem is NP-complete for any fixed $t \geq 4$. The complexity of the tree 3-spanner problem remains an open issue.

**4.1. Finding a tree 2-spanner.** Our main concern is to find a tree 2-spanner in a graph. In order to design an efficient tree 2-spanner-finding algorithm, we first investigate the structure of tree 2-spanners in a graph. We then describe a linear-time algorithm. Furthermore, we give a characterization of tree 2-spanner-admissible graphs in terms of decomposition. Because of Observation 1.4, we can restrict our attention to nonseparable graphs.

*Remark.* It is interesting to note that tree 2-spanner-admissible graphs coincide with trigraphs introduced by Bondy [7] in his work on cycle double covers; our characterization results are quite similar to his. In particular, our decomposition theorem (Theorem 4.4) for tree 2-spanner-admissible graphs and Lemma 4.1 have been previously obtained by Bondy. They are reformulated here in our terminology for the sake of completeness.

LEMMA 4.1 (Bondy [7]). *Let $G$ be a nonseparable graph and $T$ be an arbitrary tree 2-spanner of $G$. Then for every 2-cut $\{u, v\}$ of $G, uv \in E(T)$.*

*Proof.* Let $\{u, v\}$ be a 2-cut of $G$. Let $H_1$ be a connected component of $G - \{u, v\}$. Let $G_1 = G[V(H_1) \cup \{u, v\}]$ and $G_2 = G - V(H_1)$. Then each $G_i, i = 1, 2$, contains a $(u, v)$-path $P_i$ of length at least two. If $uv \notin E(T)$, then for each edge in $E(P_i) \setminus E(T)$, there is a path of length two in $T \cap G_i$ between its two ends. Thus there is a $(u, v)$-path $Q_i$ in $T \cap G_i$. $Q_1$ and $Q_2$ would then be a cycle in $T$, which is a contradiction. Hence $uv \in E(T)$. □

THEOREM 4.2. *Let $G$ be a nonseparable graph. Then a spanning tree $T$ of $G$ is a tree 2-spanner iff for each triconnected component $H$ of $G, T \cap H$ is a spanning star of $H$.*

*Proof.* If $T \cap H$ is a spanning star of $H$ for each triconnected component $H$ of $G$, then $T \cap H$ is a tree 2-spanner of $H$. Since each edge of $G$ belongs to some triconnected component, it follows that $T$ is a 2-spanner of $G$.

Conversely, suppose that $T$ is a tree 2-spanner of $G$. We first show that for each triconnected component $H$ of $G, T' = T \cap H$ is a tree 2-spanner of $H$. It is trivial if $H$ consists of a single edge. Thus we may assume that $|V(H)| \geq 3$. Then $H$ contains at least one edge $uv \notin E(T)$, since $H$ contains a cycle. Because $T$ is a tree 2-spanner, there exists a vertex $w$ such that $uw, vw \in E(T)$. If $w$ is not in $H$, then $u$ and $v$ are the only vertices in $H$ adjacent to $w$, since $H$ is a triconnected component and $|V(H)| \geq 3$. This implies that $\{u, v\}$ is a 2-cut. By Lemma 4.1, $uv$ would be an edge in $T$, which is a contradiction. Therefore, $w$ is in $H$ and thus $uw, vw$ are in $T'$. This implies that $T'$ is a tree 2-spanner of $H$.

It remains to be shown that $T'$ is a star. Suppose that $T'$ is not a star; then there is an edge $xy \in E(T')$ that is not incident to any leaf. Let $T'_x$ and $T'_y$ be the two connected components of $T' - xy$ containing vertices $x$ and $y$, respectively. Note that both $T'_x$ and $T'_y$ contain at least two vertices. For two arbitrary vertices $u \in V(T'_x) \setminus \{x\}$ and $v \in V(T'_y) \setminus \{y\}$, it is easy to see that $d_{T'}(u, v) \geq 3$. This implies $uv \notin E(H)$. Then $\{x, y\}$ would be a 2-cut of $H$, contrary to $H$ being a triconnected component. Therefore, $T'$ is a spanning star of $H$. □

COROLLARY 4.3. *A triconnected graph $G$ admits a tree 2-spanner iff it contains a universal vertex.*

Lemma 4.1 and Theorem 4.2 can be used to obtain a characterization of tree 2-spanner-admissible graphs in terms of decomposition. A graph $G$ is an *edge bonding* of two graphs $G_1$ and $G_2$ if $G = G_1 \cup G_2$ and $G_1 \cap G_2$ is an edge.

THEOREM 4.4 (Bondy [7]). *A graph $G$ is tree 2-spanner admissible iff each block $H$ of $G$ is either*

(1) *a triconnected graph with a universal vertex or*

(2) *an edge bonding (on edge $e$) of two tree 2-spanner-admissible graphs where $e$ is a tree edge in both graphs.*

*Proof.* Because of Observation 1.4, we only need to consider a block $H$ of $G$. If $H$ contains a universal vertex $u$, then the set of edges incident with $u$ induces a tree 2-spanner of $H$. If $H$ is an edge bonding of two tree 2-spanner-admissible graphs $H_1$ and $H_2$ on a tree edge $e$, then the edge bonding of a tree 2-spanner $T_1$ of $H_1$ and a tree 2-spanner $T_2$ of $H_2$ on edge $e$ yields a tree 2-spanner of $H$.

Conversely, suppose that $H$ is tree 2-spanner admissible. If $H$ has no 2-cut, then it is triconnected and by Corollary 4.3 contains a universal vertex. Otherwise, $H$ has a 2-cut $\{x, y\}$; by Lemma 4.1, then, $xy$ is an edge of $H$ and belongs to every tree 2-spanner of $H$. Let $H'$ be a connected component of $H - \{x, y\}, H_1 = H[V(H') \cup \{x, y\}]$, and $H_2 = H - V(H_1)$. It can then be deduced from Theorem 4.2 that $T \cap H_1$ and $T \cap H_2$ are tree 2-spanners of $H_1$ and $H_2$, respectively. Hence, $H$ is an edge bonding of two tree 2-spanner-admissible graphs $H_1$ and $H_2$ on edge $xy$. This completes the proof. □

We now use the above results to derive an algorithm for finding a tree 2-spanner $T$ (if it exists) in a graph $G$. The algorithm can be outlined as follows (details are left to the reader). First, find all blocks of $G$. Then, for each block, find all 2-cuts (if there is a 2-cut that does not induce a binding edge, then $G$ contains no tree 2-spanner, by Lemma 4.1) and triconnected components. Put all binding edges in $T$. Now, for each triconnected component $H$, find a spanning star containing all edges of $H$ that have been put into $T$ so far and put it in $T$ (if such a spanning star does not exist, then $G$ contains no tree 2-spanner by Lemma 4.1 and Theorem 4.2). Finally, if $T$ is a spanning tree, then it is a tree 2-spanner; otherwise, $G$ contains no tree 2-spanner. This algorithm can be implemented in linear time by using the triconnected component-finding algorithm of Hopcroft and Tarjan [22] and standard techniques.

THEOREM 4.5. *A tree 2-spanner (if it exists) of a graph can be found in $O(m + n)$ time.*

**4.2. The skeleton tree.** A tree spanner may be required to have some additional properties, such as a degree constraint, a bound on the diameter, or a limit on the number of leaves. Here we conduct a further investigation of the structure of tree 2-spanners to provide a useful tool in dealing with the construction of tree 2-spanners with additional properties. Henceforth, we assume that all graphs in this subsection are tree 2-spanner-admissible.

By Theorem 4.2, every tree 2-spanner of a triconnected graph is a spanning star. Thus there is nothing more to be said about the structure of tree 2-spanners in a triconnected graph. In light of Lemma 4.1, we can restrict our attention to nonseparable graphs with binding edges. We show that any tree 2-spanner of such a graph can be obtained from a "skeleton tree" of the graph by properly adding "compound leaves" to the skeleton tree. This result also gives another clear picture of the structure of tree 2-spanner-admissible graphs. In the rest of this subsection, we assume that $G$ is a tree 2-spanner-admissible graph that is nonseparable and contains at least one binding edge. We start with the subgraph of $G$ induced by the set of its binding edges.

LEMMA 4.6. *The set of binding edges of $G$ induces a tree.*

*Proof.* Let $B$ be the set of binding edges and $T_B = G[B]$. Clearly, $T_B$ is a forest, since $G$ admits a tree 2-spanner $T$ and every edge in $B$ belongs to $T$ by Lemma 4.1. We need to show only that $T_B$ is connected.

Let $\mathcal{H}$ be the set of triconnected components of $G$. Construct a bipartite graph $F$ with vertex set $B \cup \mathcal{H}$ in which $e \in B$ and $H \in \mathcal{H}$ are adjacent iff edge $e$ is in the triconnected component $H$. Since $G$ is connected, it is clear that $F$ is connected.

Let $v$ and $v'$ be two arbitrary vertices of $T_B$ and $e$ and $e'$ be two binding edges incident with $v$ and $v'$, respectively. Consider the bipartite graph $F$. Since $F$ is connected, there is an $(e, e')$-path $P = e_1 H_1 e_2 \ldots H_{k-1} e_k$ in $F$, where $e_i \in B, H_i \in \mathcal{H}, e_1 = e$, and $e_k = e'$. Thus for any two elements $e_i, e_{i+1} \in B, H_i$ is a triconnected component of $G$ that contains edges $e_i$ and $e_{i+1}$. By Theorem 4.2, $e_i$ and $e_{i+1}$ share a vertex. It is now easy to deduce that there is a $(v, v')$-path in $T_B$, since each $e_i$ is an edge in $T_B$. Therefore, $T_B$ is connected and the proof is complete. □

Let $\mathcal{T}(G)$ denote the set of tree 2-spanners of $G$ and $\mathcal{E}(G)$ denote the set of edges of $G$ contained in every tree 2-spanner of $G$, i.e., $\mathcal{E}(G) = \cap_{T \in \mathcal{T}(G)} E(T)$. Note that a *nontrivial tree* is a tree with at least one edge.

LEMMA 4.7. *$G[\mathcal{E}(G)]$ is a nontrivial tree.*

*Proof.* Obviously, $G[\mathcal{E}(G)]$ is a forest. Since $G$ is tree 2-spanner admissible, by Lemma 4.1, every binding edge of $G$ is contained in $G[\mathcal{E}(G)]$. Furthermore, for any edge $e \in \mathcal{E}(G)$, if $e$ is not a binding edge of $G$, then it belongs to a unique triconnected component $H$ of $G$. Since $H$ contains at least one binding edge of $G$, it follows from Theorem 4.2 that edge $e$ shares a vertex with at least one binding edge of $G$. By Lemma 4.6 and the assumption that $G$ contains a binding edge, we see that $G[\mathcal{E}(G)]$ is connected and hence a nontrivial tree. □

Because $G[\mathcal{E}(G)]$ induces a nontrivial tree that belongs to every tree 2-spanner of $G$, we call it the *skeleton tree* of $G$ and denote it by $\mathcal{S}(G)$. Let us now examine the structure of $G - V(\mathcal{S}(G))$. Recall that two disjoint subgraphs are fully joined iff every vertex in one subgraph is adjacent to every vertex in the other.

LEMMA 4.8. *Each connected component $C$ of $G - V(\mathcal{S}(G))$ is fully joined with a unique edge of $\mathcal{S}(G)$. Moreover, for any tree 2-spanner $T$ of $G$, each vertex in $C$ is a leaf of $T$.*

*Proof.* By Lemma 4.1, the skeleton tree $\mathcal{S}(G)$ contains the subgraph $\mathcal{S}$ induced by the set of binding edges of $G$. Therefore, each connected component $C$ of $G - V(\mathcal{S}(G))$ is a subgraph of a connected component $C'$ of $G - V(\mathcal{S})$. Since $C'$ belongs to a unique triconnected component $H$ of $G$, so does $C$. If $H$ contains more than one edge of $\mathcal{S}(G)$, say $e_1$ and $e_2$, then it follows from Theorem 4.2 that $e_1$ and $e_2$ must share a vertex $u$ and all the edges between $u$ and $C$ belong to $T$. This implies that every vertex of $C$ is in $\mathcal{S}(G)$, which contradicts the choice of $C$. Therefore, $H$ contains a unique edge $e$ of $\mathcal{S}(G)$. If one end of $e$ is not a universal vertex of $H$, then by Theorem 4.2, all the edges between the other end of $e$ and $C$ belong to $T$. Again, this contradicts the choice of $C$. Therefore, $C$ is fully joined with a unique edge $e$ of $\mathcal{S}(G)$. By Theorem 4.2, each vertex of $C$ is a leaf of $T$. □

Because of the above lemma, we call each connected component of $G - V(\mathcal{S}(G))$ a *compound leaf.* Then every edge $e$ of the skeleton tree $\mathcal{S}(G)$ has a set (possibly empty) of compound leaves fully joined with it. Let the two ends of $e$ be $x$ and $y$. Then for any compound leaf $C$ fully joined with $e, V(C) \cup \{x, y\}$ induces a triconnected component $H$ of $G$. Note that both $x$ and $y$ are adjacent to every vertex in $C$. The set of edges

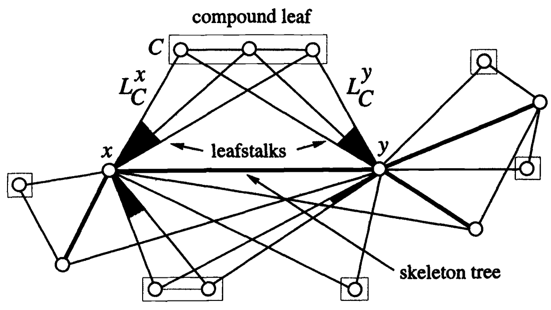


FIG. 6. *The skeleton tree, compound leaves, and leafstalks.*

between $x$ and $C$($y$ and $C$) forms a star $L_C^x(L_C^y)$ and will be referred to as a *leafstalk* of $C$. These concepts are illustrated in Fig. 6, where thick lines depict the skeleton tree, each box contains a compound leaf, and each shaded triangle indicates a nontrivial leafstalk (a leafstalk with more than one edge).

THEOREM 4.9 (skeleton tree theorem). *Let $G$ be a tree 2-spanner-admissible, nonseparable graph that contains binding edges. A spanning tree $T$ is a tree 2-spanner of $G$ iff it is obtained from the skeleton tree $\mathcal{S}(G)$ of $G$ by adding to $\mathcal{S}(G)$ exactly one leafstalk for each compound leaf of $G$.*

*Proof.* This follows from Theorem 4.2, Lemma 4.7, and Lemma 4.8. □

We now turn to the construction of the skeleton tree $\mathcal{S}(G)$ of $G$. First, we find all binding edges and triconnected components of $G$. As discussed in Lemma 4.6 and the proof of Lemma 4.7, these binding edges form a tree $S$ that is a subtree of $\mathcal{S}(G)$. Then we extend $S$ to $\mathcal{S}(G)$ by considering triconnected components one by one. For each triconnected component $H$, if $H$ contains two distinct edges $e_1$ and $e_2$ of $S$, then $e_1$ and $e_2$ share a vertex $u$ and we put into $\mathcal{S}(G)$ all the edges of $H$ that are incident with $u$ (by Theorem 4.2); if $H$ contains only one edge $e = uv$ of $S$ and one end of $e$, say $u$, is not a universal vertex of $H$, then we put into $\mathcal{S}(G)$ all the edges of $H$ that are incident with $v$ (by Theorem 4.2). The correctness of the above algorithm follows from our previous discussions.

By using the linear-time, triconnected component-finding algorithm of Hopcroft and Tarjan [22] and standard techniques, we can construct the skeleton tree and find all compound leaves in linear time. Therefore, the skeleton tree provides a handy and useful tool in constructing tree 2-spanners with certain properties. For instance, with the aid of the skeleton tree, the problem of finding a tree 2-spanner of bounded degree is easily solved in linear time. It is interesting to note that the corresponding degree-bounded spanning tree problem is NP-complete ([ND1] in [19]). Skeleton trees have also been used in the design of a polynomial-time algorithm for determining whether a 2-connected graph contains a tree 2-spanner isomorphic to a given tree [11], [13]. We note again that the corresponding isomorphic spanning tree problem is NP-complete ([ND8] in [19]). Further applications of skeleton trees can be found in the construction of quasi-tree 2-spanners (§5.2), as well as in the construction of nearly distance-preserving spanning trees [11].

**4.3. NP-completeness for $t \geq 4$.** Although a tree 2-spanner in a graph can be constructed in linear time, the problem of finding a tree $t$-spanner seems to be very hard for $t \geq 3$; in fact, as we show here, the problem is intractable for any fixed $t \geq 4$. As a consequence, the minimum $t$-spanner problem on unweighted graphs is

NP-complete for any fixed $t \geq 4$. The complexity of finding a tree 3-spanner in a graph is still unknown.

*Remark.* Stronger NP-completeness results hold for the minimum $t$-spanner problem on unweighted graphs. In fact, the problem is NP-complete for any fixed $t \geq 2$ [12], [26], even when restricted to graphs of bounded degree [15].

THEOREM 4.10. *For any fixed $t \geq 4$, the tree $t$-spanner problem is NP-complete.*

*Proof.* It is clear that the problem is in NP. To establish the NP-completeness, we present a polynomial transformation from 3SAT. By Observation 1.5, we need to consider only integral values of $t$. Let $t \geq 4$ be a fixed integer and $(U, C)$ be an arbitrary instance of 3SAT. We construct a graph $G$ such that $C$ is satisfiable iff $G$ has a tree $t$-spanner. Call a path with $t$ edges a *$t$-path.* The following result is useful in our construction.

LEMMA 4.11. *Let $G$ be a graph and $e$ an edge of $G$. Let $G'$ be a graph formed from $G$ by adding two distinct $t$-paths $P_1, P_2$ (all internal vertices of $P_1$ and $P_2$ are new vertices) between the two ends of $e$ and $T$ be a tree $t$-spanner of $G'$. Then $e \in E(T)$.*

*Proof.* We first notice that for any edge $e'$ of either $P_1$ or $P_2$, there is only one path in $G' - e'$ of length $\leq t$ between the two ends of $e'$. Furthermore, this unique path contains edge $e$. It follows that if $e$ is not in $T$, then all edges of $P_1$ and $P_2$ would have to be in $T$. However, $P_1$ and $P_2$ form a cycle, which contradicts $T$ being a tree. $\square$

From now on, by forcing an edge, we mean adding two distinct $t$-paths between the two ends of the edge. Such an edge will be called a *forced edge,* and the two $t$-paths will be called *forcing paths.* Denote $|U|$ by $n$ and $|C|$ by $m$. The graph $G$ is constructed as follows.

For each variable $u_i \in U, 1 \leq i \leq n$, construct a graph $H_i$ by

1. taking five vertices $x_i, u_i, \bar{u}_i, y_i$ and $z_i$,
2. adding edges $x_iy_i, x_iu_i, x_i\bar{u}_i, z_iu_i$, and $z_i\bar{u}_i$,
3. joining $y_i$ with $z_i$ by a $(t-2)$-path (all internal vertices on the path are new vertices) and forcing every edge on the path, and
4. joining $u_i$ with $\bar{u}_i$ by a $(t-3)$-path (all internal vertices on the path are also new vertices) and forcing every edge on the path as well.

Figure 7 shows the graph $H_i$ for $t = 4$. Next, put $H_1, \ldots, H_n$ together by identifying vertices $x_1, \ldots, x_n$ into a single vertex $x$ to form the variable setting component $H$. Vertices $u_i$ and $\bar{u}_i$ of $H_i$ will be used to represent the literals $u_i$ and $\bar{u}_i$, respectively, and they are called *literal vertices.*

For each clause $c_j \in C, 1 \leq j \leq m$, create a new vertex $c_j$, called a *clause vertex,* and add an edge between $c_j$ and each of its three distinct literal vertices in $H$. Note that each literal vertex is either a vertex $u_i$ or a vertex $\bar{u}_i$.

It is easy to see that $G$ can be constructed in polynomial time. It remains to be shown that $C$ is satisfiable iff $G$ has a tree $t$-spanner. Before describing the proof, we note the following important property of the graph $G$, which enables us to define a proper truth assignment for $C$ in terms of a tree $t$-spanner of $G$.

LEMMA 4.12. *Any tree $t$-spanner $T$ of $G$ contains exactly one of the two edges $xu_i$ and $x\bar{u}_i$ for each $1 \leq i \leq n$.*

*Proof.* Clearly, because $T$ contains the forced path between $u_i$ and $\bar{u}_i$, at most one of $xu_i$ and $x\bar{u}_i$ can be in $T$ (otherwise we would have a cycle in $T$). We need to show that $T$ contains at least one of $xu_i$ and $x\bar{u}_i$. Suppose that neither $xu_i$ nor $x\bar{u}_i$ is in $T$. Then the shortest path in $F = G - E(H_i)$ between $x$ and $u_i$ consists of at least three edges (edge $xl$ for some literal $l$ and edges $lc$ and $cu_i$ for some clause $c$ that

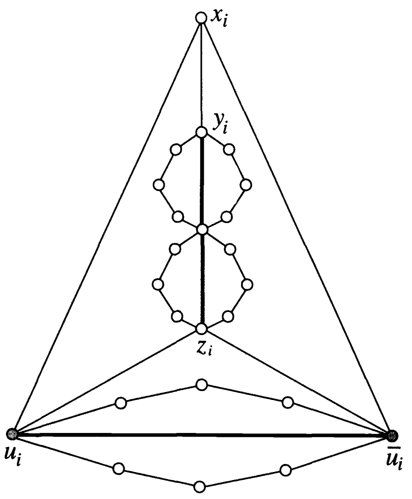


FIG. 7. *Component $H_i$ for $t = 4$.*

contains both $l$ and $u_i$). Thus $d_F(x, u_i) \geq 3$, and, similarly, $d_F(x, \bar{u}_i) \geq 3$. Consider two cases that depend on whether $xy_i$ is in $T$.

*Case* 1. $xy_i \in E(T)$. If neither $z_iu_i$ nor $z_i\bar{u}_i$ is in $T$, then no tree $(x, u_i)$-path is present inside $H_i$ and thus

$$d_T(x, u_i) \geq d_F(x, u_i) \geq 3.$$

But then

$$d_T(z_i, u_i) = d_T(z_i, x) + d_T(x, u_i) \geq (t-1) + d_F(x, u_i) \geq t + 2,$$

contrary to $T$ being a $t$-spanner. Therefore, at least one of $z_iu_i$ and $z_i\bar{u}_i$ is in $T$. Without loss of generality, we may assume that $z_iu_i \in E(T)$. Note that $z_i\bar{u}_i \notin E(T)$, as there is a forced $(u_i, \bar{u}_i)$-path in $T$. Since a tree path is unique between any two vertices, we have

$$d_T(x_i, \bar{u}_i) = d_T(x_i, z_i) + d_T(z_i, u_i) + d_T(u_i, \bar{u}_i) = 2t - 3 > t$$

for $t \geq 4$, again a contradiction to $T$ being a $t$-spanner.

*Case* 2. $xy_i \notin E(T)$. Then no tree path is present inside $H_i$ between $x$ and $u_i$ or $x$ and $\bar{u}_i$, and thus

$$d_T(x, u_i) \geq d_F(x, u_i) \geq 3$$

and

$$d_T(x, \bar{u}_i) \geq d_F(x, \bar{u}_i) \geq 3.$$

Then

$$d_T(x, z_i) = \min\{d_T(x, u_i) + d_T(u_i, z_i), d_T(x, \bar{u}_i) + d_T(\bar{u}_i, z_i)\} \geq 4.$$

It follows that

$$d_T(x, y_i) = d_T(x, z_i) + d_T(z_i, y_i) \geq t + 2,$$

contrary to $T$ being a $t$-spanner.

Since both cases lead to contradictions, we conclude that $T$ contains exactly one of two edges $xu_i$ and $x\bar{u}_i$ for each $1 \leq i \leq n$. $\square$

We now prove that $C$ is satisfiable iff $G$ has a tree $t$-spanner. Suppose that $C$ is satisfiable and let $\xi$ be a satisfying truth assignment for $C$. We construct a spanning tree $T$ of $G$ as follows:

1. for each forced edge $e$, put edge $e$ in $T$;

2. for each forcing path, arbitrarily delete one edge and then put the remaining edges in $T$;

3. for each variable $u_i, 1 \leq i \leq n$, if $\xi(u_i) = 1$, then put edges $xu_i$ and $z_iu_i$ in $T$; otherwise ($\xi(u_i) = 0$), put edges $x\bar{u}_i$ and $z_i\bar{u}_i$ in $T$;

4. for each clause $c_j, 1 \leq j \leq m$, arbitrarily pick a true literal $l_j$ in $c_j$ and put edge $c_jl_j$ in $T$.

Note that each clause vertex is a leaf in $T$. It is a routine matter to verify that $T$ is a tree $t$-spanner.

Conversely, suppose that $T$ is a tree $t$-spanner of $G$. We need to present a truth assignment $\xi_T$ that satisfies $C$. By Lemma 4.12, $T$ contains exactly one of two edges $xu_i$ and $x\bar{u}_i$ for each $1 \leq i \leq n$. Therefore, we can define a truth assignment $\xi_T$ by setting, for $1 \leq i \leq n, \xi_T(u_i) = 1$ whenever $xu_i \in E(T)$ and $\xi_T(u_i) = 0$ otherwise. It remains to be shown that $\xi_T$ satisfies $C$.

Suppose that some clause $c_j$ only contains false literals under $\xi_T$. Notice that any two literals are joined by a path in $T$ that avoids clause vertices. Therefore, $c_j$ is a leaf in $T$. Then for a clause edge $c_jl_j$ not in $T, d_T(c_j, l_j) = 2t - 3 \geq t + 1$ for $t \geq 4$, since the distance between any two false literals in $T$ is $2t - 4$, contrary to $T$ being a $t$-spanner. Therefore, each clause in $C$ contains at least one true literal under $\xi_T$ and thus $C$ is satisfiable. This completes the proof. $\square$

**5. Tree spanners in digraphs.** In this section, we consider tree spanners and quasi-tree spanners in digraphs. At first glance, it seems that tree spanner problems on digraphs are at least as hard as tree spanner problems on undirected graphs. Surprisingly, a tree $t$-spanner of a digraph can be found in almost-linear time. In fact, even a minimum tree spanner (i.e., a tree $t$-spanner with $t$ as small as possible) of a digraph can be found in almost-linear time. On the other hand, the situation for quasi-tree spanners in digraphs is closer to that for tree spanners in undirected graphs. We will use the results developed in §§3 and 4 for undirected graphs to obtain similar results for quasi-tree spanners.

Throughout this section, $G = (V, A; w)$ is a weighted digraph and $\tilde{G} = (V, E; \tilde{w})$ denotes the underlying undirected graph of $G$. Recall that for an arc $(x, y) \in A, \tilde{w}(xy) = w((x, y))$ if $(y, x) \notin A$ and $\tilde{w}(xy) = \min\{w((x, y)), w((y, x))\}$ if $(y, x) \in A$. Also, recall that an *in-neighbor* of a vertex $x$ in $G$ is a vertex $y$ such that $(y, x) \in A$ and an *out-neighbor* of $x$ is a vertex $z$ such that $(x, z) \in A$. A vertex $v$ is a *source* if it has no in-neighbors and an *intermediate vertex* if it has both in- and out-neighbors. A spanning subgraph $T$ of $G$ is a spanning tree if $T$ contains no directed cycle and $\tilde{T}$ is a tree. For convenience, we say that $G$ is connected (triconnected) whenever its underlying graph $\tilde{G}$ is connected (triconnected). The meanings of blocks and connected components of $G$ should be understood in the same manner.

**5.1. Finding a minimum tree spanner.** Recall that in a digraph $G$, a vertex $x$ reaches vertex $y$ (i.e., $y$ is reachable from $x$) if there is a directed $(x, y)$-path in $G$. By the definition of a tree spanner, it is easy to see that a spanning tree $T$ of $G$ is a tree spanner iff it preserves reachability of $G$, i.e., $x$ reaches $y$ in $G$ iff $x$ reaches $y$ in $T$. Unlike an undirected graph, then, a spanning tree of $G$ is not necessarily a tree spanner. In fact, we have the following result:

LEMMA 5.1. *A digraph $G$ contains at most one tree spanner.*

*Proof.* Let $S$ and $T$ be two arbitrary tree spanners of $G$. If $S \neq T$, then there is an arc $(x, y)$ in $S$ that is not in $T$. Thus there is a directed $(x, y)$-path $P$ in $T$, since $T$ is a tree spanner. Let $z$ be an internal vertex of $P$. Then there is a directed $(x, z)$-path $Q$ and a directed $(z, y)$-path $Q'$ in $S$, since $S$ is a tree spanner. This implies that there are two distinct, directed $(x, y)$-paths $QQ'$ and $xy$ in $S$, which contradicts $S$ being a tree. Therefore, $S = T$ and $G$ contains at most one tree spanner. $\square$

Because of the above lemma, we need to consider only the problem of finding the tree spanner $T$ in a digraph $G$, since $T$ is automatically a minimum tree spanner and we can use it to solve the tree $t$-spanner problem by comparing $t$ with the stretch index of $T$. Recall that an *acyclic digraph* is a digraph that contains no directed cycle.

LEMMA 5.2. *If a digraph $G$ admits a tree spanner, then $G$ is acyclic.*

*Proof.* Let $T$ be a tree spanner of $G$. Suppose that $G$ contains a directed cycle $C$. Any two vertices of $C$ are then mutually reachable in $T$ since $T$ is a spanner of $G$, which contradicts $T$ being a tree. Hence $G$ is acyclic. $\square$

In light of the above lemma, we will hereafter assume that $G$ is acyclic. $G$ then contains a source $s$. We now present a necessary and sufficient condition for $G$ to admit a tree spanner in terms of $G - s$. Note that $N_G^+(s)$ is the set of out-neighbors of $s$ in $G$ and that a *trivial digraph* (a digraph with a single vertex) is itself a tree spanner.

THEOREM 5.3. *Let $G$ be an acyclic digraph and $s$ be a source of $G$. $G$ then admits a tree spanner iff each connected component $H_i$ of $G - s$ contains a tree spanner $T_i$ such that there is a vertex $v_i \in V(H_i) \cap N_G^+(s)$ that reaches every vertex of $V(H_i) \cap N_G^+(s)$ through arcs of $T_i$.*

*Proof.* If the condition of the theorem is satisfied, then it is readily checked that $T_1 \cup \cdots \cup T_k + \{sv_1, \ldots, sv_k\}$, where $k$ is the number of connected components of $G - s$, is the tree spanner of $G$, since for $i \neq j$ there are no arcs between $H_i$ and $H_j$.

Conversely, suppose that $G$ contains a tree spanner $T$. Then $s$ is a source in $T$. We first show that each connected component $H_i$ contains a unique vertex $v_i$ adjacent to $s$ in $T$. Since $G$ is connected, $H_i$ by definition contains at least one such vertex. Suppose that $H_i$ contains two such vertices $u$ and $u'$. Then there is a $(u, u')$-path $P$ in $\tilde{H}_i$. Each edge in $P$ corresponds to a unique arc in $H_i$, as $G$ is acyclic; for each such arc $(x, y)$, there is a directed $(x, y)$-path $P_{xy}$ in $T$, since $T$ is a tree spanner. Let $T_i = T \cap H_i$. It is easy to see that $P_{xy}$ lies entirely in $T_i$. It follows that there is a $(u, u')$-path $P'$ in $\tilde{T}_i$. However, $P'$ together with $su$ and $su'$ forms a cycle in $\tilde{T}$, a contradiction. Therefore, $s$ is adjacent to a unique vertex $v_i$ of $H_i$ in tree $T$. Clearly, $v_i \in V(H_i) \cap N_G^+(s)$.

Now, by the definition of $H_i$, we easily see that $T_i$ is connected; $T_i$ is thus a tree spanner of $H_i$. Furthermore, for each out-neighbor $v$ of $s$ in $H_i$, there is a directed $(s, v)$-path $Q$ in $T$, since $T$ is a tree spanner. Then the $(v_i, v)$-section of $Q$ is a directed $(v_i, v)$-path in $T_i$, and hence $v$ is reachable from $v_i$ in $T_i$. $\square$

It is trivial to transform the above theorem into a recursive procedure for finding the tree spanner of $G$. In order to implement the procedure efficiently, we present an iterative version. Let $\{1, \ldots, n\}$ be the vertex set of $G$. Since $G$ is acyclic, we can assume that the vertices of $G$ have been topologically ordered, i.e., if $(i, j)$ is an arc of $G$, then $i < j$. Let $G_i$ denote the subgraph of $G$ induced by vertices $\{i, \ldots, n\}$. Vertex $i$ is then a source of $G_i$ by the definition of the topological ordering of $G$. Therefore, vertex $v_i$ in Theorem 5.3 is the vertex with smallest number in $V(H_i) \cap N_G^+(s)$. Note that the out-neighbors of vertex $i$ in $G_i$ are the same as those of $i$ in $G$ and hence will be denoted by $N^+(i)$. Figure 8 depicts an acyclic digraph, its tree spanner, and a

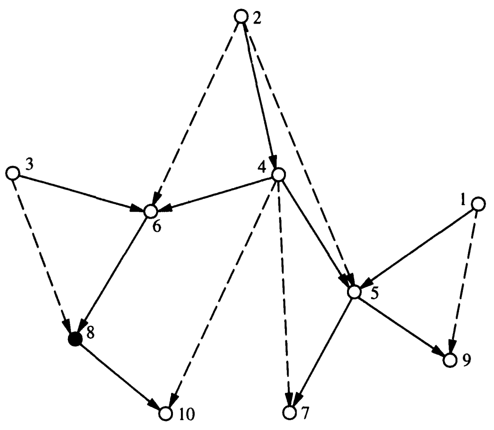


FIG. 8. *An acyclic digraph $G$, its tree spanner, and a topological ordering of $V$.*

topological ordering of its vertices. The following procedure finds the tree spanner of $G$ when it contains one.

```
PROCEDURE TREESPANNER($G, T$); {Find the tree spanner $T$ of an acyclic digraph $G$.}
begin
1.  $T \leftarrow$ the trivial tree consisting of vertex $n$;
2.  for $i \leftarrow n - 1$ downto 1 do
3.      $\mathcal{H}_i \leftarrow \{H | (H$ is a connected component of $G_{i+1}) \wedge (V(H) \cap N^+(i) \neq \emptyset)\}$;
4.      for each $H \in \mathcal{H}_i$ do
5.          $v_{i,H} \leftarrow \min\{j | j \in v(H) \cap N^+(i)\}$;
6.          if $v_{i,H}$ reaches every vertex in $V(H) \cap N^+(i)$ through arcs of $T$
               then $T \leftarrow T + (i, v_{i,H})$ else output "No" EXIT;
        end for;
    end for;
7.  return $T$;
end TREESPANNER.
```

Notice that $T$ is a forest during the computation of the above procedure. Let $T_i$ denote the forest $T$ after the normal completion of the $(n - i)$th iteration of the "for" loop at line (2). By Theorem 5.3, it is clear that $T_i$ consists of tree spanners of the connected components of $G_i$; thus $T_1$ is the tree spanner of $G$. However, a straightforward implementation of the procedure may take $O(mn)$ time.

We now refine the procedure to obtain a more efficient algorithm. We notice the following: first, the check at line (6) can be postponed after the completion of the "for" loop at line (2), since $v_{i,H}$ reaches all vertices in $V(H) \cap N^+(i)$ through arcs of $T_i$ iff it reaches these vertices through arcs of $T_1$; second, the computation at lines (3) and (5) requires only the vertex sets of connected components of $G_{i+1}$; and third, the connected components of $G_i$ can be obtained from those of $G_{i+1}$ by merging vertex $i$ and all connected components in $\mathcal{H}_i$ into a single component.

Based on the above observations, we use sets to maintain the connected components of $G_i$. Initially, we have $n$ sets consisting of $n$ single vertices. These sets will be merged to represent the connected components of $G_i$ during the process. Thus line (3) can be carried out by finding all sets containing the out-neighbors of vertex $i$. We are now ready to present an algorithm that finds the tree spanner of $G$. The algorithm first decides if $G$ is acyclic, then finds a spanning tree $T$ of $G$, and finally verifies if $T$ is the tree spanner of $G$. It also computes the stretch index $t$ of $T$ when $T$ is a tree spanner.

ALGORITHM TREE-SPANNER$(G, T, t)$\{Find the tree spanner $T$ of $G$.\}
**Input:** A weighted digraph $G = (V, A; w)$;
**Output:** The tree spanner $T$ and its stretch index $t$ if $G$ admits a tree spanner; otherwise output "No".

```
begin
1.  if $G$ is not acyclic then output "No" EXIT
      else compute a topological ordering of $G$;
    {Vertices $1, \ldots, n$ is a topological ordering of $G$.}
2.  $T \leftarrow$ the trivial tree consisting of vertex $n$;
3.  Create set $\{i\}$ for each vertex $i$ of $G$;
4.  for $i \leftarrow n-1$ downto 1 do
5.      $\mathcal{H}_i \leftarrow \emptyset$;
6.      for each out-neighbor $k$ of $i$ do
        {Compute $\{H | (H$ is a connected component of $G_{i+1}) \wedge (V(H) \cap N^+(i) \neq \emptyset)\}$.}
7.          Find the set $H_k$ containing vertex $k$;
8.          $\mathcal{H}_i \leftarrow \mathcal{H}_i \cup \{H_k\}$;
        end for;
9.      for each set $H \in \mathcal{H}_i$ do
10.         $v_{i,H} \leftarrow \min\{j | j$ is an out-neighbor of $i$ in $H\}$;
11.         $T \leftarrow T + (i, v_{i,H})$;
        end for;
12.     Merge $\{i\}$ and all $H \in \mathcal{H}_i$ into a single set;
    end for;
13. if $T$ is a tree spanner
      then compute the stretch index $t$ of $T$
      else output "No";
end TREE-SPANNER.
```

We now consider the complexity of the above algorithm. Line (1) takes $O(m+n)$ (cf. [1], [30]). Vertex $v_{i,H}$ at line (10) can be found in $O(|N^+(i)|)$ time by keeping track of the set $H_k$ for each out-neighbor $k$ of $i$ at line (7). By Theorem 2.1(b), line (13) can be carried out in linear time. The merge operation at line (12) and the find operation at line (7) constitute a sequence of union-and-find operations on disjoint sets; there are at most $m + n$ operations in total. By using the well-known "path compression on balanced trees" technique for disjoint set manipulation, these $\leq m + n$ union-and-find operations can be implemented in $O((m+n)\alpha(m+n, n))$ time (cf. [1], [30]), where $\alpha$ is a functional inverse of Ackermann's function and, for all feasible large $m$ and $n, \alpha(m, n) \leq 4$ [30].

The remaining computation takes linear time. The overall running time of the algorithm is thus $O((m + n)\alpha(m + n, n))$. Therefore, we can state the following theorem.

THEOREM 5.4. *The minimum tree spanner of a weighted digraph and its stretch index can be computed in $O((m+n)\alpha(m+n, n))$ time.*

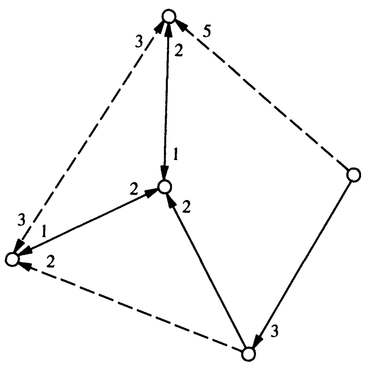


FIG. 9. *A quasi-tree* 1.5*-spanner.*

**5.2. Quasi-tree spanners in digraphs.** Recall that a quasi-tree of $G$ is a spanning subgraph $T$ such that $\tilde{T}$ is a tree; recall also that $T$ is a quasi-tree $t$-spanner if it is a $t$-spanner of $G$. See Fig. 9 for an example of a quasi-tree spanner. The notion of quasi-tree spanners is intended to capture the underlying tree structure of the spanner. As we will see, results on quasi-tree spanners in digraphs are quite similar to those of tree spanners in undirected graphs. We begin by considering relationships between quasi-tree spanners in $G$ and tree spanners in $\tilde{G}$.

LEMMA 5.5. *Let $T$ be a quasi-tree $t$-spanner of $G$. If both arcs $(x,y)$ and $(y,x)$ belong to $G$, then $(x,y) \in A(T)$ iff $(y,x) \in A(T)$.*

*Proof.* It suffices to show that $(x,y) \in A(T)$ implies $(y,x) \in A(T)$. Suppose $(y,x) \notin A(T)$. Then there is a directed $(y,x)$-path $P$ in $T$. It follows that the corresponding edges of $P$ in $\tilde{T}$ together with edge $xy$ form a cycle in $\tilde{T}$, which contradicts $\tilde{T}$ being a tree. □

LEMMA 5.6. *If $T$ is a quasi-tree $t$-spanner of $G$, then $\tilde{T}$ is a tree $t$-spanner of $\tilde{G}$.*

*Proof.* Let $xy$ be an arbitrary edge in $\tilde{G}-\tilde{T}$. We need to show $d_{\tilde{T}}(x,y) \leq t\cdot\tilde{w}(xy)$. By the definition of $\tilde{T}$, it is easy to see that $d_{\tilde{T}}(u,v) \leq d_T(u,v)$ for any two vertices $u,v \in V$. If exactly one of arcs $(x,y)$ and $(y,x)$, say $(x,y)$, is in $G$, then $(x,y)$ is in $G-T$, and thus

$$d_{\tilde{T}}(x,y) \leq d_T(x,y) \leq t\cdot w((x,y)) = t\cdot\tilde{w}(xy),$$

since $\tilde{w}(xy) = w((x,y))$. Otherwise, both $(x,y)$ and $(y,x)$ are arcs in $G$ and thus are in $G-T$ by Lemma 5.5. Then $d_T(x,y) \leq t\cdot w((x,y))$ and $d_T(y,x) \leq t\cdot w((y,x))$. Therefore,

$$d_{\tilde{T}}(x,y) \leq \min\{d_T(x,y), d_T(y,x)\} \leq t\cdot\min\{w((x,y)), w((y,x))\} = t\cdot\tilde{w}(xy).$$

This proves the lemma. □

In light of the above two results, we can use the results on tree spanners in §§3 and 4 to obtain similar results for quasi-tree spanners. Given a weighted undirected graph $F$, we construct a weighted digraph $D$ by replacing each edge $xy$ of $F$ with two arcs $(x,y)$ and $(y,x)$ and setting $w'((x,y)) = w'((y,x)) = w(xy)$, where $w$ and $w'$ are the weighting functions of $F$ and $D$, respectively. The following two NP-completeness results can be readily obtained from Theorem 3.9 in §3.2 and Theorem 4.10 in §4.3, respectively, by using Lemmas 5.5 and 5.6.

THEOREM 5.7. *For any fixed $t > 1$, it is NP-complete to determine whether a weighted digraph contains a quasi-tree $t$-spanner, even if all arcs have integral weights.*

THEOREM 5.8. *For any fixed $t \geq 4$, it is NP-complete to determine whether an unweighted digraph contains a quasi-tree $t$-spanner.*

On the other hand, the results in §§3.1 and 4.1 can be extended to quasi-tree 1-spanners in weighted digraphs and quasi-tree 2-spanners in unweighted digraphs, respectively.

We first discuss the weighted case. Suppose that $G$ admits a quasi-tree 1-spanner $T$. Then by Lemma 5.6, $\tilde{T}$ is a tree 1-spanner of $\tilde{G}$. Therefore, $\tilde{T}$ is the unique minimum spanning tree of $\tilde{G}$ by Theorem 3.7 of §3.1. By Lemma 5.5, $T$ is uniquely determined by $\tilde{T}$. Therefore, it is easy to see that the following algorithm finds a quasi-tree 1-spanner in a weighted digraph. First find a minimum spanning tree $\tilde{T}$ of $\tilde{G}$; then construct the maximum quasi-tree $T$ corresponding to $\tilde{T}$ by putting into $T$, for every edge $xy$ of $\tilde{T}$, all arcs between vertices $x$ and $y$ in $G$; and finally verify if $T$ is a 1-spanner. Since $\tilde{G}$ can be obtained from $G$ in linear time, $\tilde{T}$ can be found in $O(m \log \beta(m,n))$ time, $T$ can be constructed from $\tilde{T}$ in linear time, and verification takes linear time by Theorem 2.1(c), we have the following result.

THEOREM 5.9. *The quasi-tree 1-spanner of a weighted digraph can be found in $O(m \log \beta(m,n))$ time.*

We now turn our attention to finding a quasi-tree 2-spanner in an unweighted digraph $G$. We first consider triconnected digraphs. Remember that by a triconnected digraph $G$, we mean that $\tilde{G}$ is triconnected. Also, bear in mind that a vertex $u$ will be referred to as a universal vertex of $G$ whenever it is a universal vertex of $\tilde{G}$. Finally, recall that an intermediate vertex is any vertex with both in- and out-neighbors.

THEOREM 5.10. *A triconnected digraph $G$ admits a quasi-tree 2-spanner iff it contains a universal vertex $u$ such that, for any intermediate vertex $v$ of $G - u$, both $(u,v)$ and $(v,u)$ are arcs of $G$.*

*Proof.* If $G$ contains such a universal vertex $u$, then it is readily checked that the set of arcs between $u$ and the remaining vertices of $G$ induces a quasi-tree 2-spanner of $G$. Conversely, suppose that $G$ admits a quasi-tree 2-spanner $T$. It follows from Lemma 5.6 and Theorem 4.2 that $\tilde{T}$ is a spanning star of $\tilde{G}$ centred at a vertex, say $u$. Therefore, $u$ is a universal vertex of $\tilde{G}$ and hence of $G$. Let $v$ be an arbitrary intermediate vertex of $G$. If $(u,v)$ is an arc of $G$, then there must be a vertex $x$ such that $(v,x)$ is an arc of $G$, since $v$ is an intermediate vertex. If $x \neq u$, then $(v,x)$ is an arc of $G - T$. Then there is a directed $(v,x)$-path $P$ of length 2 in $T$, since $T$ is a quasi-tree 2-spanner of $G$. Notice that $\tilde{T}$ is a spanning star. Thus $P$ passes through vertex $u$, which implies that $(v,u)$ is an arc of $G$. By a similar argument, we can deduce that $(u,v)$ is an arc of $G$ if $(v,u)$ is an arc of $G$. □

To illustrate the above theorem, a triconnected digraph and its quasi-tree 2-spanner are depicted in Fig. 10. It is clear that we can use the above theorem to find a quasi-tree 2-spanner in a triconnected digraph. We need only to find all intermediate vertices $I$ and all universal vertices $U$ of $G$ and then check if there is a vertex $u \in U$ such that all possible arcs between $u$ and $I$ appear in $G$. If such a vertex $u$ exists, then the set of arcs between $u$ and the remaining vertices of $G$ forms a quasi-tree 2-spanner; otherwise, $G$ has no quasi-tree 2-spanner. It is easy to see that this method takes linear time.

It is possible to extend the structural results in §4.1 to quasi-tree 2-spanners and then obtain an algorithm for constructing quasi-tree 2-spanners. However, an easy way to construct a quasi-tree 2-spanner is to use the skeleton tree. Because of Theorem 5.10, we need to consider only a nonseparable digraph $G$ that is not triconnected. As we have mentioned, if $G$ contains a quasi-tree, then $\tilde{G}$ is tree 2-spanner admissible. Then $\tilde{G}$ contains a skeleton tree $\tilde{S} = \mathcal{S}(\tilde{G})$, which corresponds to a subgraph $S$ of $G$.

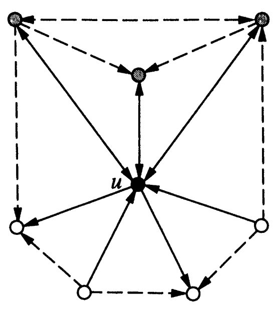


FIG. 10. *A quasi-tree 2-spanner in a triconnected digraph.*

TABLE 2
*The complexity status of tree spanner problems.*

| $t$ | Weighted graphs | Unweighted graphs | Directed graphs |
|---|---|---|---|
| 1 | $O(m \log \beta(m,n))$ | $O(m+n)$ | $O((m+n)\alpha(m+n,n))$ |
| (1, 3) | NPc | $O(m+n)$ | $O((m+n)\alpha(m+n,n))$ |
| [3, 4) | NPc | ? | $O((m+n)\alpha(m+n,n))$ |
| $[4,\infty)$ | NPc | NPc | $O((m+n)\alpha(m+n,n))$ |
| $\infty$ | $O(m \log \beta(m,n))$ | $O(m+n)$ | $O((m+n)\alpha(m+n,n))$ |

Since $\tilde{S}$ belongs to every tree 2-spanner of $\tilde{G}$, by Lemma 5.6, any quasi-tree 2-spanner of $G$ must contain $S$. Therefore, $S$ must be a 2-spanner of the subgraph of $G$ induced by vertices in $S$. For a compound leaf $C$ of $\tilde{G}$, let $e_C$ be the unique edge in $\tilde{S}$ with which $C$ is fully joined and $\tilde{H}_C$ be the triconnected component of $\tilde{G}$ containing $C$. Let $\tilde{L}_C^1$ and $\tilde{L}_C^2$ denote the two leafstalks in $\tilde{H}_C$ with the addition of edge $e_C$. Denote the subgraphs in $G$ corresponding to $\tilde{H}_C, \tilde{L}_C^1$, and $\tilde{L}_C^2$ by $H_C, L_C^1$, and $L_C^2$, respectively. In light of the skeleton tree theorem (Theorem 4.9 in §4.2), for each compound leaf $C$ we need to check only if either $L_C^1$ or $L_C^2$ is a 2-spanner of $H_C$.

To summarize, we outline the quasi-tree 2-spanner algorithm as follows. First, decide if $\tilde{G}$ is tree 2-spanner admissible. If it is, then find all blocks of $G$. For each block $B$ of $G$, if it is triconnected, then use Theorem 5.10 to find its quasi-tree 2-spanner; otherwise, construct the skeleton tree $\tilde{S}_B$ of $\tilde{B}$ and the corresponding subgraph $S_B$ in $B$. Check if $S_B$ is a 2-spanner of $B[V(S_B)]$. Finally, for each compound leaf $C$ of $B$, check if either $L_C^1$ or $L_C^2$ is a 2-spanner of $H_C$. If $G$ passes all of the above checks, then it contains a quasi-tree 2-spanner; otherwise, it does not. The actual quasi-tree 2-spanner of $G$ can be obtained by keeping track of $S_B, L_C^1$, and $L_C^2$.

The correctness of this algorithm follows from our discussions. We now estimate the complexity of the algorithm. It has been shown in §§4.1 and 4.2 that whether $\tilde{G}$ is tree 2-spanner admissible can be decided in linear time and that the skeleton tree of $\tilde{B}$ can be found in linear time. We also mentioned that it takes linear time to find a quasi-tree 2-spanner in $B$ if $B$ is triconnected. Furthermore, checking if $S_B$ is a 2-spanner of $B[V(S_B)]$ takes linear time by Theorem 2.1(c). Since all of the remaining operations can be carried out in linear time as well, the algorithm takes linear time.

THEOREM 5.11. *A quasi-tree 2-spanner in an unweighted digraph can be found in linear time.*

**6. Concluding remarks.** In this paper, we introduced the notion of tree spanners and studied the theoretical and algorithmic aspects of the subject. In particular, we considered the complexity of tree spanner problems for weighted, unweighted, and directed graphs. The current complexity status of tree spanner problems is summarized in Table 2, where row "$\infty$" indicates the complexity of finding a tree spanner (with minimum weight if $G$ is weighted). The complexity of quasi-tree spanner problems on weighted and unweighted digraphs is the same as that of tree spanner problems on weighted and unweighted graphs, respectively.

Note that the tree 3-spanner problem on unweighted graphs and the quasi-tree 3-spanner problem on unweighted digraphs remain open. We conjecture that the tree 3-spanner problem on unweighted digraphs is NP-complete; if true this would imply the NP-completeness of the quasi-tree 3-spanner problem on unweighted digraphs.

One can also consider the tree $t$-spanner problem for restricted families of graphs. For partial $k$-trees, it is easily deduced from the results of Arnborg et al. [4] that the problem is polynomial-time solvable for any fixed $t$, since it is a monadic second-order problem. However, the problem is open for planar graphs, bounded degree graphs, and many other interesting families of graphs.

In terms of applications, it is desirable to construct tree spanners with small stretch factors. Is there a polynomial-time algorithm for finding a tree $t$-spanner such that $t$ is close to the stretch factor of the minimum tree spanner? The notion of tree spanners can also be extended to other families of graphs. In general, given a family $\mathcal{F}$ of graphs, one can ask whether a graph $G$ contains a spanner $H \in \mathcal{F}$. The problem is particularly interesting for families of graphs that underlie communication network structures or parallel machine architectures, since graphs that contain these graphs as spanners capture some important properties of jobs that can be carried out on these networks or machines. However, we expect that the problem is hard for most families of graphs.

**Acknowledgments.** Some material in this paper has previously appeared in the first author's Ph.D. dissertation under the supervision of the second author. The authors thank the referees for useful suggestions.

## REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison–Wesley, Reading, MA, 1974.

[2] I. Althöfer, *On optimal realization of finite metric spaces by graphs*, Discrete Comput. Geom., 3 (1988), pp. 103–122.

[3] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, *On sparse spanners of weighted graphs*, Discrete Comput. Geom., 9 (1993), pp. 81–100.

[4] S. Arnborg, J. Lagergren, and D. Seese, *Easy problems for tree-decomposable graphs*, J. Algorithms, 12 (1991), pp. 308–340.

[5] B. Awerbuch, A. Baratz, and D. Peleg, *Efficient broadcast and light-weight spanners*, manuscript, 1992.

[6] S. Bhatt, F. Chung, F. Leighton, and A. Rosenberg, *Optimal simulations of tree machines*, in 27th IEEE Foundations of Computer Science, Toronto, 1986, pp. 274–282.

[7] J. A. Bondy, *Trigraphs*, Discrete Math., 75 (1989), pp. 69–79.

[8] J. A. Bondy and G. Fan, *Cycles in weighted graphs*, Combinatorica, 11 (1991), pp. 191–205.

[9] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, North–Holland, New York, 1976.

[10] L. Cai, *Spanning 2-trees*, manuscript, 1994.

[11] ———, *Tree Spanners: Spanning Trees that Approximate Distances*, Ph.D. thesis, University of Toronto, Toronto, Canada, 1992; Technical Report 260/92, Department of Computer

Science, University of Toronto, 1992.

[12] ———, *NP-completeness of minimum spanner problems*, Discrete Appl. Math., 48 (1994), pp. 187–194.

[13] L. CAI AND D. G. CORNEIL, *Isomorphic tree spanner problems*, Algorithmica, to appear.

[14] L. CAI AND J. M. KEIL, *Computing visibility information in an inaccurate simple polygon*, Internat. J. Comput. Geom. Appl., submitted.

[15] ———, *Spanners in graphs of bounded degree*, Networks, 24 (1994), pp. 233–249.

[16] L. P. CHEW, *There is a planar graph almost as good as the complete graph*, in Proc. 2nd ACM Symposium on Computational Geometry, Yorktown Heights, NY, 1986, pp. 169–177.

[17] M. L. FREDMAN AND R. E. TARJAN, *Fibonacci heaps and their uses in improved network optimization algorithms*, J. Assoc. Comput. Mach., 34 (1987), pp. 596–615.

[18] H. N. GABOW, Z. GALIL, T. H. SPENCER, AND R. E. TARJAN, *Efficient algorithms for finding minimum spanning trees in undirected and directed graphs*, Combinatorica, 6 (1986), pp. 109–122.

[19] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman, San Francisco, 1979.

[20] S. L. HAKIMI AND S. S. YAU, *Distance matrix of a graph and its realizability*, Quart. Appl. Math., 22 (1964), pp. 305–317.

[21] D. HAREL AND R. E. TARJAN, *Fast algorithms for finding nearest common ancestors*, SIAM J. Comput., 13 (1984), pp. 338–355.

[22] J. HOPCROFT AND R. E. TARJAN, *Dividing a graph into triconnected components*, SIAM J. Comput., 2 (1973), pp. 135–158.

[23] A. L. LIESTMAN AND T. SHERMER, *Additive graph spanners*, Networks, 23 (1993), pp. 343–364.

[24] ———, *Additive spanners for hypercubes*, Parallel Process. Lett., 1 (1992), pp. 35–42.

[25] ———, *Grid spanners*, Networks, 23 (1993), pp. 123–133.

[26] D. PELEG AND A. A. SCHÄFFER, *Graph spanners*, J. Graph Theory, 13 (1989), pp. 99–116.

[27] D. PELEG AND J. D. ULLMAN, *An optimal synchronizer for the hypercube*, in Proc. 6th ACM Symposium on Principles of Distributed Computing, Vancouver, 1987, pp. 77–85.

[28] D. PELEG AND E. UPFAL, *A tradeoff between space and efficiency for routing tables*, in Proc. 20th ACM Symposium on Theory of Computing, Chicago, 1988, pp. 43–52.

[29] D. RICHARDS AND A. L. LIESTMAN, *Degree-constrained pyramid spanners*. Parallel Distrib. Comput., to appear.

[30] R. E. TARJAN, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, 1983.

[31] P. WINKLER, *The complexity of metric realization*, SIAM J. Discrete Math., 1 (1988), pp. 552–559.

SIAM J. DISC. MATH.
Vol. 8, No. 3, pp. 388–400, August 1995

# COMBINATORIAL ALGORITHM FOR A LOWER BOUND ON FRAME RIGIDITY *

D. S. FRANZBLAU†

**Abstract.** A classic problem is that of computing the rigidity of a network of rigid bars or, more formally, computing the degrees of freedom of a *frame,* a graph with a generic straight-line embedding in Euclidean space. More recently, in trying to explain the effect of the internal structure of glassy materials on their rigidity, researchers have investigated the relationship between the structure of network models and the number of degrees of freedom. It would thus be valuable to have an efficient, purely combinatorial algorithm to compute the degrees of freedom of an arbitrary frame. For frames in two dimensions, there are several such algorithms, all based on a theorem of Laman. However, despite considerable effort, no one has yet found such an algorithm for frames in three or more dimensions. Here, a new combinatorial approach, similar to ear decomposition, is introduced and shown to give a practical algorithm for computing a nontrivial lower bound on the number of degrees of freedom in three dimensions. Results of computational studies on an important class of network models of glasses are given, suggesting that the resulting bound is close to the exact answer in the cases of interest. The algorithm has been implemented in $O(n)$ space and $O(n^2)$ time for bounded-degree networks with $n$ vertices and has already provided new results of interest on these networks.

**Key words.** rigidity, degrees of freedom, frame, framework, network, graph, algorithm, ear decomposition, amorphous solid, glass, zero-frequency mode, dynamical matrix

**AMS subject classifications.** primary, 70J10, 68R10; secondary, 70J05, 05C85, 05C38, 90C27

## 1. Introduction.

The problem of determining the rigidity of a network of rigid bars connected by joints has been studied for over a century [Ma1864]. Formally, the problem is to compute the number of degrees of freedom of a frame, which is a graph embedded in Euclidean space with distance constraints imposed by the edges. The computation of the number of degrees of freedom also has interesting applications to the study of glasses that are modeled as networks of atoms and bonds [Ph82], [Th83]. If one is given a specific embedding of the graph (i.e., its *geometry*), one can solve the problem by computing the rank of the matrix representing the constraints, using ideas dating back at least to Lagrange [Lg1788]. In studying models of glasses, however, one would like to know the effect of the network *topology* on the degrees of freedom (also called zero-frequency modes). In this case, the problem of interest is to compute the number of degrees of freedom for graphs embedded *generically* [LY82] (meaning essentially that the vertices do not lie on an algebraic surface), a number that is independent of the embedding. The standard approach in studying models of glasses is to choose a particular embedding and compute the rank of the appropriate matrix. However, standard matrix methods, such as Gaussian elimination, use $\Theta(n^2)$ storage and $\Theta(n^3)$ time for a system of $n$ vertices, making this approach practical only for networks of a few hundred vertices. Thus, it would be valuable to have an efficient algorithm that is purely combinatorial (requires only the graph as input).

* Received by the editors July 22, 1991; accepted for publication (in revised form) May 5, 1994. Part of this research was performed under a joint study agreement at the IBM T. J. Watson Research Center.

† Department of Mathematics, Vassar College, Poughkeepsie, New York 12601. Present address: Center for Discrete Mathematics and Theoretical Computer Sciences (DIMACS), Rutgers University, P.O. Box 1179, Piscataway, New Jersey 08855.

On the other hand, finding a purely combinatorial algorithm for computing degrees of freedom in three dimensions is a key open problem in mathematical rigidity theory. The existing combinatorial approaches work only in two dimensions and rely on a theorem of Laman [Lm70] (also proved in [LY82]). A consequence of this theorem is that counting degrees of freedom can be reduced to computing a maximum independent set in a certain matroid. Elegant algorithms that test independence have been devised. One, due to Lovász and Yemini [LY82], uses a theorem of Nash–Williams to reduce the problem to finding a special decomposition of a graph into forests. Another approach, due to Crapo [Cr93], reduces the problem in a different way to finding a decomposition of a graph into trees. Both of these algorithms are implemented using a classic matroid-partitioning algorithm of Edmonds. Gabow and Westermann [GW88] devised new general matroid algorithms, which they specialized to give an $O(n^2)$ algorithm for computing degrees of freedom in two dimensions. Hendrickson [He90] gave a different $O(n^2)$ algorithm based on bipartite matching.

Despite considerable effort, as noted in [He90], [LY82], Laman's theorem does not seem to generalize to three or more dimensions. No combinatorial algorithm for computing the generic degrees of freedom is known in higher dimensions, although interesting related results have been obtained, for example, by Tay [Ta84] and White and Whiteley [WW87].

This paper provides a new combinatorial strategy that yields a practical algorithm for computing a nontrivial lower bound on the degrees of freedom and suggests a new approach for computing degrees of freedom combinatorially. The algorithm given here has been implemented and used to approximate degrees of freedom in network models much larger than were examined previously. This work has already uncovered an unexpected empirical relationship between the rigidity and the average network degree in the standard model of glasses described below in §8 [FT92]. The algorithm can be adapted for networks that are embedded in two or three dimensions and have either universal or fixed-angle joints (as discussed below). Numerical results are given here for a well-known network model of glasses which suggest that the bound produced by the algorithm compares favorably with the exact answer. An efficient implementation of the algorithm that has proved practical for inputs of up to 4,000 vertices is given. For bounded-degree graphs, the implementation uses $O(n)$ space and $O(n^2)$ time and is much faster than standard matrix methods.

There are two main results in this paper which are combined to construct the algorithm. The first of these is a recurrence relation (inequalities 3(a)–(c) of §5) for computing a lower bound on the number of degrees of freedom. The recurrence relation is derived from a simple but powerful constraint-counting argument [Ma1864], [Ph82], [Th83] that is implicit in [Lm70] as well. The recurrence is used to construct the chain decomposition algorithm (§5), in which the graph is first reduced to a collection of cycles by the successive removal of special paths called "chains." The order in which chains are removed is crucial to the quality of the lower bound. The second main result is that one can obtain a "locally optimal" chain decomposition by using two simple rules for selecting chains (Theorem 1, §6).

Since an important motivation for this paper is to provide a useful tool for studying rigidity in models of glasses as in [HT85], the results here are stated for frames with *fixed-angle joints,* in which the angle between each pair of adjacent edges is specified. In the existing mathematical literature on rigidity, frames are assumed to have *universal joints,* meaning that edges may rotate freely about the vertices. All of the results here can easily be restated for frames with universal joints. The fixed-angle joint problem can in fact be reduced to the universal joint problem by adding bars between each pair of second neighbors in the graph; the algorithm presented here is more direct and faster in practice.

The remainder of this paper is organized as follows. In §2, the number of degrees of freedom of a frame ($\psi$) is defined. In §3, the fundamental constraint-counting strategy is explained. In §§4 and 5, the recurrence relation for counting constraints and the resulting chain decomposition (CD) algorithm are derived. Rules for choosing a locally optimal chain decomposition are given in §6. Implementation of the algorithm is sketched in §7, and computational results are presented in §8. A summary of results and open questions is given in §9.

**2. Terminology and definition of degrees of freedom.** Throughout this paper, standard graph-theoretic terminology will be used [BM76]. A *graph* or *network* $G = (V, E)$ consists of a set $V$ of *vertices* and a set $E$ of *edges,* unordered pairs $[i, j]$ of distinct vertices in $V$. The terms *frame* and *network model* will be used to refer to a graph such that each vertex $i$ is assigned a point $\mathbf{b}_i$ in Euclidean $k$-dimensional space, and each edge is a straight line segment. For concreteness, assume $k = 3$ (it is straightforward to modify the discussion in this section for arbitrary $k$). Both $V$ and $E$ are assumed to be finite. Use of the methods here for infinite, periodic systems is discussed in §8.

To determine the rigidity of a frame, one treats the edge lengths and angles between adjacent edges as constraints on "motions" of the vertices and then computes the number of degrees of freedom, here called $\psi$. This number is defined as the dimension of the subspace of "infinitesimal motions" that do not violate the constraints. A detailed sketch of the definition and its derivation is provided in this section. However, the results of this paper depend only on the inequalities (2a, b) of §3. Readers interested only in the lower bound algorithm may go directly to §3.

An "infinitesimal motion" of a vertex is represented by adding a "small" displacement vector $\mathbf{y}_i$ to $\mathbf{b}_i$. Let $\mathbf{b}_{ij} = \mathbf{b}_j - \mathbf{b}_i$ and $\mathbf{y}_{ij} = \mathbf{y}_j - \mathbf{y}_i$. Let $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_{3n}) = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ be the vector describing the complete set of $3n$ independent vertex displacements. To determine whether such a displacement violates the constraints imposed by the rigid bars and fixed-angle joints, one can introduce a "potential" function $U(\mathbf{x})$ such that $U(\mathbf{x}) = 0$ if and only if $\mathbf{x}$ is a motion that does not violate the constraints. For example, if $\mathbf{x}$ is a (rigid) translation of the system, $U(\mathbf{x}) = 0$. Since we are only concerned with $U(\mathbf{x})$ for $\mathbf{x} \approx \mathbf{0}$, one may use the quadratic potential [HT85]

$$U(\mathbf{x}) = \sum(\mathbf{b}_{ij} \cdot \mathbf{y}_{ij})^2 + \sum(\mathbf{b}_{ij} \cdot \mathbf{y}_{ik} + \mathbf{y}_{ij} \cdot \mathbf{b}_{ik})^2. \tag{1}$$

The first sum is over all pairs $(i, j)$ such that $[i, j]$ is an edge of $G$. The second sum is over all triples $(i, j, k)$ such that $[i, j]$ and $[i, k]$ are each edges. Note that if one includes only the first sum, $U$ is a potential function for a system with universal joints.

One can show that the set $\{\mathbf{x} | U(\mathbf{x}) = 0\}$ is a linear subspace. First observe that $U(\mathbf{x}) = \mathbf{x}^T\mathbf{D}\mathbf{x}$, where $\mathbf{D}$ is the $3n \times 3n$ symmetric matrix with terms

$$D_{pq} = \frac{\partial^2 U}{\partial x_p \partial x_q},$$

where the partial derivatives are evaluated at $\mathbf{x} = \mathbf{0}$. One can then show that $\mathbf{D}\mathbf{x} = \mathbf{0}$ if and only if $U(\mathbf{x}) = 0$.

The matrix $\mathbf{D}$ is called the *dynamical matrix* [AM76] and has a simple physical interpretation. If $\mathbf{x} = \mathbf{x}(t)$ is a function of time and if $U(\mathbf{x})$ represents the potential energy of the system, then the equations of motion for the system are of the form $\mathbf{C}\mathbf{x}''(t) + \mathbf{D}\mathbf{x}(t) = \mathbf{0}$. Thus, $\mathbf{D}$ is the matrix analog to the force constant of a spring obeying Hooke's law. (See [LT85, §5.12].)

The *number of degrees of freedom* of a frame $G$ is defined to be the dimension of the kernel of $\mathbf{D}, \{\mathbf{x} | \mathbf{D}\mathbf{x} = \mathbf{0}\}$, and will be denoted here by $\psi = \psi(\mathbf{D})$. (In lattice dynamics, a basis vector of the kernel is called a "floppy" or "zero-frequency" mode and represents a vibrational mode that does not increase the energy of the system.)

There is an equivalent definition that is the foundation of all combinatorial methods for computing $\psi$. First write $U = \mathbf{g}^T\mathbf{g}$, where $\mathbf{g}$ is the vector whose components are $(\mathbf{b}_{ij} \cdot \mathbf{y}_{ij})$ and $(\mathbf{b}_{ij} \cdot \mathbf{y}_{ik} + \mathbf{y}_{ij} \cdot \mathbf{b}_{ik})$. Let $\mathbf{A}$ be the matrix such that $\mathbf{g} = \mathbf{A}\mathbf{x}$; then, each row of $\mathbf{A}$ represents exactly one constraint. One can show that $\mathbf{D} = \mathbf{A}^T\mathbf{A}$, and $\operatorname{rank}(\mathbf{D}) = \operatorname{rank}(\mathbf{A})$. Thus, since $\operatorname{rank}(\mathbf{A})$ is the number of independent rows (and columns) of $\mathbf{A}$ and the number of columns of $\mathbf{A}$ is $3n, \psi = 3n - \operatorname{rank}(\mathbf{A})$. In other words, we have the well-known result that $\psi$ is $3n$ (the number of degrees of freedom of $n$ unconstrained vertices) minus the number of independent constraints [Th83]. $\mathbf{A}$ is also called the *rigidity matrix.*

There are various geometric degeneracies that can affect $\psi$. [Gu90] has a detailed discussion in the case of universal joints in two dimensions (see also [WW87]). The case of fixed-angle joints is even more complex. However, we assume throughout that all graphs are embedded generically as in [LY82], and hence $\psi(G)$ is well defined.

We need another simple but important observation. If $G$ has three or more vertices (which do not all lie on a line), the kernel of $\mathbf{D}$ must contain six independent vectors representing three rigid translations and three rigid rotations. Hence, we shall assume throughout that $\psi \geq 6$. Note that in $k$ dimensions, there are $k$ independent translations and $k(k-1)/2$ rotations.

## 3. Computation of $\psi$ by counting constraints.

A simple lower bound on $\psi$, the number of degrees of freedom, is given in [Th83] and is the starting point for the algorithm given here. The bound comes from the definition of $\psi$ as $3n$ minus the number of nonredundant constraints, where a single constraint is either an edge length or an angle between a pair of adjacent edges. Let the *degree* of a vertex $v, \deg(v)$, be the number of edges $[v, w]$ that have $v$ as an end point. Although there are $\mathrm{C}(\deg(v), 2)$ angle pairs at a given vertex $v$, if the degree of $v$ is at least 2, the number of independent angle constraints associated with $v$ is at most $2\deg(v) - 3$. If $m$ is the number of edges there are at most $m$ edge-length constraints. Thus, one obtains

$$\text{(2a)} \qquad \psi(G) \geq 3n - m - \sum (2\deg(v) - 3),$$

where the sum is over all vertices of degree 2 or more. Assuming $G$ has three or more vertices, there are six rigid motions, so we also have

$$\text{(2b)} \qquad \psi(G) \geq 6.$$

The maximum of the right sides of (2a) and (2b) will be referred to henceforth as the *mean field bound,* using the terminology of [Th83]. Despite the simplicity of this bound, it has had considerable impact on the understanding of glasses [HT85], [Ph82], [Th83]. (In view of (2b), many authors define the degrees of freedom to be $\psi - 6$; it is straightforward to rewrite the results here using the alternate definition.)

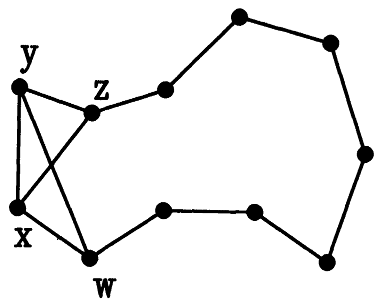


FIG. 1. *Example in which the mean field bound overcounts constraints. Since three constraints determine a triangle, constraints on edges $\overline{yz}, \overline{zx}$ and angle $\angle yzx$ also determine $\overline{yx}, \angle zyx$, and $\angle yxz$; the constraints on $\overline{xw}, \angle yxw$, and $\angle zxw$, determine $\overline{yw}, \angle xyw, \angle xwy$, and $\angle wyz$, for a total of seven redundant constraints.*

It is not hard to construct examples in which $\psi$ is larger than the mean field bound. For example, in the frame of Fig. 1 with $n = 11$ and $m = 13$, applying (2) gives $\psi \geq 6$. However, seven of the constraints are redundant, so in fact, $\psi \geq 8$. The method introduced in the following sections finds such redundant constraints essentially by detecting subgraphs in which (2b) rather than (2a) is applicable.

**4. Recurrence relation for counting constraints.** The new approach to computing a lower bound on $\psi$ is based on a recurrence relation derived from the inequalities in (2). The terminology below will be used to express this relation in a compact form.

A *path* of length $q$ in $G$ is a subgraph containing a sequence of $q$ edges in which at most two edges share any vertex; i.e., a subgraph containing distinct vertices, $h_0, h_1, h_2, \ldots, h_q$ and $q$ edges, $[h_0, h_1], [h_1, h_2], \ldots, [h_{q-1}, h_q]$. A path will usually be represented by listing the sequence of vertices with no commas: $h_0h_1h_2 \ldots h_q$. The length of (number of edges in) path $P$ will be denoted by $|P|$. A *cycle* of length $q$ is the same as a path of length $q$ except that its end points are the same ($h_0 = h_q$).

A *chain* is a path or cycle $C$ such that, if vertex $i$ is not an end point of $C$, then the degree of $i$ in $G$ is exactly 2 and if $i$ is an end point of $C$, then the degree of $i$ in $G - C$ is at least 2. ($G - C$ is the graph obtained by deleting the edges of $C$ and then deleting any vertices of degree zero.) Note that if $C$ is a path, then each end point has degree 3 or more and if $C$ is a cycle, then its (single) end point has degree 4 or more. (Under certain conditions, a chain is also called an "ear.")

For example, in the (disconnected) graph in Fig. 2(d) in §5, *ijkl* and *nopn* are both chains of length 3 and *mr* is a chain of length 1. The cycle *ihgfi* is not a chain because the degree of $i$ is only 3; *abcdea* is not a chain because *every* vertex has degree 2. Note that each graph in Fig. 2 is $G - C$ for a chain in the succeeding graph. The recurrence in the following lemma is obtained by counting constraints due to a single chain.

LEMMA 1. *If $C$ is a chain in $G, \psi(G) \geq \psi(G - C) + |C| - 6$.*

*Proof.* Observe that

$$\begin{aligned}\psi(G) = 3n &- \#\{\text{constraints in } G - C\} \\ &- \#\{\text{constraints due to } C \text{ alone}\} \\ &- \#\{\text{constraints at end points of } C\}\end{aligned}$$

and

$$\psi(G-C) = 3(n-|C|+1) - \#\{\text{constraints in } G-C\}.$$

For $C$ alone, there are $|C|$ edge constraints and $|C|-1$ angle constraints. Whether $C$ is a cycle or path, the number of additional independent angle constraints at the end point(s) of $C$ is at most four. The result then follows from simple algebra. □

In a graph of minimum degree 2, notice that if $C$ is a chain, then $G-C$ also has minimum degree 2. (The *minimum degree* of a graph is the minimum degree among all vertices.) Furthermore, $G$ has no chains if and only if *every* vertex has degree 2, i.e., $G$ is a union of disjoint cycles. Thus, in this case, a basis for the recurrence is provided by the following lemmas, which are proved easily by counting constraints.

LEMMA 2. *If a graph $R$ is a cycle, then*

$$\psi(R) \geq |R|.$$

LEMMA 3. *If $G$ has connected components $H_1, H_2, \ldots, H_m$, then*

$$\psi(G) = \sum \psi(H_i).$$

Throughout this paper, all graphs are assumed to have minimum degree at least 2. There is no loss of generality because vertices of degree 0 or 1 can be deleted first and taken into account by using constraint-counting arguments as in §3. If $u$ has degree 0, then $\psi(G) = \psi(G-u)+3$. If $u$ is a vertex of degree 1, with neighbor $v$, let $G_1$ be $G$ with edge $[u,v]$ deleted (but $v$ remaining). If $\deg(v) < 3$ in $G$, then $\psi(G) \geq \psi(G_1) + 3 - \deg(v)$. If $\deg(v) \geq 3$, it follows that $\psi(G) \geq \psi(G_1)$.

## 5. Chain decomposition algorithm.

In the chain decomposition (CD) algorithm, chains are removed until only cycles remain. Therefore, $\psi$ is initialized using Lemmas 2 and 3. Finally, chains are added back, using Lemma 1 to update $\psi$ at each step.

To describe the decomposition the following notation will be used. Let $P^* = (P_1, P_2, \ldots, P_k)$ be any sequence of paths or cycles. Let $G(0) = G, G(1) = G - P_1$, and in general, $G(i) = G(i-1) - P_i$. If $P_i$ is a chain in $G(i-1)$ for all $i$, then $P^*$ is a *chain sequence* in $G$. If, in addition, $G(k)$ contains no chains, then $P^*$ is a *chain decomposition* of $G$. (This is similar to an "ear decomposition.") One possible chain decomposition for the graph of Fig. 2(d) is $(mr, lrqn, ijklmn)$; another is $(nopn, mnqr, ijkl)$.

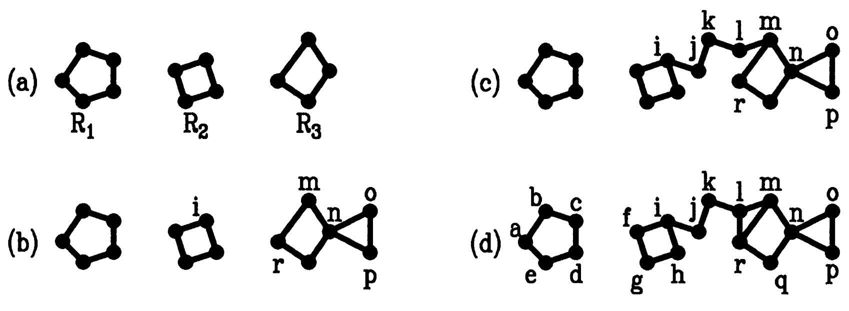


FIG. 2. *Example of CD algorithm on the network in* (d), *given the chain decomposition* $(lr, ijklm, nopn)$. (a) $\psi_1(i) := 6, i = 1,2,3$; (b) $\psi_1(3) := \max\{6+3-6, 6\} = 6$; (c) $\psi_1(2) := 6+6+4-6 = 10, \psi(3) := 0$; (d) $\psi_1(2) := \max\{10+1-6, 6\} = 6, \psi_1 := 6+6+0 = 12$.

To use the recurrence effectively, one must refine Lemmas 1 and 2 using (2b) to give

$$\psi(G) \geq \max\{\psi(G-C)+|C|-6, 6\}, \tag{3a}$$

$$\psi(G) \geq \max\{|R|, 6\}. \tag{3b}$$

(This is valid since all graphs have minimum degree 2 and 3 or more vertices.)

Another simple but key observation is that the constraints introduced by a chain can only affect the vertices in the connected component containing that chain. Thus, if $C$ is contained in component $H$ and if $G-H$ is nonempty (i.e., $\psi(G-H) \geq 6$), Lemma 1 can be further refined to give

$$\psi(G) \geq \psi(G-H)+\psi(H-C)+|C|-6. \tag{3c}$$

The CD algorithm is given below and illustrated in Fig. 2. The input is a graph $G$, of minimum degree 2 (see the discussion at the end of §4). The output of the algorithm is a lower bound on $\psi$, denoted $\psi_1(G)$. The notation $\psi_1(i)$ means $\psi_1$ of the current component $i$.

```
CHAIN DECOMPOSITION ALGORITHM
1. Find a chain decomposition of $G$: $C^* = C_1, C_2, \ldots, C_k$.
2. Let $R_1, R_2, \ldots, R_j$ be the cycles that remain after the chains are removed
   from $G$.

   **For** each $i$, let component $i$ be $R_i$; $\psi_1(i) := \max\{|R_i|, 6\}$. **End for.**

3. **For** each $m$ from $k$ down to 1:
   *Case* 1. $C_m$ has both end points in component $i$.
       Add $C_m$ to component $i$; $\psi_1(i) := \max\{\psi_1(i)+|C_m|-6, 6\}$.
   *Case* 2. $C_m$ has end points in distinct components $i$ and $j$ $[\psi_1(i), \psi_1(j) \geq 6]$.
     (a) $\psi_1(i) := \psi_1(i)+\psi_1(j)+|C_m|-6; \psi_1(j) := 0$.
     (b) Add component $j$ and $C_m$ to component $i$; delete component $j$.
   **End for**
4. $\psi_1 := \sum \psi_1(i)$.
```

## 6. Finding a good chain decomposition.

The order of chains in a decomposition is critical. For example, the chain decomposition used in Fig. 2 gives the bound $\psi_1 = 12$, whereas the decomposition $(ijkl, nopn, mnqr)$ for the same frame yields the better bound $\psi_1 = 15$.

Let $\psi_1(C^*)$ be the output of the CD algorithm using the decomposition $C^*$. Ideally, one would like to find an *optimal* chain decomposition, i.e., a $C^*$ such that $\psi_1(C^*) \geq \psi_1(\widehat{C}^*)$ for any other chain decomposition $\widehat{C}^*$. I have not yet found an efficient method to construct an optimal decomposition but have discovered two rules that are necessary in an optimal decomposition and sufficient to produce a *locally optimal* decomposition (see Theorem 1 below). The first rule, motivated by examples such as in Fig. 2, is to create new connected components whenever possible. The second rule, motivated by examples such as in Fig. 1, is to remove the longest chains first. Using these rules as heuristics works well in practice.

The terms used in Theorem 1 are as follows. First, a chain $C$ in a connected component $H$ will be called a *disconnecting chain* if $H-C$ has two nonempty connected components (e.g., *ijkl* in Fig. 2(d)). Otherwise, $C$ is a *nondisconnecting chain.* Next, a chain decomposition, $C^* = (C_1, \dots, C_i, C_{i+1}, \dots C_k)$, is *locally optimal* if, given any $i$ such that it is possible to exchange $C_i$ and $C_{i+1}$ (i.e., $\widehat{C}^* = (C_1, \dots, C_{i+1}, C_i, \dots, C_k)$ is also a chain decomposition), $\psi_1(C^*) \geq \psi_1(\widehat{C}^*)$. In other words, $C^*$ is locally optimal if no interchange of successive chains leads to a better decomposition.

THEOREM 1. *The chain decomposition $C^* = (C_1, C_2, \dots)$ is locally optimal if, for each $i \geq 1$, chain $C_i$ is selected using the following two rules:*

1. *If there is any disconnecting chain in $G(i-1)$, then $C_i$ is a disconnecting chain.*

2. *Otherwise, $C_i$ is a nondisconnecting chain of maximum length in $G(i-1)$. (Recall that $G(i-1)$ is the graph $G$ after chains $C_1, C_2, \dots, C_{i-1}$ have been removed.)*

*Proof.* See the appendix. □

**7. $O(m^2)$ implementation.** The CD algorithm has three phases: (1) selecting a chain decomposition; (2) initializing $\psi_1$ of each remaining cycle; (3) adding chains back in reverse order, updating $\psi_1$ at each step. Detecting a disconnecting chain is similar to the problem of finding an "articulation point" in a graph, so a natural strategy for selecting each chain in the decomposition is a modified depth-first search [Ba88, p. 184]. Since each edge is in at most one chain, this leads to an $O(m)$ bound on both time and storage for finding each chain (where $m$ is the number of edges).

It is not hard to show that the number of chains in a decomposition is exactly $m-n$: letting $c$ be the number of chains in a decomposition, $2m = \sum \deg(\mathbf{v}) = 2n+2c$. Hence, phase 1 can be implemented with $O(m)$ space and $O(m^2)$ time. Since the graphs of interest in applications are all of bounded degree, in practice, the algorithm uses $O(n)$ space and $O(n^2)$ time. This is a significant savings over straightforward rank or eigenvalue computation, which uses $\Theta(n^2)$ storage and $\Theta(n^3)$ time.

For very large networks it would be desirable to reduce the time needed to select each chain. An obvious improvement is to store the lengths of nondisconnecting chains in a heap [Ba88, p. 71] in order to facilitate finding a maximum length chain. Removing a single chain affects at most three existing chains since at most two new vertices of degree 2 are created, so the heap could be updated in $O(\log m)$ steps. Thus, it is likely that there is an $O(m \log m)$ implementation of phase 1; the key problem that remains is how to update the depth first search "forest" efficiently after each chain removal.

One strategy for reconstructing the graph in phase 3 is to treat it as a sequence of $m-n$ "union-find" operations on the set of $m$ edges, which can be implemented well within $O(m \log m)$ time [Ba88, p. 304]. However, one can also use the recursive structure of the algorithm to perform phases 2 and 3 efficiently. The main observation is that each chain that is removed can be associated with a node in a binary tree. This is illustrated in Fig. 3 for a decomposition of one component of the graph of Fig. 2(d). The root of the tree represents an initial graph $G$ (assumed connected). When a chain is removed, a child is created for each resulting component (one or two). A node is a leaf if the corresponding component is a single cycle.

To compute $\psi_1$, define $\psi_1(k)$ for each node $k$ of the binary tree. Then $\psi_1(k)$ is computed when either $k$ is a leaf (using Lemma 2) or $\psi_1$ has been computed for the children of $k$ (using the refined version of Lemma 1). Then, $\psi_1$ of the root is $\psi_1(G)$.

If the nodes of the binary tree are created in depth-first order (as indicated in Fig. 3) then in order to represent the component corresponding to a node, one need only store a pointer to the root of a tree in the current depth-first search forest (for the original graph). The only other information that one must store at a node is

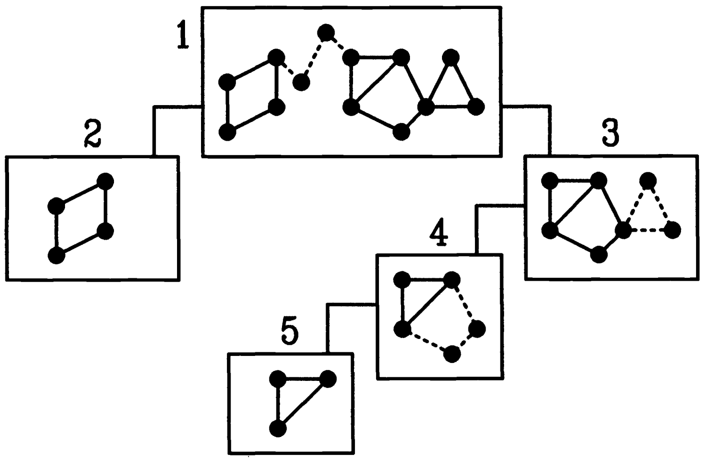


FIG. 3. *Binary tree representation of components created by removing chains in a decomposition. The nodes (rectangular boxes) are numbered in depth-first order, as generated; the node at the top is the root. The chain corresponding to each node is shown as a dotted line. The graph is a component of the network in Fig.* 2(d); *the chain decomposition* (*ijkl*, *nopn*, *mnqr*) *is selected using the rules of* §6. *The values of* $\psi_1$ *are computed while "backing up" from each node and so are computed in the following order:* $\psi_1(5) := 6, \psi_1(4) := 6, \psi_1(3) := 6, \psi_1(2) := 6, \psi_1(1) := 9$.

(1) pointers to its parent and children, (2) the length of the chain removed, and (3) the value of $\psi_1$ for the node. Since the tree has $O(m)$ nodes, phases 2 and 3 can be completed in only $O(m)$ steps, using $O(m)$ additional storage. (Note: to reduce the number of tree nodes, children are generated for a node only for disconnecting chains; information on nondisconnecting chains can be stored in one separate stack.)

**8. Illustration of lower bound on a standard network model.** Although it would be most desirable to have a guaranteed bound on the error $\psi - \psi_1$, I do not know of such a bound at present. I have, however, tested the algorithm extensively on a class of network models commonly used to study rigidity of glasses [HT85]. Typical output is illustrated in Fig. 4. The results show that the lower bound is generally quite close to the exact answer and is substantially better than the (linear) mean field bound, which was previously the only bound available for systems too large to be treated with matrix methods.

The graphs used here were derived from a diamond structure [AM76]. Each sequence of input graphs was obtained by deleting edges selected at random. (Such a model has been used to simulate glasses made up of atoms that can form two, three, or four bonds with neighboring atoms [HT85].) For consistency, in the rare instances in which the depletion resulted in a disconnected graph, that trial was not used. Also, if a selected edge had an end point of degree 2 it was not removed, so that all graphs had minimum degree 2.

The plots in Fig. 4 show the results of computing $\psi_1$ as well as the mean field bound (2a) and the exact value of $\psi$ for two sequences of graphs obtained as described above. The exact answer was computed from the dynamical matrix $\mathbf{D}$ by counting the number of zeros in the list of eigenvalues; $\psi_1$ was computed by implementing the CD algorithm.

The average degree $\bar{r}$ (the sum of the degrees divided by the number of vertices) was chosen as the independent variable to facilitate comparison with [HT85]. For a graph of minimum degree 2, the mean field bound is linear in $\bar{r}$, which can be seen by

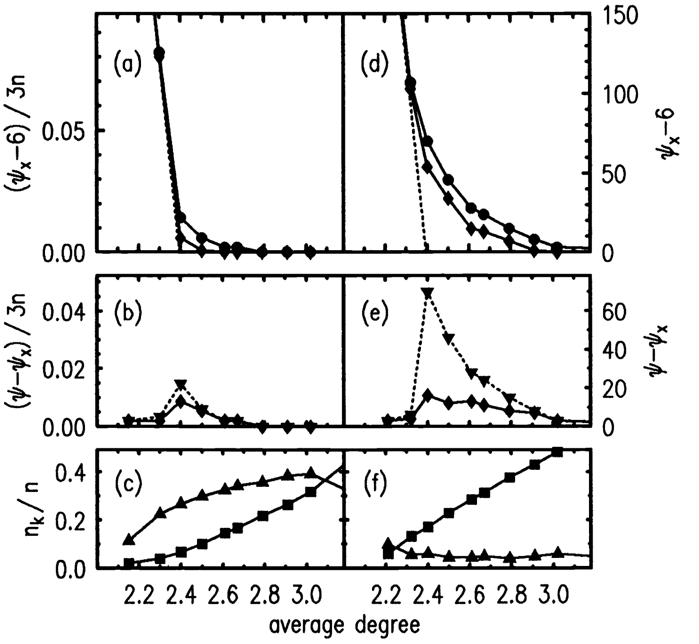


FIG. 4. *Comparison of lower bounds on a finite graph* ($n = 512$), *representing a diamond structure network with random edge removal* (a–c) *and biased edge removal* (d–f). $\psi_x$ *denotes some computed value of* $\psi$. *The number of vertices is* $n$, *and* $n_k$ *is the number of vertices of degree* $k$, *where* $n = n_2 + n_3 + n_4$. *The right axis is labeled with true values; the left axis is labeled with the same values divided by* $3n$ *to be independent of network size; both the left and right labels apply to each figure.* (a, d) *average degree versus* $(\psi_x - 6)$, *where* $\psi_x$ *is the mean field bound* (*dotted line*), $\psi_1$ (*diamonds*), *and the exact value* ($\psi + 3$ *as explained in text*) (*circles*); (b, e) *average degree versus error,* $(\psi + 3 - \psi_x)$, *where* $\psi_x$ *is the mean field bound* (*inverted triangles*) *and* $\psi_1$ (*diamonds*); (c, f) *average degree versus* $n_k/n$ *for* $k = 3$ (*triangles*) *and* $k = 4$ (*squares*).

applying $m = (1/2)\sum \deg(v) = (n/2)\bar{r}$ to (2a) to get

$$\psi(G) \geq n\left(6 - \frac{5}{2}\bar{r}\right). \tag{4}$$

The degree $\bar{r} = 2.4$ is a critical point, since $(6 - (5/2)\bar{r}) = 0$ there; the results of [HT85], [Th83] suggest that there is a transition between floppy and rigid states in glasses that occurs near this critical degree. This is reflected in the errors in the approximations, plotted in Figs. 4 (b, e). In each case, both the mean field bound and the CD algorithm are very accurate when the average degree is far from 2.4, and the largest error occurs at 2.4. However, the CD algorithm gives a much more accurate overall picture of the behavior of $\psi$ .

In the graphs represented in Figs. 4(a–c), edges were selected uniformly at random for removal. To verify the robustness of the CD algorithm, the graphs represented in Figs. 4(d–f) were constructed by biasing the random edge selection to increase the number of vertices of degrees 2 and 4, which would be expected to produce more long (floppy) chains. The difference between the two methods of edge selection can be seen in Figs. 4(c, f), which give the fractions of degree 3 and degree 4 vertices obtained for each trial.

The error in the mean field bound, shown in Figs. 4(b, e), is significantly larger when biased rather than random depletion is used. However, the error in the CD bound is relatively small in both cases. In another test case (not shown), I used extremely biased depletion in which case edges were removed only from a fixed region of the initial graph, hence generalizing the example of Fig. 1. The mean field bound is very poor in this case, but the error in the CD bound was even smaller than in the cases shown. Thus, the CD algorithm is a useful tool for approximating the effects of the topology of a frame on its degrees of freedom.

Although the algorithm is described for finite frames, in the study of solids, an infinite model is generally preferred since surface effects are eliminated. The frames represented in Fig. 4 were derived from a 512-vertex, three-dimensional frame representing an infinite diamond structure. That is, the infinite periodic network is embedded in $\mathbf{R}^3$, an appropriate parallelpiped is selected as a "unit cell," and the plane boundaries of the cell are identified (as in a torus). (I also tested the algorithm using a finite fragment of the diamond structure and achieved with similar results.)

Three "extra" zero-frequency modes must be added to the exact value of $\psi$ for an infinite system. These represent three rigid rotations, which are included implicitly in computing $\psi_1$ but which do not appear as vectors in the kernel of the dynamical matrix. The reason is that in the potential (1) of §2, the edges represent vectors between vertices in the *infinite system* and each vertex is actually an infinite equivalence class of vertices. A displacement vector $\mathbf{x}$ represents the simultaneous identical displacements of all vertices in each class. Thus, a rigid rotation of the vertices in the finite representation is generally *not* a degree of freedom (zero-frequency mode) of the infinite system.

**9. Conclusions.** This work introduces a new combinatorial approach to the problem of determining the rigidity of a network of rigid bars. I have described an algorithm based on this approach that computes a lower bound on the number of degrees of freedom for frames with fixed-angle joints in three dimensions which has proven useful in the study of network models of glasses. The method can easily be adapted for work for frames that have universal joints; one need only modify the inequalities in (2) and (3).

A natural next step is to compute nontrivial bounds on the error $\psi - \psi_1$. I have preliminary results that promise a complementary algorithm for computing an upper bound on $\psi$, which would allow one to bound the error for any specific input. It would also be of interest to find an efficient algorithm for determining an optimal chain decomposition, i.e., one that maximizes $\psi_1$. The more difficult question remains open, however: for frames in generic position and arbitrary dimension, does there exist an efficient combinatorial algorithm for computing $\psi$ exactly?

**Appendix: Proof of Theorem 1.** Given any $i$, let $C^* = (C_1, \ldots, C_i, C_{i+1}, \ldots, C_k)$ and $\widehat{C}^* = (C_1, \ldots, C_{i+1}, C_i, \ldots C_k)$. To prove the theorem, it is sufficient to show the following. If $C_i$ is a disconnecting chain or $C_{i+1}$ is a nondisconnecting chain and $|C_i| \geq |C_{i+1}|$, then $\psi_1(C^*) \geq \psi_1(\widehat{C}^*)$. This statement follows directly from a case analysis given in the three lemmas stated and proved below.

To simplify the notation, let $C_i$ and $C_{i+1}$ be the first two chains in a decomposition, called $C_A$ and $C_B$. We assume that both $C_A, C_B$ and $C_B, C_A$ are chain sequences, i.e., the two chains do not "interfere" with each other. (There are analogous lemmas even when the chains cannot be directly interchanged; they are more complicated to state and prove but follow from similar arguments.) The shorthand $\psi_1(C_A, C_B)$ and $\psi_1(C_B, C_A)$ will be used to mean the result of applying the CD algorithm to the respective decompositions $(C_A, C_B, C_1^*)$ and $(C_B, C_A, C_1^*)$, where $C_1^*$ is a fixed decomposition of $G - C_A - C_B$.

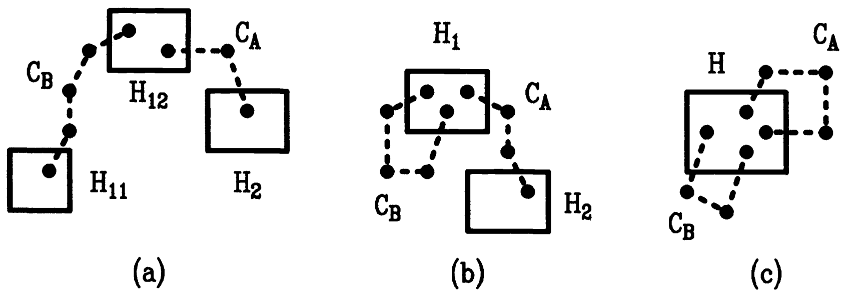


FIG. 5. *Representation of cases described in Lemmas* 4, 5, *and* 6. *Boxes represent connected subgraphs. Chains, shown as dotted lines, can have any number of edges.* (a) $C_A$ *and* $C_B$ *are both disconnecting chains* (*Lemma* 4); (b) $C_A$ *is disconnecting, but* $C_B$ *is nondisconnecting* (*Lemma* 5); (c) $C_A$ *and* $C_B$ *are both nondisconnecting* (*Lemma* 6).

To facilitate the proofs of the lemmas, let $((x)) = \max\{x, 6\}$. Then the following algebraic rules are straightforward to prove

(a) $x \leq ((x))$.
(b) If $x \leq y$, then $((x)) \leq ((y))$.
(c) If $y \geq 0$, then $((x + y)) \leq ((x)) + y$.
(d) If $x \leq y$, then $((z + y)) + x \leq ((z + x)) + y$.

LEMMA 4. *If* $C_A$ *and* $C_B$ *are disconnecting chains, then* $\psi_1(C_A, C_B) = \psi_1(C_B, C_A)$.

*Proof.* Assume that $C_A$ and $C_B$ are in the same connected component $H$ (otherwise the statement is trivial) and that $G = H$. Since $C_A, C_B$ and $C_B, C_A$ are both chain sequences, we can assume that $H - C_A$ has components $H_1, H_2$ and that $C_B$ is contained in $H_1$. Assume that $H_1 - C_B$ has components $H_{11}, H_{12}$ and that the end point of $C_A$ is in $H_{12}$. (See Fig. 5(a).)

Then, since $\psi_1(H_i) \geq 6$ for any component $H_i$,

$$\begin{aligned}\psi_1(C_A, C_B) &= \psi_1(H_1) + \psi_1(H_2) + |C_A| - 6\\ &= \psi_1(H_{11}) + \psi_1(H_{12}) + |C_B| - 6 + \psi_1(H_2) + |C_A| - 6,\end{aligned}$$

$$\begin{aligned}\psi_1(C_B, C_A) &= \psi(H_{12} \cup C_A \cup H_2) + \psi_1(H_{11}) + |C_B| - 6\\ &= \psi_1(H_{12}) + \psi_1(H_2) + |C_A| - 6 + \psi_1(H_{11}) + |C_B| - 6.\end{aligned}$$

Clearly, these are the same, only summed in different orders. □

LEMMA 5. *If* $C_A$ *is disconnecting and* $C_B$ *is nondisconnecting, then* $\psi_1(C_A, C_B) \geq \psi_1(C_B, C_A)$.

*Proof.* Again assume that $C_A$ and $C_B$ are in the same connected component $H$ and that $H - C_A$ has components $H_1$ and $H_2$. We may assume that $C_B$ has both end points in $H_1$ and that $G = H$. (See Fig. 5(b).)

Again, since $\psi_1(H_i)$ is always at least 6,

$$\begin{aligned}\psi_1(C_A, C_B) &= \psi_1(H_1) + \psi_1(H_2) + |C_A| - 6\\ &= ((\psi_1(H_1 - C_B) + |C_B| - 6)) + \psi_1(H_2) + |C_A| - 6,\\ \psi_1(C_B, C_A) &= ((\psi([H_1 \cup C_A \cup H_2] - C_B) + |C_B| - 6))\\ &= ((\psi_1(H_1 - C_B) + \psi_1(H_2) + |C_A| - 6 + |C_B| - 6)).\end{aligned}$$

Letting $x = \psi_1(H_1 - C_B) + |C_B| - 6$ and $y = \psi_1(H_2) + |C_A| - 6$ the result follows from algebraic rule (c). □

LEMMA 6. *If $C_A$ and $C_B$ are nondisconnecting and $|C_A| \geq |C_B|$, then $\psi_1(C_A, C_B) \geq \psi_1(C_B, C_A)$.*

*Proof.* Again, the only case of interest is when $C_A, C_B$ are in the same component $H$ and $G = H$. (See Fig. 5(c).) It may happen that $C_B$ becomes a disconnecting chain after $C_A$ is removed, and vice versa, but this does not affect the computations. We get

$$\begin{aligned}\psi_1(C_A, C_B) &= ((\psi_1(H - C_A) + |C_A| - 6)) \\ &= ((((\psi_1(H - C_A - C_B) + |C_B| - 6)) + |C_A| - 6)), \\ \psi_1(C_B, C_A) &= ((((\psi_1(H - C_A - C_B) + |C_A| - 6)) + |C_B| - 6)).\end{aligned}$$

If $z = \psi_1(H - C_A - C_B) - 6, x = |C_B|$, and $y = |C_A|$, then the result follows from algebraic rules (d) and (b). □

**Acknowledgments.** I am grateful for the use of the computing facilities at the IBM T. J. Watson Research Center. I thank R. Connelly, B. Servatius, J. Tersoff, and H. Wilf for helpful discussions.

## REFERENCES

[AM76] N. W. ASHCROFT AND N. D. MERMIN, *Solid State Physics*, Holt, Rinehart, and Winston, New York, 1976.

[Ba88] S. BAASE, *Computer Algorithms*, 2nd ed., Addison-Wesley, Reading, MA, 1988.

[BM76] J. A. BONDY AND U. S. R. MURTY, *Graph Theory with Applications*, Macmillan, London, 1976.

[Cr93] H. CRAPO, *On the generic rigidity of structures in the plane*, Adv. Appl. Math., to appear.

[FT92] D. S. FRANZBLAU AND J. TERSOFF, *Elastic properties of a network model of glasses*, Phys. Rev. Lett., 68 (1992), pp. 2172–2175.

[GW88] H. N. GABOW AND H. H. WESTERMANN, *Forests, frames and games: Algorithms for matroid sums and applications*, in Proceedings of the 20th Symposium on the Theory of Computing, ACM, 1988, pp. 407–421.

[Gu90] E. GUYON, S. ROUX, AND A. HANSEN, *Nonlocal and nonlinear problems in the mechanics of disordered systems: Application to granular media and rigidity problems (review)*, Rep. Progr. Phys., 53 (1990), pp. 373–419.

[HT85] H. HE AND M. F. THORPE, *Elastic Properties of Glasses*, Phys. Rev. Lett., 54 (1985), pp. 2107–2110.

[He90] B. A. HENDRICKSON, *The molecule problem: Determining conformation from pairwise distances*, Ph.D. Thesis, Technical Report 90-1159, Department of Computer Science, Cornell University, Ithaca, NY, 1990.

[Lg1788] J.-L. LAGRANGE, *Mécanique Analytique*, 1788. 4th ed. in Oeuvres de Lagrange, Vol. 11, Gauthier-Villars, Paris, 1887.

[Lm70] G. LAMAN, *On graphs and rigidity of plane skeletal structures*, J. Engrg. Math., 4 (1970), pp. 331–340.

[LT85] P. LANCASTER AND M. TISMENETSKY, *The Theory of Matrices*, 2nd ed., Academic Press, FL, 1985.

[LY82] L. LOVÁSZ AND Y. YEMINI, *On generic rigidity in the plane*, SIAM J. Alg. Discrete Methods, 3 (1982), pp. 91–98.

[Ma1864] J. C. MAXWELL, *On the calculation of the equilibrium and stiffness of frames*, Philos. Mag., 27 (1864), p. 294; in The Sci. Papers of J. C. Maxwell, Vol. 1, Cambridge Univ. Press, London, 1890, p. 598.

[Ph82] J. C. PHILLIPS, *The physics of glass*, Phys. Today, 35 (February 1982), p. 27.

[Ta84] T.-S. TAY, *Rigidity of multigraphs. I. Linking rigid bodies in n-space*, J. Combin. Theory Ser. B, 36 (1984), pp. 95–112.

[Th83] M. F. THORPE, *Continuous deformations in random networks*, J. Non-Cryst. Solids, 57 (1983), pp. 355–370.

[WW87] N. WHITE AND W. WHITELY, *The algebraic geometry of motions of bar-and-body frameworks*, SIAM J. Alg. Discrete Methods, 8 (1987), pp. 1–32.

SIAM J. DISC. MATH.
Vol. 8, No. 3, pp. 401–427, August 1995

# APPROXIMATION ALGORITHMS FOR MINIMUM-TIME BROADCAST *

GUY KORTSARZ[†] AND DAVID PELEG[†]

**Abstract.** This paper deals with the problem of broadcasting in minimum time in the telephone and message-passing models. Approximation algorithms are developed for arbitrary graphs as well as for several restricted graph classes.

In particular, an $O(\sqrt{n})$-additive approximation algorithm is given for broadcasting in general graphs, and an $O(\log n/\log\log n)$ (multiplicative) ratio approximation is given for broadcasting in the open-path model. This also results in an algorithm for broadcasting on random graphs (in the telephone and message-passing models) that yields an $O(\log n/\log\log n)$ approximation with high probability.

In addition, the paper presents a broadcast algorithm for graph families with small separators (such as chordal, $k$-outerplanar, bounded-face planar, and series-parallel graphs), with approximation ratio proportional to the separator size times $\log n$. Finally, an efficient approximation algorithm is presented for the class of graphs representable as trees of cliques.

**Key words.** broadcast, approximation

**AMS subject classifications.** 05C85, 68Q25, 90C27, 90B35

## 1. Introduction.

### 1.1. Minimum-time broadcast.

One of the most important efficiency measures of a communication network is the speed by which it delivers messages between communicating sites. Therefore, much of the research in this area concentrates on developing techniques for minimizing message delay.

This work concerns efficient algorithms for broadcast in a communication network. The network is modeled by a connected graph and assumes the *telephone* communication model (cf. [HHL88]). In this model messages are exchanged during *calls* placed over edges of the network. A round is a series of calls carried out simultaneously. Each round is assumed to require one unit of time, so round $t$ begins at time $t-1$ and ends at time $t$. A vertex may participate in at most one call during a given round, however, there are no limitations on the amount of information that can be exchanged during a given call. At a given round, if a call is placed over an edge $e$, we say that $e$ is *active* in this round, otherwise it is *idle*. The set of rules governing the activation of edges at each round is called a *schedule*.

A *broadcasting* problem refers to the process where a distinguished vertex $v$ originates a message $M$ that has to become known to all other processors. The efficiency of a broadcast scheme is usually measured by the number of time units it takes to complete the broadcast. Given a scheme $S$ for broadcasting in a graph $G$, denote the broadcasting time from $v$ using $S$ by $b(v,G,S)$. Define $b(v,G)$, the *broadcast time* of a vertex $v$ in $G$, as the minimum time for broadcasting a message originating at $v$ in

* Received by the editors March 22, 1993; accepted for publication (in revised form) June 1, 1994.

† Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot 76100, Israel. The research of the second author was supported in part by an Allon Fellowship, a Bantrell Fellowship, a Walter and Elise Haas Career Development Award, and a grant from the Israeli Basic Research Foundation.

$G$, i.e.,

$$b(v, G) = \min_S \{b(v, G, S)\}.$$

We denote it simply by $b(v)$ when the context is clear. We denote

$$b(G) = \max_v \{b(v, G)\}.$$

Given a network $G = (V, E)$ and an originator $u$, the *minimum broadcast time* (MBT) problem is to broadcast the message from $u$ to the rest of the vertices in $b(u)$ time units. This problem has received considerable attention in the literature. For example, broadcasting in trees is studied in [SCH81], and broadcasting in grid graphs is studied in [FH78]. For a comprehensive survey on the subject of gossiping and broadcasting see [HHL88].

The MBT problem in general graphs is NP-complete (cf. [GJ79]) and, thus, is unlikely to be solved exactly. However, several approaches toward coping with the MBT problem have been considered in the literature. A dynamic programming formulation for determining $b(v)$ and a corresponding broadcast scheme for an arbitrary vertex $v$ are proposed in [SW84]. Since the exact algorithm is not efficient for large networks, several heuristics are presented in [SW84] for achieving a broadcast scheme with good performance.

Note that at each round, the informed vertices transmit the message outside to the uninformed vertices. Since calls are placed along nonadjacent edges, the set of active edges at each round constitutes a *matching* between the informed and the uninformed vertices. In general it may be preferable to transmit the message first to vertices with certain properties, e.g., to vertices whose degrees are maximal among the uninformed vertices and thus are likely to be able to inform a large number of vertices. Thus the approach of [SW84] is based on assigning weights to the vertices and looking for a matching that also maximizes the sum of the weights of the vertices. The drawback of such a heuristic approach is that it provides no guarantee on the performance of the algorithm.

Consequently, in this paper we consider *approximation schemes* for broadcast, namely, algorithms that may not give an exact solution but still give a solution that is *guaranteed* to be "not too far" from the optimum.

**1.2. Related communication models and primitives.** Most of our results in what follows are formulated for the broadcast operation in the telephone model. However, a number of these results apply directly, or with minor changes, to some other related communication primitives and models. Let us now introduce these alternate models and primitives.

A generalization of broadcast that we consider is to assume the set $V_0$ of informed vertices at the beginning of the run need not consist of a single vertex but can be an arbitrary subset of $V$. Denote this problem by SMBT, and denote the time needed to broadcast from $V_0$ by $b(V_0, G)$ or $b(V_0)$. As mentioned in [GJ79], SMBT is NP-complete even for $k = 4$ where $k$ is the bound on the time for completing the broadcast.

Another important and well-studied communication primitive is the *gossip* operation. A gossip problem refers to the process of message dissemination, where each vertex $v$ originates a message $m_v$ and all messages have to become known to all vertices. The problem of performing the gossip primitive in minimum time is called the *minimum gossip time* (MGT) problem. The problem of efficient gossiping has also received considerable attention, mainly in the telephone model. For example, the papers [HMS72], [FP80] concern gossiping on the complete graph and on grid graphs, respectively.

Let us now turn to alternative communication models. Several generalizations of the telephone model appear in the literature. In [Far80], Farley suggests reconsidering the assumption that a vertex may call only neighboring vertices. Farley defines a possible variant of the model using long-distance calls, called the *open-path* model. In the open-path model, communication is carried along vertex disjoint paths. At each round, an informed member $v$ may call an uninformed vertex $u$ on an (arbitrarily long) path, adding $u$ to the set of informed vertices. Two paths corresponding to two different pairs must be vertex disjoint. We note the time needed to complete the broadcast from a distinguished vertex $v$ in the graph $G$ in the open-path model, by $b_{\mathrm{op}}(v, G)$ (or $b_{\mathrm{op}}(v)$). We also denote

$$b_{\mathrm{op}}(G) = \max_{v \in V}\{b_{\mathrm{op}}(v, G)\}.$$

The problem of broadcasting from a vertex $v$ in $b_{\mathrm{op}}(v, G)$ time units is referred to in what follows as OMBT.

In fact, Farley defines a second "long-distance" variant named the *open-line* model. This model is similar to the open-path model, except that the paths used in a communication round need only be *edge* disjoint. This model is less interesting for our purposes, since the problem of approximating broadcast in this model is essentially solved up to logarithmic factors. This is because, as shown in [Far80], the open-line model enables broadcast from an arbitrary vertex in $\lceil \log n \rceil$ time units.

Another common model of communication is the *message-passing* model, which is based on the assumption that a processor may send at most one fixed-size message in each time step, along one of its outgoing edges, but the communication pattern need not be a "matching." That is, it is possible that $v_1$ sends a message to $v_2$ while during the same round $v_2$ sends a message to $v_3$.

**1.3. Contributions.** In what follows we consider approximation schemes for broadcast (and some related primitives). Formally, we call an algorithm $A$ for broadcasting on a family of graphs $\mathcal{F}$ a *k-approximation scheme* if for every $G \in \mathcal{F}$ and vertex $v \in V$,

$$b(v, G, A) \leq k \cdot b(v, G).$$

We say that a scheme $S$ has a $(k, k')$-*approximation ratio* if

$$b(v, G, S) \leq k \cdot b(v, G) + k'.$$

A ratio is $k$-*additive* if it is an $(O(1), k)$-approximation ratio.

In this paper, we give approximation schemes for broadcast on several networks classes and analyze their approximation ratio.

The next two sections are dedicated to preparing the background for our approximation algorithms. Section 2 introduces the basic notions and definitions concerning transmission schedules and broadcast. Next, §3 presents some of our main technical tools. Specifically, it defines the *minimum weight cover* (MWC) problem and its close variant named the *minimum vertex weight cover* (MVWC) problem and provides a pseudopolynomial algorithm for solving them. The usefulness of this algorithm for handling transmission scheduling problems is demonstrated via a "toy example" of the *bipartite edge scheduling* (BES) problem. The MWC algorithm is used in virtually all our subsequent approximations for MBT.

In §4 we turn to approximations for the broadcast problem itself. To motivate the need for approximations, we examine three heuristics proposed in [SW84] for the

MBT problem, analyze their behavior on the wheel graph (see Fig. 2), and show that their output solution could be away from the optimum by a factor as high as $\sqrt{n}$ (on the $n$-vertex wheel). We then give an approximation algorithm for general $n$-vertex graphs with an $O(\sqrt{n})$-additive ratio.

Next, we show that in the open-path model, the OMBT problem has an approximation algorithm on arbitrary $n$-vertex graphs with a (multiplicative) approximation ratio $O(\log n/\log\log n)$. This algorithm has the additional desirable property that it solves the MBT problem (in the original telephone model) on random graphs (taken from the class $G_{n,p}$) and yields an approximation ratio $O(\log n/\log\log n)$ with high probability.

Section 5 presents an approximation scheme for broadcasting on graphs enjoying small separators, with approximation ratio $\varphi(n)\cdot\log n$ for graph families with separators of size $\varphi(n)$ (on $n$-vertex graphs). In particular, this algorithm yields an $O(\log n)$ approximation for *chordal* and $O(1)$-*separable* $n$-vertex graphs (including *series-parallel* graphs and $k$-*outerplanar* graphs for fixed $k$) and an $O(n^{1/4}/\sqrt{\log n})$ approximation for $n$-vertex *bounded-face* planar graphs.

Finally, in §6 we consider a special family of graphs called *trees of cliques,* generalizing trees, and give an approximation scheme for broadcasting on such graphs with additive-$O(\log^2 n)$ ratio.

Our results for the broadcast operation carry over to the gossip operation as well. To see this, note that in the telephone model, any scheme for broadcast can be used to perform the gossip operation. This is done as follows. First, fix a root vertex $v$ and perform a *convergecast* operation (which is the opposite primitive to broadcast), collecting all the messages from the rest of the vertices to $v$, using the broadcast scheme in reverse. Then $v$, knowing all the information, performs a broadcast of the combined message. It follows that our results for MBT hold for MGT as well.

Although we formulate our statements in the telephone model, virtually all our results for broadcast hold also for the message-passing model, since as far as the broadcast operation is concerned, these two models are equivalent in power. (This is no longer the case for more involved operations, such as gossip.)

## 2. Preliminaries.

### 2.1. Graph definitions.

Throughout this paper we use the following terminology to describe the behavior of the network. Our network is represented by a graph $G=(V,E)$, and we denote the number of vertices of a graph $G$ by $n$ and the number of edges by $m$.

Given a graph $G=(V,E)$ and two vertices $v,w\in V$, we denote the number of edges in a shortest path between $v$ and $w$ by $\mathrm{dist}(v,w)$. We denote $\mathrm{Diam}(v)=\max_w\{\mathrm{dist}(v,w)\}$. The diameter of the graph $G$ is $\mathrm{Diam}(G)=\max_v\{\mathrm{Diam}(v)\}$.

A *cluster* in a graph $G$ is a subset $V'$ of the vertices such that the subgraph induced by $V'$ is *connected.* Two clusters $V',V''$ are said to be *disjoint* if $V'\cap V''=\emptyset$. Two disjoint clusters $C_1$ and $C_2$ are said to be *independent* if there is no edge connecting a vertex of $C_1$ to a vertex of $C_2$.

DEFINITION 2.1. *Let $G=(V,E)$ be a graph, and let $S\subset V$ be a subset of the vertices. A subtree $T=(V_1,E_1)$ of $G$ rooted at a distinguished vertex $v_0$ is a* shortest paths tree ($SPT$) leading from $v_0$ to $S$ *iff $S\subseteq V_1$, each path from $v_0$ to $v_i\in S$ in $T$ is a shortest path in $G$, and every leaf of $T$ belongs to $S$. Denote an SPT leading from a vertex $v_0$ to a set $S$ by $SPT(v_0,S)$.*

We now state the definition of a *control graph* of a subset $V_0\subseteq V$, in a graph $G=(V,E)$. This definition will be useful in most of our approximation algorithms.

DEFINITION 2.2. *Suppose that the clusters (i.e., connected components) formed when extracting $V_0$ from the graph $G$ are $\{C_1, \dots, C_k\}$. The* control graph *of $V_0$ in $G$ is a bipartite graph $D_{V_0,G} = (V_1, V_2, A)$, where $V_1 = V_0, V_2 = \{C_1, \dots, C_k\}$, and $A$ contains an edge $(v, C_i)$ iff there is an edge between $v$ and some vertex of $C_i$ in $G$.*

### 2.2. Schedules and broadcast.

DEFINITION 2.3. *Given a graph $G = (V, E)$, two edges $e_1, e_2 \in E$ are called* adjacent *iff they share* exactly *one vertex and* nonadjacent *otherwise. A subset of edges $E' \in E$ is an* independent set *(of edges) iff every two edges $e_1, e_2 \in E'$ are nonadjacent.*

*Fact* 2.4. At each round $t$, the set of active edges is independent.

The MBT problem is formally defined as follows. Let $v_0 \in V$ be a distinguished vertex. A broadcast from $v_0$ is a sequence

$$\{v_0\} = V_0, E_1, V_1, E_2, \dots, E_k, V_k = V$$

such that for $1 \le i \le k$, the following hold.

1. $V_i \subseteq V$ and $E_i \subseteq E$.
2. Each edge in $E_i$ has exactly one end point in $V_{i-1}$.
3. The set $E_i$ is an independent set of edges.
4. $V_i = V_{i-1} \cup \{v : (u, v) \in E_i\}$.

In this case we say that the broadcast is performed in $k$ time units. The MBT problem concerns looking for the minimal $k$ such that broadcasting in $k$ time units is possible, for a given graph $G$ and vertex $v_0$. This $k$ is denoted $b(v_0, G)$.

For any connected graph $G$ of $n$ vertices and originator $u, b(u) \ge \lceil \log n \rceil$,[1] since in each time unit the number of informed vertices can at most double. Another simple lower bound for $b(v)$ for an arbitrary $v$ is $b(v) \ge \text{Diam}(v)$, since a vertex may only send information to a neighboring vertex at each round. An example of a graph for which $b(G) = \lceil \log n \rceil$ is $K_n$, the complete graph of $n$ vertices.

In any connected graph $G$, a broadcast from a vertex $u$ determines a spanning tree rooted at $u$. The parent of a vertex $v$ is the vertex $w$ that transmitted the message to $v$. Clearly, one may assume that such a vertex is unique. Even when using an arbitrary spanning tree, it is clear that at each step the set of informed vertices grows by at least one. Thus for each network $G, b(G) \le n - 1$. We cannot always improve upon this result. For example, in $S_n$, the star of $n$ vertices, the broadcast time is $b(S_n) = n - 1$. We summarize this discussion as follows.

*Fact* 2.5.

1. For every graph $G = (V, E)$ and vertex $v \in V$, $\lceil \log n \rceil \le b(v) \le n - 1$.
2. $b(K_n) = \lceil \log n \rceil$.
3. $b(S_n) = n - 1$.
4. For every graph $G = (V, E)$ and vertex $v \in V, b(v, G) \ge$ Diam$(v)$.

Note that Fact 2.5 holds in the open-path model as well, except of course for claim 4; we cannot argue that $b_{\text{op}}(G) \ge \text{Diam}(G)$.

## 3. The minimum-weight cover problem.

The MWC problem is a basic tool we use in our approximation algorithms for MBT. In this section we define the problem and provide a solution for it.

### 3.1. The problem.

In order to describe the MWC problem we need some preliminary definitions. Let $G = (V_1, V_2, A, \omega)$ be a bipartite graph with bipartition

[1] All logs in this paper are taken to base 2.

$(V_1, V_2)$, edge set $A$, a weight function $\omega : A \mapsto \mathbb{Z}^+$ on the edges, and no isolated vertices. A feasible solution to the MWC problem is a *control* function $F : V_2 \rightarrow V_1$, where $F(v_2) = v_1$ implies $(v_1, v_2) \in A$. Each vertex $v_1 \in V_1$ is called a *server,* and each vertex $v_2 \in V_2$ is called a *customer.* If $F(v_2) = v_1$ we say that $v_1$ *controls* (or *dominates*) $v_2$.

We adopt the following notational convention.

DEFINITION 3.1. *Consider a bipartite graph $G = (V_1, V_2, A, \omega)$ as above and a control function F. For each server $v \in V_1$, we denote the clients dominated by v by $\mathcal{D}_1(v), \ldots, \mathcal{D}_k(v)$ and the edges connecting them to v by $e_i^v = (v, \mathcal{D}_i(v))$. Furthermore, we assume (without loss of generality) that these vertices are ordered so that $\omega(e_i^v) \geq \omega(e_{i+1}^v)$ for every i.*

*The* weight *of F is defined as*

$$\mathcal{W}(F) = \max_{v \in V_1} \{ \max_i \{ i + \omega(e_i^v) \} \}.$$

The MWC problem is now defined as follows. Given a bipartite graph $G = (V_1, V_2, A, \omega)$ as above, determine a control function $F : V_2 \rightarrow V_1$ whose weight $\mathcal{W}(F)$ is minimal. We call this function $F$ the *minimum control function* for $G$ (or just the minimal function).

It is important to note that in all the applications to the MWC problem given in this paper, the weights satisfy $\omega(e) \leq n$. Thus, in order to use this problem as a basic auxiliary tool for the study of MBT, a pseudopolynomial solution for it will suffice.

A special variant of the MWC problem arises when for each $v_2 \in V_2$ the weights of the edges entering $v_2$ are identical. In this case, we might as well associate the weight with $v_2$ itself. We call this variant of the problem the MVWC problem. If *all* the weights are identical (thus without loss of generality all the weights are 0), a solution only needs to minimize the maximal number of vertices dominated by a single vertex of $V_1$, i.e., minimize the size of the largest inverse image of $F$; hence, it is possible to use the modified weight function

$$\omega'(F) = \max_{v_1 \in V_1} |\{ v_2 \in V_2 : F(v_2) = v_1 \}|.$$

**3.2. An algorithm for MWC.** This subsection presents a pseudopolynomial solution to the MWC problem. The algorithm is based on a procedure FLOW, which given an instance $G = (V_1, V_2, A, \omega)$ and a positive integer $j$, checks if there exists a control function $F$ for $G$ of weight $\mathcal{W}(F) \leq j$. This procedure is then used to search for the minimum control function by going over the possible $j$ values.

The solution method employed by Procedure FLOW for this problem involves flow techniques. Specifically, the procedure constructs a modified *flow graph* $\hat{G}_j$ based on $G$ and $j$, with the property that $G$ has a control function $F$ with weight $\mathcal{W}(F) \leq j$ iff it is possible to push $|V_2|$ units of flow from the source to the sink on $\hat{G}_j$.

The graph $G$ is modified by Procedure FLOW into $\hat{G}_j$ as follows. Create a source vertex $s$ and a sink vertex $t$. Notice that for the function $F$ to be of weight $j$, a server $v$ cannot dominate a customer $u$ such that $\omega(v, u) \geq j$. Assume that $\omega_v$ is the maximal weight that is less than or equal to $j - 1$ of an edge incident to $v \in V_1$. Duplicate $v$ into $\omega_v + 1$ different copies and arrange the copies in an arbitrary order $v_1, \ldots, v_{\omega_v+1}$. For $v_1$, the first copy of $v$, create a directed edge $(s, v_1)$ with capacity $j - \omega_v$ and a directed edge $(v_1, u)$ with capacity 1, from $v_1$ to every customer $u \in V_2$ such that $(v, u) \in A$. For $v_i$ the $i$th copy of $v, i \geq 2$, create a directed edge $(s, v_i)$ with capacity 1

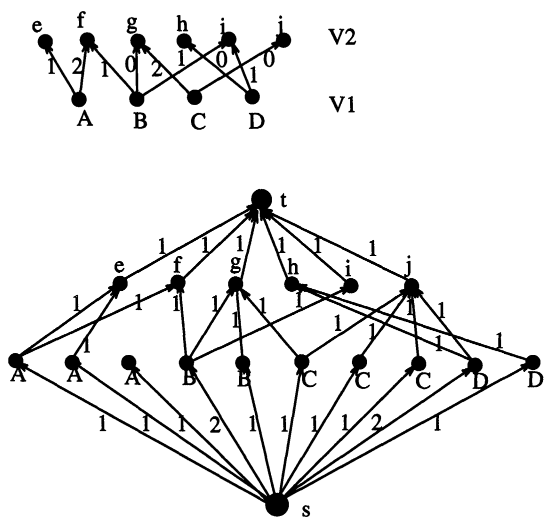


FIG. 1. *A bipartite graph $G$ with its weights, and the corresponding flow graph $\hat{G}_3$.*

and a directed edge $(v_i, u)$ with capacity 1 to all the customers $u$ such that $(v, u) \in A$ and $\omega(v, u) \leq \omega_v - i + 1$. Finally create for each customer $u \in V_2$ a directed edge $(u, t)$ with capacity 1. See Fig. 1 for an example of a graph $G$ and the associated flow graph $\hat{G}_3$.

PROCEDURE FLOW
Input: a graph $G = (V, E)$ and an integer $j$.

1. Construct the flow graph $\hat{G}_j$.
2. Compute the maximal flow on $\hat{G}_j$ from $s$ to $t$.

Since there are exactly $|V_2|$ edges entering $t$ and each of them is of capacity 1, the maximal flow from $s$ to $t$ cannot exceed $|V_2|$. We now claim the following lemma.

LEMMA 3.2. *Consider an MWC problem on a given graph $G$ and the corresponding flow graph $\hat{G}_j$ constructed by procedure* FLOW *for some integer $j$. Then the maximal flow from $s$ to $t$ on $\hat{G}_j$ is $|V_2|$ iff there exists a control function $F$ for $G$ such that $\mathcal{W}(F) \leq j$. Furthermore, the required control function $F$ for $G$ can be computed efficiently from the flow assignment for $\hat{G}_j$.*

*Proof.* For the first direction, we assume that there exists an integral flow function for $\hat{G}_j$ with $|V_2|$ flow units entering $t$ and show how to (efficiently) construct the required control function $F$.

Let us note that since each edge pointing from a vertex $v_2 \in V_2$ to $t$ has capacity 1 and all the edges entering $v_2$ have capacity 1 as well, only a single edge entering $v_2$ may carry positive flow of one unit. Hence since the total flow is $|V_2|$, one such edge must exist for each vertex $v_2 \in V_2$. Consequently, define $F(v_2) = v_1$ such that $v_1$ is the vertex for which the flow through $(v_1, v_2)$ is 1.

We need to show that this definition of $F$ meets the requirement, namely, $\mathcal{W}(F) \leq j$. This is shown as follows. Note that since the total flow entering all the copies of a vertex $v \in V_1$ does not exceed $j$, each server of $V_1$ may dominate no more than $j$ customers in $V_2$. For every server $v \in V_1$ and integer $m$, denote the number of $v$'s customers $\mathcal{D}_l(v)$ such that $\omega(e_l^v) \geq m$ by $r_m(v)$; alternatively, recalling that the edges

$e_l^v$ are ordered in nonincreasing weight order, we have

$$r_m(v) = \max\{l : \omega(e_l^v) \geq m\}.$$

We argue the following.

CLAIM 3.3. $r_m(v) \leq j - m$.

*Proof.* Consider first the value $m = \omega_v$. Since only the first copy of $v$ can dominate vertices $u$ such that $\omega(v,u) = \omega_v$, $v$ does not dominate more than $j - \omega_v$ such vertices.

Now consider $m < \omega_v$. Note that only the copies number $2, 3, \ldots, \omega_v - m + 1$ can aid in increasing the value of $r_m(v)$, each by at most 1. Thus $r_m(v) \leq j - \omega_v + (\omega_v - m + 1 - 2) + 1 = j - m$. □

COROLLARY 3.4. *Every* $1 \leq i \leq k$ *satisfies* $i \leq j - \omega(e_i^v)$.

*Proof.* Let $m = \omega(e_i^v)$. By the last claim, $r_m(v) \leq j - m$, so it remains to show that $i \leq r_m(v)$. This follows readily from the definition of $r_m(v)$. □

The remainder of the proof of this direction follows in a straightforward way from Definition 3.1: since by the last corollary $\max_i\{i + \omega(e_i^v)\} \leq j$ for every $j$, it follows that $\mathcal{W}(F) \leq j$ as well.

To prove the other direction, assume that there exists a control function $F$ for $G$ such that $\mathcal{W}(F) \leq j$. Thus each server dominates no more than $j$ customers, and furthermore $v$ does not dominate more than $j - i$ customers whose corresponding weights are $i$ or larger. We now use $F$ to define a flow function for the flow graph $\hat{G}_j$ as follows.

Consider a server $v$. As before, order its customers $\mathcal{D}_1(v), \ldots, \mathcal{D}_k(v)$ by nonincreasing weights of their edges. Augment the flow through $v_1$, the first copy of $v$, by $j - \omega_v$, adding a flow of 1 from $v_1$ to each of $\mathcal{D}_1(v), \ldots, \mathcal{D}_{j-\omega_v}(v)$. The weight of the next client of $v$, namely, $\mathcal{D}_{j-\omega_v+1}(v)$, is smaller than $\omega_v$, and therefore there is a directed edge from the second copy, $v_2$, to that vertex. Thus it is still possible to augment the flow by 1 through $v_2$. In a similar way, it is possible to continue this process and augment the flow by $k$ through all of $v$'s clients, $\{\mathcal{D}_1(v), \ldots, \mathcal{D}_k(v)\}$. Since this is true for any server, it follows that in total, a maximal flow of $|V_2|$ can be attained. □

The minimum weight control function itself can now be found by using procedure FLOW within a binary search on the possible range of $j$ values. Formally, we do the following.

```
PROCEDURE MWC
1. Start with $j_1 \leftarrow \min_e\{\omega(e)\} + 1$ and $j_2 \leftarrow \max_e\{\omega(e)\} + |V_2|$.
2. repeat
   (a) $j \leftarrow \lceil \frac{j_1+j_2}{2} \rceil$.
   (b) Apply Procedure FLOW to $G$ with $j$.
   (c) If the maximal flow is $|V_2|$
       /* hence there exists a control function $F$ of weight $j$ */
       then set $j_1 \leftarrow j$, else set $j_2 \leftarrow j$.
     until $j = j_2 \leq j_1 + 1$.
3. Return the minimum control function $F$ corresponding to the maximal flow
   computed on $\hat{G}_{j_1}$.
```

Clearly, when this process terminates, we are left with a minimum control function $F$, whose weight is $\mathcal{W}(F) = j_1$ for the value of $j_1$ upon termination.

The flow computation is performed for at most a polynomial number of times. Note, however, that a vertex may be duplicated in a number of copies that can equal the maximal number in the input; hence the solution is not strongly polynomial in the input. To summarize, we have established the following.

THEOREM 3.5. *There exists a pseudopolynomial algorithm for solving the MWC problem.*

Since MVWC is a special case of MWC, we can deduce the following corollary.

COROLLARY 3.6. *There exists a pseudopolynomial algorithm MVWC for solving the MVWC problem.*

**3.3. The bipartite edge scheduling problem.** Before embarking on our main task of approximating broadcast problems, let us first try to demonstrate the way in which the MWC algorithm can be used for this purpose. To do that, we introduce a model problem called the BES problem, which will serve as an illustrative example for the usefulness of MWC.

The BES problem is defined as follows. Assume that we are given a bipartite graph $G = (V_1, V_2, A)$ where the vertices of $V_1$ know an `initiation` message that must be broadcast to the vertices of $V_2$. That is, the initial set of informed vertices is $V_1$. Again, call each vertex in $V_1$ a *server* and each vertex in $V_2$ a *customer.*

Further suppose that each customer $v_2 \in V_2$ has a task $t_{v_2}$ to perform. The customer must first receive the `initiation` message before it can start performing its task. Moreover, the *length* of the task $t_{v_2}$ (i.e., the time it takes $v_2$ to complete it) depends upon the *source* of the `initiation` message, namely, which vertex of $V_1$ transmits the message to $v_2$. Formally, for each edge $e = (v_1, v_2) \in A$ there is a weight $\omega(e) \in \mathbb{Z}^+$, such that if $v_2$ receives the `initiation` message from $v_1$, then it takes $\omega(e)$ time units for $v_2$ to complete the task $t_{v_2}$, starting from the arrival time of the message.

The BES problem is to minimize the completion time of the entire process, namely, the time by which every vertex in $V_2$ completes its job. The *bipartite vertex scheduling* (BVS) problem is the variant of BES in which every vertex $v' \in V_2$ has entering edge weights that are identical (and hence can be thought of as vertex weights, $\omega(v')$).

To see the relationship between the BES problem and the previously discussed MWC problem, note that MWC is in fact the main component in solving the problem, since once a control function $F$ is defined for the graph $G$, it is intuitively most efficient for each server $v$ to send the `initiation` message along its edges in nonincreasing order of weights. It is therefore easy to see that the following (pseudopolynomial) procedure will be optimal for BES.

```
ALGORITHM BES
1. Apply Procedure MWC to compute a minimal control function $F$ for $G$.
2. Every server $v \in V_1$ sends the initiation message to its clients $\mathcal{D}_1(v), \ldots,$
   $\mathcal{D}_k(v)$ in this order.
       /* recall that clients are ordered by nonincreasing edge weights */
3. Each customer $v' \in V_2$ starts performing its task immediately after it is in-
   formed.
```

*Fact* 3.7. Algorithm BES optimally schedules any BES instance.

Since BVS is a special case of BES, we can deduce the following corollary.

COROLLARY 3.8. *There exists an optimal pseudopolynomial algorithm for BVS.*

*Example.* In order to demonstrate the relevance of the BES for broadcasting, let us consider the following restricted variant of SMBT. Let $V_0$ be the base set of an SMBT instance. Suppose that when extracting $V_0$ from the graph, the remainder of the graph becomes disconnected and breaks into $k$ (connected) clusters $C_1, \ldots, C_k$.

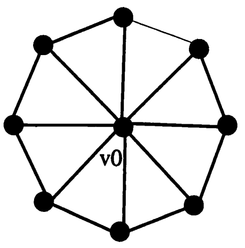


FIG. 2. *The wheel of nine vertices.*

Furthermore, suppose that every cluster $C_i$ contains a *representative* vertex $v_i$, such that every vertex $v \in V_0$ is either connected to $v_i$ or not connected to any vertex of $C_i$. Finally, assume that we can optimally compute the broadcast times $b(v_i, C_i)$ for every cluster $C_i$ (e.g., when the clusters $C_i$ are, for example, trees or grids or of logarithmic size).

Intuitively, we can view this variant of the broadcasting problem as a special case of the BVS problem, as follows. Form the control graph $D_{V_0,G}$ of $V_0$ in $G$ as in Definition 2.2. Define the weight $\omega(C_i)$ on a vertex $C_i \in V_2$ as the time required by the representative vertex, $v_i$, for broadcast in $C_i$. Now apply the BVS algorithm to this graph, taking $V_1$ to be the base set and $V_2$ to be the collection of clusters $\{C_1, \ldots, C_k\}$ and regarding the *task* of each vertex $C_i$ in $V_2$ as performing broadcast in the cluster by the representative vertex $v_i$. Since in each of the clusters only the representative vertex is connected to the "outside world," the vertices outside the cluster cannot aid in this internal broadcast process; hence, the vertex weights on $V_2$ are defined appropriately.

It is straightforward to show that the solution to the BVS problem yields exactly the minimum broadcast time from $V_0$. Further, note that since the vertex weights in this problem represent broadcast times in a cluster of the graph, it follows from Fact 2.5 that they are no larger than $n - 1$, and hence the BVS instance can be solved polynomially.

## 4. Approximating broadcast in general graphs.

In this section we give an approximation scheme for broadcasting in general graphs, both in the telephone model and in the open-path model. We begin by motivating the need for approximation schemes for broadcasting.

### 4.1. The heuristic approach.

In this subsection we demonstrate the fact that the natural heuristics proposed in [SW84] might be inadequate in some simple cases. The example considered is a *wheel*, i.e., a cycle of $n-1$ vertices numbered $1, \ldots, n-1$, arranged by increasing order of indices, with an extra vertex $v_0$ connected to all the vertices in the cycle (Fig. 2). Let us first give tight bounds on the minimum broadcast time from $v_0$.

LEMMA 4.1. *In the n-vertex wheel, $b(v_0) = \Theta(\sqrt{n})$.*

*Proof.* First let us show that $b(v_0) = \Omega(\sqrt{n})$. Assume that the broadcast takes $k$ time units. The vertex $v_0$ can inform up to $k$ different vertices. Therefore there is a vertex $w$ whose distance on the ring from the vertices directly informed by $v_0$ is at least $\lfloor \lceil (n-1)/k \rceil /2 \rfloor$. Thus the time for informing $w$ is at least $\lfloor \lceil (n-1)/k \rceil /2 \rfloor + 1$. The minimum broadcast time is achieved when $\lfloor \lceil (n-1)/k \rceil /2 \rfloor + 1 = k$, in which case $k \geq \sqrt{(n-1)/2}$, yielding the desired lower bound on the broadcast time.

To prove that $b(v_0) = O(\sqrt{n})$, note that if $v_0$ informs all vertices whose index is congruent to 1 mod$\lceil\sqrt{n-1}\rceil$ and then these cycle vertices inform the rest of the vertices, the total time for broadcasting is no more than $3 \cdot \lceil\sqrt{n-1}\rceil/2 + 1$ time units. □

Now consider the following three heuristics suggested in [SW84] for MBT. The first heuristic is based on defining, for a set of vertices $V' \subseteq V$,

$$D_G(V') = \sum_{v \in V'} d(v).$$

At each round $i$, select from all possible maximum matchings between the set of informed vertices and the set of uninformed vertices, a matching satisfying that the set $V'' \subseteq V \setminus V_i$ of "receiving" end vertices in the matching has maximal $D_G(V'')$. Let us describe a possible broadcast scenario. At the first round, $v_0$ delivers the message to the vertex 1. At round 2, $v_0$ calls $n-1$ and 1 calls 2. It is easy to see that starting from the third round one can choose a maximal matching that enlarges the set of informed vertices by 3 at each round. For example, at the third round $v_0$ may call $n-3, n-1$ calls $n-2$, 2 calls 3, and so on. Note that the above scenario conforms with the given heuristic, since the degree of all the vertices in the wheel (except $v_0$) is 3. Thus using this heuristic, it might take $\Omega(n)$ time units to complete the broadcasting.

A second heuristic approach suggested in [SW84] is the following. Define the *eccentricity* of a set $V' \subset V$ to be

$$\mathrm{Dist}(V') = \sum_{v \in V'} \mathrm{Diam}(v).$$

Choose, among all the possible maximal matchings, a matching for which the eccentricity of the set of newly informed vertices is maximal. In a way similar to that above, it is easy to see that there are cases in which this approach leads to a broadcast time of $\Omega(n)$. The last heuristic suggested in [SW84] is a combination of the former two, and it, too, may lead to an $\Omega(n)$ broadcast scheme.

It follows that, for this example, all of these heuristics may yield broadcast times that are $\Omega(\sqrt{n})$ times worse than optimal. We do not know whether this ratio is guaranteed by any of the three heuristics.

**4.2. Approximating MBT.** In this subsection we consider approximation schemes for broadcasting in general graphs. By Fact 2.5, the scheme that chooses an arbitrary broadcast tree is at worst an $(n-1)/\lceil\log n\rceil$ approximation scheme. We improve upon this approximation ratio and present an algorithm that guarantees an $O(\sqrt{n})$-*additive* ratio.

The method used for the approximation is based on dividing the set of vertices into clusters of size $\lceil\sqrt{n}\rceil$ and broadcasting separately on those clusters. Let us start by describing the various tools used by the algorithm. At the heart of our scheme is the following decomposition lemma.

LEMMA 4.2. *The set of vertices of any graph $G = (V, E)$ can be (polynomially) decomposed into two sets of clusters $\mathcal{A}$ and $\mathcal{B}$, such that $|\mathcal{A}| \leq \sqrt{n}$, the clusters in $\mathcal{A} \cup \mathcal{B}$ are pairwise disjoint, $(\bigcup \mathcal{A}) \cup (\bigcup \mathcal{B}) = V$, the size of each cluster $C' \in \mathcal{A}$ is $|C'| = \lceil\sqrt{n}\rceil$, the size of each cluster $C' \in \mathcal{B}$ is bounded by $|C'| \leq \sqrt{n}$, and the clusters in $\mathcal{B}$ are pairwise independent.*

*Proof.* The proof the lemma follows by direct construction. Let us next present the decomposition procedure. The algorithm maintains three different sets of clusters $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$.

```
PROCEDURE DECOMPOSE
1. At the start $\mathcal{C} \leftarrow \{V\}, \mathcal{A} \leftarrow \emptyset, \mathcal{B} \leftarrow \emptyset$.
2. repeat
   (a) Choose a cluster $C$ in $\mathcal{C}$ and an arbitrary (connected) subcluster $C'$ of $C$
       such that $|C'| = \lceil\sqrt{n}\rceil$.
   (b) Remove $C'$ from $C$, and set $\mathcal{A} \leftarrow \mathcal{A} \cup \{C'\}$.
       /* Now $C \setminus C'$ is composed of several independent clusters. */
       Add the clusters $\{C'' : C'' \subseteq C \setminus C', |C''| = \lceil\sqrt{n}\rceil\}$ to $\mathcal{A}$.
       Add the clusters $\{C'' : C'' \subseteq C \setminus C', |C''| > \lceil\sqrt{n}\rceil\}$ to $\mathcal{C}$.
       Add the remaining clusters in $C \setminus C'$ to $\mathcal{B}$.
   until $\mathcal{C} = \emptyset$.
```

It is clear from the construction that all the clusters in $\mathcal{A}$ have exactly $\lceil\sqrt{n}\rceil$ vertices. Thus the number of clusters in $\mathcal{A}$ cannot exceed $\sqrt{n}$. It is also clear that the number of vertices in each cluster in $\mathcal{B}$ is no more than $\sqrt{n}$.

To prove that the clusters in $\mathcal{B}$ are independent, it is sufficient to claim that at each stage, all the clusters of $\mathcal{B} \cup \mathcal{C}$ are such. The proof is by induction on the stage number. At the basis $\mathcal{B} = \mathcal{C} = \emptyset$ and the claim follows vacuously. Now assume that after stage $i$ all the clusters in $\mathcal{C}$ and $\mathcal{B}$ are pairwise independent. In the next stage we exclude a subset of vertices from some cluster $C \in \mathcal{C}$; thus the remaining vertices decompose into independent clusters. The clusters added to $\mathcal{B}$ and $\mathcal{C}$ are a subset of those clusters; thus the claim follows by this fact and the induction hypothesis. □

Note that the construction of Procedure DECOMPOSE in the proof of Lemma 4.2 allows us to find such a decomposition in $O(E \cdot \sqrt{V})$ time, since checking which are the newly formed clusters at each stage takes $O(E)$ time.

Our next tool exploits the use of a minimum control function for broadcast. Let $G = (V, E)$ be a graph and $V' \subset V$ subset of the vertices. Form the control graph of $V'$ in $G, D_{V',G} = (V_1, V_2, A)$, as in Definition 2.2. Let the weight of each edge be 0. In this scenario we claim the following.

LEMMA 4.3. *Let $F$ be a minimum control function for $D_{V',G}$. Then $\mathcal{W}(F) \leq b(V', G)$.*

*Proof.* Consider an optimal communication scheme $S$ from $V'$ on $G$. For every cluster $C_i \in V_2$, choose *one* of the vertices that is among the *first* to transmit the message to a vertex in $C_i$. Since the clusters $C_i$ are pairwise independent, such a vertex is in $V'$ for each $i$. The above determines a function $F'$ from $V_2$ to $V_1$.

Consider an arbitrary vertex $v \in V'$. If $v$ dominates $j$ clusters, then there is a cluster $C_i \in V_2$ such that $v$ transmits the message to $C_i$ in the $j$th round (or later). By the way $F'$ was chosen, $j \leq b(V', G)$. Thus by definition, $\mathcal{W}(F') \leq b(V', G)$. However, the weight of $F$ satisfies $\mathcal{W}(F) \leq \mathcal{W}(F')$. The proof follows. □

Our final tool involves broadcasting on a tree. Recall that given a tree $T = (V_1, E_1)$ and a vertex $v \in V_1$ it is easy to compute the optimal scheme for broadcasting on $T$ from $v$ [SCH81]. Let us call the optimal scheme for broadcasting in a tree the *OT scheme.*

In the next lemma we use a shortest paths tree $\mathrm{SPT}(v, S)$ rooted at a vertex $v$ and leading to a set $S$ of vertices (see Definition 2.1). Note that it is easy to construct such a tree in time polynomial in $|E|$ using a shortest path tree algorithm; simply construct a shortest path tree $T$ spanning all the graph vertices, and iteratively exclude from it each leaf not belonging to $S$, until no such leaf exists.

LEMMA 4.4. *Transmitting a message from a vertex $v$ to a subset $V' \subseteq V, |V'| = l$ of the graph, can be performed in no more than $l - 1 + \mathrm{Diam}(v)$ time units.*

*Proof.* Construct an arbitrary tree $\mathrm{SPT}(v, V')$ rooted at $v$ and leading to the $l$ vertices of $V'$. Use the OT scheme to broadcast the message to all the members of the tree. Consider any leaf $u$. We would like to show that $u$ gets the message within the specified time bounds. This is done by "charging" each time unit elapsing until $u$ gets the message to a distinct vertex of the tree and then bounding the number of charges.

Consider the situation immediately after an ancestor $v'$ of $u$ receives the message. The vertex $v'$ is currently the lowest ancestor of $u$ that knows the message. Thereafter $v'$ starts delivering the message to its children. When $v'$ delivers the message to a child whose subtree $T'$ does not include $u$, choose arbitrarily a leaf in $T'$ and charge this time unit to the leaf. When $v'$ delivers the message to the ancestor of $u$, charge this time unit to $v'$.

Note that at every time unit we charge a single vertex, on account of $u$, and thus the total number of units charged is exactly the time before the message reaches $u$. Also note that no leaf is charged twice and that $u$ is not charged. Finally note that every ancestor of $u$ (except $u$ itself) is charged once. Thus the time it takes the message to reach $u$ is bounded by $\mathrm{Diam}(v)$ plus the number of leaves in $T$ beside $u$. Since each leaf is a member of $V'$, the proof follows. □

We are now ready to combine the three tools discussed in the above lemmas into an algorithm, named APPROX_MBT, for approximate broadcast on general graphs.

ALGORITHM APPROX_MBT

Input: a graph $G = (V, E)$ and a distinguished vertex $v_0 \in V$.

1. Decompose the vertices of $V$ into two sets of clusters $\mathcal{A}$ and $\mathcal{B}$ using Procedure DECOMPOSE of Lemma 4.2.
2. Choose for each cluster $C$ in $\mathcal{A}$ a single representative vertex $v_C$. Let $R$ denote the set of representatives, $R = \{v_C | C \in \mathcal{A}\}$.
3. Transmit the message from $v_0$ to all the vertices of $R$ by choosing an arbitrary tree $\mathrm{SPT}(v_0, R)$ leading from $v$ to $R$ and applying the *OT* scheme to the tree.
4. Choose for each cluster $C \in \mathcal{A}$ an arbitrary spanning tree rooted at its representative $v_C$, and broadcast (in parallel) in the clusters of $\mathcal{A}$ according to the OT scheme.
5. Construct the bipartite control graph $D_{V_0,G}$ where $V_0 = (\bigcup \mathcal{A}) \cup \{v_0\}$. Compute a minimum control function $F$ on $D_{V_0,G}$ using Procedure MWC. Assume that a vertex $v'$ dominates clusters $C_1, \ldots, C_k \in \mathcal{B}$. Choose for each $C_i$ an arbitrary vertex $v_i \in C_i$ connected to $v'$, and deliver the message from $v'$ to $v_1, \ldots v_k$ (in arbitrary order). This is done in parallel for all the dominating vertices of $V_0$.
6. Choose for each cluster in $\mathcal{B}$ an arbitrary spanning tree rooted at an informed vertex, and transmit the message in parallel to all the vertices in the clusters of $\mathcal{B}$ using the OT scheme in each cluster.

THEOREM 4.5. *The broadcast time of Algorithm* APPROX_MBT *from a vertex* $v_0$ *in a graph* $G$ *is bounded by* $3 \cdot \sqrt{n} + \mathrm{Diam}(v_0) + b(v_0)$ *time units.*

*Proof.* By Lemmas 4.4 and 4.2 the time it takes to complete stage 2 is bounded by $\sqrt{n} - 1 + \mathrm{Diam}(v)$. The fact that each cluster in $\mathcal{A}$ has exactly $\lceil \sqrt{n} \rceil$ vertices and Fact 2.5 imply that stage 3 takes no more than $\sqrt{n}$ time units. By Lemma 4.3, $\mathcal{W}(F) \leq b((\bigcup \mathcal{A}) \cup \{v_0\}) \leq b(v_0)$; thus stage 4 takes no more than $b(v_0)$ time units. Finally, the fact that the clusters in $\mathcal{B}$ are of size no larger than $\sqrt{n}$ implies that stage 5 takes no more than $\sqrt{n} - 1$ time units. □

The ratio guaranteed by the theorem is $O(\sqrt{n})$-additive. Consequently, whenever the broadcast time of a network is $\Omega(\sqrt{n})$, e.g., in the wheel of $n$ vertices, the scheme of Algorithm APPROX_MBT is a constant approximation scheme. For example, for each network whose diameter is at least $\sqrt{n}$, the above is a 5 approximation scheme. However, in the general case, the optimal broadcasting scheme may achieve time that is close to $\lceil \log n \rceil$. Thus, in the general case our method is a $(3 \cdot \sqrt{n}/\lceil \log n \rceil + 2)$-approximation scheme and also an $O(\sqrt{n}/\mathrm{Diam}(G))$-approximation scheme. In order to improve upon this it may be necessary to achieve, say, a good approximation scheme for networks of "small" diameter.

**4.3. Broadcasting in the open-path model.** Let us turn to the open-path communication model. Algorithm APPROX_MBT can be generalized to give a good approximation scheme for the open-path broadcasting problem. It is easy to see that Lemma 4.3 holds even in the open-path model.

LEMMA 4.6. *Let $T$ be a tree rooted at $v$, with up to $k$ leaves. Then it is possible to broadcast a message from the root $v$ to all the vertices of the tree in the open-path model, in no more than $2 \cdot k + \log n$ time units.*

*Proof.* We first recall the following fact, proven in [Far80].

*Fact* 4.7 [Far80]. Broadcasting in the open model on a path of $m$ vertices can be completed in $\lceil \log m \rceil$ time units.

To prove Lemma 4.6 we give a two-stage procedure. In the first stage we proceed in the following recursive fashion. As soon as a vertex $v'$ is informed, it checks if the subtree $T_{v'}$ rooted at it contains a vertex of degree at least 3, i.e., with at least two children. If no such vertex exists (i.e., the subtree $T_{v'}$ is a path), then $v'$ does nothing. Otherwise, assume that the highest such vertex in the tree rooted at $v'$ is $v''$. That is, the subtree $T_{v'}$ is composed of a path connecting $v'$ to $v''$, plus the subtree $T_{v''}$ (note that possibly $v'' = v'$). Then $v'$ makes a long-distance call to all the *children* of $v''$, in arbitrary order.

It is easy to see that when this first stage is finished, the tree can be decomposed into a union of disjoint paths where for each path, one end vertex knows the message. During the second stage, each informed end vertex of each path informs the rest of the vertices in the path as in Fact 4.7.

In analyzing the delays occurring before a message reaches a leaf $u$, we separate our analysis to the first and second stages. This first stage is handled by a charging rule somewhat similar to that used in the proof of Lemma 4.4. Note that at each time step $t$, there is a unique ancestor $\mathrm{low}(u,t)$ of $u$ that is responsible for it, namely, the lowest ancestor currently holding the message. At time step $t$, $\mathrm{low}(u,t)$ sends the message to some child $v''$ of a vertex $v'$ with at least two children. If $u$ does not belong to the subtree $T_{v''}$, charge a delay to an arbitrary leaf in $T_{v''}$. When the message is sent to a child $v''$ of a vertex $v'$ such that $v''$ is a ancestor of $u$, charge the delay to $v'$. Note that only leaves and vertices with degree at least 3 are charged, and none of them is charged more than once. The number of vertices of degree 3 or higher is no more than $k - 2$; hence, the total delay in stage 1 is bounded by $2 \cdot k - 2$.

In the second stage the broadcast takes place along vertex disjoint paths, each with no more than $n$ vertices. Hence by Fact 4.7 this stage can be completed in $\lceil \log n \rceil$ time units. In total, the communication delay is bounded by $2 \cdot k - 2 + \lceil \log n \rceil$. □

This discussion motivates the following approach for approximating OMBT. First we define sets of representatives, $\{R_1, \ldots, R_f\}$, where $R_1 = V, |R_f| \leq \log n$, and $f \leq \log n / \log\log n$. To each set $R_j$ and vertex $v \in R_j$ there is a corresponding tree $T_j^v$, containing at least $\lceil \log n \rceil$ vertices of $R_{j-1}$. The trees corresponding to different vertices in $R_j$ are *vertex disjoint.*

The main algorithm operates in $f$ stages. The first stage informs the vertex set $R_f$. The algorithm then proceeds to inform the sets $R_j$ in reverse order of indices, i.e., at the end of stage $i$, the message is known by the set $R_{f-i+1}$, and the goal of the next stage is for $R_{f-i+1}$ to inform $R_{f-i}$.

We next present Procedure CHOOSE_REP, whose task is to choose the sets $R_i$ of representatives and the corresponding trees $T_i^v, v \in R_i$. After that, we give the main algorithm that uses Procedure CHOOSE_REP to approximate OMBT.

Throughout the execution of Procedure CHOOSE_REP we extract trees from $G$.

```
PROCEDURE CHOOSE_REP
Input: A graph $G = (V, E)$ and a distinguished vertex $v_0 \in V$.
  1. $R_1 \leftarrow V, i \leftarrow 1$.
  2. repeat
     (a) $\mathcal{C} \leftarrow \{V\}, R_{i+1} \leftarrow \emptyset$.
     (b) while $\mathcal{C} \neq \emptyset$ do:
         (i) Choose cluster $A \in \mathcal{C}$. Select $\lceil \log n \rceil$ vertices in $A \cap R_i$ arbitrarily, except
             that if $v_0 \in A$, take $v_0$ as one of them. Let the chosen vertices be
             $v_1, \ldots, v_{\lceil \log n \rceil}$, such that we set $v_1 = v_0$ if $v_0 \in A$. Select in $A$ a subtree
             $T_i^{v_1}$ leading from $v_1$ to $\{v_2, \ldots, v_{\lceil \log n \rceil}\}$.
         (ii) Extract $T_i^{v_1}$ from $A$.
         (iii) Set $\mathcal{C} \leftarrow \mathcal{C} \cup \{B : B$ is a connected component of $A \setminus T^{v_1}, |B \cap R_i| >$
             $\lceil \log n \rceil\}$.
        end-while
     (c) For every tree $T_i^{v_1}$ obtained in stage (b), add its root $v_1$ to $R_{i+1}$.
     (d) $i \leftarrow i + 1$.
     until $|R_i| \leq \lceil \log n \rceil$.
```

Let us first state the following properties of Procedure CHOOSE_REP. The proof follows directly from the algorithm.

CLAIM 4.8. *Let the number of stages in Procedure* CHOOSE_REP *be $f$. Then*
1. $v_0 \in R_f$,
2. $|R_i| \leq n/\log^i n$, *and*
3. $f \leq (\log n/\log\log n)$.

We now proceed to define the main algorithm. Throughout the algorithm we maintain a set $R$ of informed vertices that equals $R_j$ for some $j$. The point is that $j$ decreases by one in each iteration, thus at the end $R = R_1 = V$.

```
ALGORITHM APPROX_OMBT
Input: A graph $G = (V, E)$ and a distinguished vertex $v_0 \in V$.
  1. Apply Procedure CHOOSE_REP on $G$ and $v_0$. Assume that the sets of repre-
     sentatives are $\{R_1, \ldots, R_f\}$.
  2. Choose an arbitrary tree leading from $v_0$ to the other vertices of $R_f$, and inform
     all the vertices in $R_f$ using the scheme suggested in the proof of Lemma 4.6.
  3. $R \leftarrow R_f, i \leftarrow f$.
  4. repeat
     (a) Each vertex $u \in R_i$ informs (in parallel) all the vertices in $T_i^u$ using the
         scheme suggested in the proof of Lemma 4.6.
     (b) Let $G_i' = G \setminus \bigcup_{u \in R_i} T_i^u$. Let $C_1^i, \ldots, C_s^i$ denote the clusters in the graph
         induced by $G_i'$.
```

```
(c) The vertices $\bigcup T_i^u$ inform a vertex $v_j$ in $C_j^i$, for each $1 \le j \le s$, using a
    minimum control function computed by Procedure MWC, as in step 5 of
    Algorithm APPROX_MBT.
(d) Choose for each $j$, a tree $TL_j$ leading from $v_j$ to the vertices in $R_{i-1} \cap C_j^i$.
(e) The vertices $v_j$ inform the vertices of $TL_j$ using the scheme of Lemma 4.6.
(f) $R \leftarrow R_{i-1}, i \leftarrow i - 1$.
until $i = 1$.
```

It is clear from the algorithm that when stage 4(e) of Algorithm APPROX_OMBT is completed, all the vertices of $R_{i-1}$ know the message. It follows that at the end all the vertices are informed.

THEOREM 4.9. *Algorithm* APPROX_OMBT *is a $O(\log n/\log\log n)$ approximation scheme for OMBT.*

*Proof.* By the definition of Procedure CHOOSE_REP, $|R_f| \le \log n$. Thus it follows by this fact and Lemma 4.6 that step 2 of Algorithm APPROX_OMBT takes $O(\log n)$ time units. Let us now analyze the communication time of stages 4(a) to 4(e) for a fixed $i$. Since for every $u \in R_i$, every leaf in $T_i^u$ belongs to $R_{i-1}$ and $|T_i^u \cap R_{i-1}| = \lceil \log n \rceil$, it follows that the number of leaves in $T_i^u$ is bounded by $\lceil \log n \rceil$; thus, by Lemma 4.6 the communication time of informing the vertices of $T_i^u$ in step 4(a) is bounded by $O(\log n)$ time units. In step 4(c) the vertices in $\bigcup T_i^u$ inform a vertex of each clusters $C_j^i$. It follows by a similar argument to Lemma 4.3 that the number of time units is bounded by $b_{\text{op}}(v_0)$. Finally, it follows from step (b) of Procedure CHOOSE_REP that for every $j, C_j^i \cap R_{i-1} \le \log n$; thus, by Lemma 4.6, step 4(e) takes no more than $O(\log n)$ time units. Summarizing, for a fixed $i$, the communication complexity of the scheme is bounded by $O(b_{\text{op}}(v_0) + \log n) = O(b_{\text{op}}(v_0))$. By Claim 4.8, $f \le \log n/\log\log n$. Hence the desired result follows. □

Let us remark that we choose trees with $\lceil \log n \rceil$ leaves since this is the only lower bound known on the broadcasting time. If, however, it is known that for some particular family of input instances the broadcast time is bounded below by $n^{1/k}$ for some $k$ smaller than $\log n/\log\log n$, then it is possible to construct trees with $n^{1/k}$ leaves and get an $O(k)$-approximation scheme for $k < \log n/\log\log n$.

**4.4. Broadcasting on random graphs.** The method of Algorithm APPROX_OMBT can be used to deal with the MBT problem as well. However, at each stage, broadcasting in a tree $T$ may take $O(\log n + h(T))$ time units, where $h(T)$ is the height of $T$. Since the diameter of a subcluster of a graph $G$ may largely increase, this may not be a good approximation scheme in the worst case.

However, it is instructive to consider the behavior of Algorithm APPROX_OMBT on random inputs. Let us consider a random graph $G \in G_{n,p}$. The graph consists of $n$ vertices, where for each pair of vertices $v, w \in V$, the edge $(v, w) \in E$ is drawn with probability $p$, where $p$ is constant, $0 < p < 1$. It is well known that the diameter of each vertex-induced subcluster of $G_{p,n}$ is bounded by $O(\log n)$ with high probability [Bol85]. For such graphs the scheme of Algorithm APPROX_MBT yields only an $O(\sqrt{n}/\log n)$-approximation ratio. In contrast, Algorithm APPROX_OMBT is an $O(\log n/\log\log n)$-approximation scheme for random graphs with high probability.

COROLLARY 4.10. *There exists a polynomial algorithm that broadcasts on a random graph $G \in G_{n,p}$ in no more then $O(\log n/\log\log n) \cdot b(G)$ time units with high probability.*

**5. Separator-based strategies for broadcasting.** An important method for dealing with optimization problems on graphs is the *divide-and-conquer* approach

[AHU74]. The idea is to find a small set of vertices whose removal splits the graph into connected components of roughly equal size and then to solve the problem recursively by handling each of the components separately. Unfortunately, general graphs do not necessarily have small separators. However, some important families of graphs do. The notion of a separator is formalized in the following definition.

DEFINITION 5.1 *Let $\varphi(n)$ be a nondecreasing function, and let $y$ and $\rho$ be fixed numbers such that* $0 < \rho < 1$.

1. *A graph $G = (V, E)$* has a $\langle \rho, y \rangle$-separator *if there exists a set $S \subset V$ such that the removal of $S$ leaves no connected component of size greater than $\rho \cdot n$, and* $|S| \leq y$.
2. *A graph $G = (V, E)$ is* $\langle \rho, \varphi(n) \rangle$-separable *if every vertex-induced subgraph $G' \subset G$ of $n'$ vertices has a $\langle \rho, \varphi(n') \rangle$-separator.*

Given a $\langle \rho, \varphi(n) \rangle$-separable graph, denote the corresponding separator of every subgraph $G'$ by $\text{sep}(G')$. This section examines the idea of using the separability property of a graph in order to achieve fast approximation schemes for broadcasting.

## 5.1. Broadcasting schemes for separable graphs.

In order to develop a separator-based broadcasting scheme, we first need to generalize Lemma 4.3. Suppose that a graph $G$ contains a set $V_0$ of informed vertices. Denote the clusters created when extracting $V_0$ from the graph by $C_1, \ldots, C_k$. Choose for each $C_i$ an arbitrary nonempty subset $C_i' \subset C_i$. We can use the fact that in broadcasting it is necessary to inform the vertices of $C_i'$ to achieve a lower bound on the best possible time for broadcasting. As before, we use Procedure MWC developed in §3.2. Let us first define a MWC instance $B' = \text{MWC}(V_0, \{C_1', \ldots, C_k'\})$ as follows.

1. Form the control graph $D_{V_0,G}$ of $V_0$ in $G$.

2. Put weights on the edges as follows. For an arbitrary vertex $v \in V_0$ connected to a vertex in $C_i$, choose a vertex $v' \in C_i$ connected to $v$ that is closest to the set $C_i'$. Attach a weight $d^v_{C_i} \equiv \text{dist}(v', C_i')$ to the edge $(v, C_i)$.

(Note that in Lemma 4.3 the construction is similar, except that we choose as subset $C_i' = C_i$, for every $i$.) We make the following claim.

LEMMA 5.2. *If $F$ is a minimal control function for $B'$, then $\mathcal{W}(F) \leq b(V_0, G)$.*

*Proof.* First we argue the following technical claim, implicitly used also in [SCH81].

CLAIM 5.3. *Let $d_i, t_i \in \mathbb{Z}^+$, for $1 \leq i \leq k$, such that $d_k \leq d_{k-1} \leq \cdots \leq d_1$, and the $t_i$'s are all distinct. Then* $\max_i\{i + d_i\} \leq \max_i\{t_i + d_i\}$.

To any communication scheme from a base set $V_0$ such that $|V_0| > 1$, there is a corresponding spanning forest of $G$, where each tree in the forest is rooted at a vertex $v$ of $V_0$ and represents the set of edges that carried the message from $v$ (i.e., the nodes of the tree are those that received their copy of the message along a path originating at $v$). Assume that $S$ is an optimal communication scheme for broadcasting from the base set $V_0$. Denote the forest corresponding to $S$ by $\mathcal{F}$. Every vertex $v \in V_0$ that sends the message to some vertex in one of the $C_i$ clusters roots a tree $T_v \in \mathcal{F}$. Let us define a function $F' : \{C_1, \ldots, C_k\} \to V_0$ as follows. Pick a vertex $w_i \in C_i'$ that is among the first in $C_i'$ to receive the message (breaking ties arbitrarily). Suppose that $w \in T_v$; then set $F'(C_i) = v$. Now consider a vertex $v \in V_0$ that dominates a nonempty set of clusters. Say that $v$ controls $C_i$ (i.e., $F'(C_i) = v$), and assume that $u_i$ is the (unique) child of $v$ in $T_v$ that is an ancestor of $w_i$. Clearly, we can assume that $u_i \in C_i$.

Say that $v$ sends the message to $u_i$ at time $t_i$. The time that passes before $u_i$ can inform any vertex in $C_i'$ (specifically $w_i$) is at least $d(u_i, C_i)$, thus by the fact that $w_i$

is one of the first vertices in $C'_i$ that received the message, $t_i + \operatorname{dist}(u, C_i) \le b(V_0, G)$, thus $t_i + d^v_{C_i} \le b(V_0, G)$, and so

$$\max_i\{t_i + d^v_{C_i}\} \le b(V_0, G).$$

Thus from Claim 5.3 we deduce that

$$\max_i\{i + d^v_{C_i}\} \le b(V_0, G).$$

Since this holds for every vertex, we conclude $\mathcal{W}(F') \le b(V_0, G)$. Since $F$ is a minimum control function, $\mathcal{W}(F) \le \mathcal{W}(F') \le b(V_0, G)$. □

Note that we can use an algorithm similar to Algorithm BES of §3.3 to establish the following fact.

*Fact* 5.4. In the above scenario, it is possible to inform at least one vertex in $C'_i$, for every $i$, in no more than $\mathcal{W}(F)$ time units.

It is possible to use Fact 5.4 in order to construct schemes for broadcasting from a distinguished vertex $v$ in a graph with a "small" separator. Let $G$ be a $\langle \rho, \varphi(n)\rangle$-separable graph. Throughout the run, the set $V_0$ will denote the set of already informed vertices.

```
ALGORITHM APPROX_SEP
1. $V_0 \leftarrow \{v\}$.
2. Construct a separator sep($G$) for $G$.
3. Build an arbitrary tree SPT($v$, sep($G$)) = $(V_1, E_1)$ rooted at $v$ and leading to
   the members of sep($G$). Broadcast the message to the vertices of the tree using
   the OT scheme.
4. $V_0 \leftarrow V_0 \cup V_1$.
5. **Repeat**
   (a) Assume the clusters formed when extracting $V_0$ from the graph are $C_1, \ldots,$
       $C_k$. Each $C_i$ has a separator sep($C_i$) = $C^i_1 \cup C^i_2 \cup \cdots \cup C^i_{l_i}$, where $C^i_1, C^i_2, \ldots,$
       $C^i_{l_i}$ are sep($C_i$)'s connected components.
       **repeat**
       (i) For each $i$ pick the lowest index $j(i)$ that has not been chosen yet (for
           $i$).
       (ii) Build the instance $B' = \mathrm{MWC}(V', \{C^i_{j(i)} : 1 \le i \le k\})$ of the MWC
            problem, as described.
       (iii) Compute a minimum control function $F$ for $B'$ using Procedure MWC.
       (iv) Send the message to at least one vertex of $C^i_{j(i)}$ for every $i$ and $j$, using
            the minimal function $F$ and the scheme suggested in Fact 5.4.
       **until** the $C^i_j$'s clusters are exhausted for every $i$ and $j$.
   (b) For every $i$ and $j$, broadcast (in parallel) the message within $C^i_j$ using the
       best known scheme for $C^i_j$. (If no good known scheme exists for the kind of
       graph $C^i_j$ is, broadcast using an arbitrary tree.)
   (c) $V_0 \leftarrow V_0 \cup$ sep($C_1$) $\cup \cdots \cup$ sep($C_k$).
   **Until** $V_0 = V$.
```

It is easy to see that when Algorithm APPROX_SEP terminates, all the vertices in $V$ are informed.

LEMMA 5.5. *On a $\langle \rho, \varphi(n)\rangle$-separable graph, Algorithm* APPROX_SEP *terminates the broadcast process from a vertex $v$ in $O(\log n) \cdot \varphi(n) \cdot b(v)$ time units.*

*Proof.* Clearly, the number of times the external loop is performed, is bounded by $O(\log n)$. Stage 3 takes no more than $\varphi(G) +$ Diam$(v)$ time units. We next must bound the number of times that the internal loop is performed. Note that this number is bounded by the maximal number of connected components of the separator sep$(C)$ of any of the clusters $C$. Further, note that after the external loop has been executed $i$ times, the size of any separator of any cluster $C$ is bounded by $\varphi(\rho^i \cdot n) \leq \varphi(n)$. Thus, the number of connected components in the graph induced by Sep$(C)$ is also bounded by $\varphi(n)$. Thus every internal loop is carried out for no more than $\varphi(n)$ times.

Next we bound the maximal time taken by each iteration of the internal loop. By Lemma 5.2 and Fact 5.4, each execution of the repeat loop of stages 5(a)(i)–(iii) takes no more than $b(V_0, G) \leq b(v, G)$ time units. Thus the total time spent in stages 5(a)(i)–(iii) (which is no more than the number of external loops times the number of internal loops times the maximum time taken by an execution of an internal stage) is bounded by $O(\log n) \cdot \varphi(n) \cdot b_{\mathrm{op}}(v)$.

We now bound the number of time units spent in step 5(b). After the external loop took place $i$ times, the size of the separator and, hence, the size of every connected component of a separator are bounded by $\varphi(\rho^i \cdot n) \leq \varphi(n)$. Thus this is also a bound on the time spent in step 5(b), for a fixed $i$. Summing up this bound for every $i$, we conclude that the total time spent in step 5(b) is bounded by

$$\sum_{1 \leq i \leq O(\log n)} \varphi(\rho^i \cdot n) \leq O(\varphi(n) \cdot \log n).$$

Thus, in total, the broadcast time is bounded by $O(\log n) \cdot \varphi(n) \cdot b(v, G)$. □

Further, it can be shown that if we can assure that every separator sep$(C)$ is connected and $b(\text{sep}(C)) \leq k$ for some integer $k < \varphi(n)$, then the bound is improved to

$$(1) \qquad O(\log n) \cdot (b(v) + k).$$

THEOREM 5.6. *Algorithm* APPROX_SEP *is an* $O(\log n) \cdot \varphi(n)$*-approximation scheme over* $\langle \rho, \varphi(n) \rangle$*-separable graphs.*

**5.2. Applications.** In this subsection we give some examples of graph families for which Algorithm APPROX_SEP can be applied. The first example is that of *chordal graphs*. A *chord* in a cycle of at least four vertices is an edge connecting two vertices that are not adjacent in the cycle. A *chordal graph* is a graph with the property that every cycle of four vertices or more has a chord. The following theorem is shown in [GRE84] regarding chordal graphs.

THEOREM 5.7 [GRE84]. *Every n-vertex chordal graph G contains a (polynomially computable) maximal clique C, such that if the vertices in C are deleted from G, every connected component in the graph induced by any of the remaining vertices is of size at most n/2.*

An $O(|E|)$-time algorithm for finding a separating clique that satisfies the condition of the theorem is also given in [GRE84].

Thus chordal graphs always have separating sets that are connected (and moreover, are cliques). Since it is possible to broadcast a message in a clique of $m$ vertices in $\lceil \log m \rceil$ time units, it follows from (1) that the time needed to broadcast in a chordal graph using the scheme of Algorithm APPROX_SEP is no more than

$$\log n \cdot (b(v, G) + \lceil \log n \rceil) + \text{Diam}(v).$$

Consequently, Algorithm APPROX_SEP is a $(2\log n + 1)$-approximation scheme for broadcasting in chordal graphs.

COROLLARY 5.8. *There exists a polynomial* $(2\log n + 1)$*-approximation scheme for broadcasting on chordal graphs.*

A second example is the family of a $c$-separable graphs, consisting of graphs for which $\varphi(n) = c$ for some constant $c$. These graphs were considered, for instance, in [FJ90]. It follows from Theorem 5.6 that Algorithm APPROX_SEP is an $O(\log n)$-approximation scheme for broadcasting in such graphs.

We now present two examples of $O(1)$-separable graph families. The class of *$k$-outerplanar graphs* is defined as follows. Consider a plane embedding of a planar graph. The nodes on the exterior face are termed layer 1 nodes. For $i > 1$, the layer $i$ nodes are those that lie on the exterior face of the embedding resulting from the deletion of all layer $j$ nodes, $j < i$. A plane embedding is $k$-outerplanar if it contains no node with layer number larger than $k$. A planar graph is $k$-outerplanar if it has a $k$-outerplanar embedding. A graph is called *outerplanar* if it is a 1-outerplanar graph, i.e., a graph that can be embedded in the plane such that all the vertices lie on one face [Har69].

In [FJ90], Frederickson and Janardan show that any $k$-outerplanar graph is $\langle 2/3, 2\cdot k\rangle$-separable. An $O(n)$-time algorithm to find the separator is given in [FJ90]. Thus Algorithm APPROX_SEP can be used to broadcast in a $k$-outerplanar graph achieving an $O(k\log n)$-approximation scheme. Thus we have the following theorem.

THEOREM 5.9. *There exists an* $O(k\log n)$*-approximation scheme for broadcasting on a $k$-outerplanar graph.*

COROLLARY 5.10. *There exists a polynomial* $O(\log n)$*-approximation scheme for broadcasting on the family of outerplanar graphs.*

The third example is the well-known family of series-parallel graphs. Two edges in a graph are in "series" if they are the only edges incident to a node and "parallel" if they join the same pair of nodes. The definition of a series-parallel graph is recursive. First, an edge is a series-parallel graph. Next, the graph obtained by replacing any edge in a series-parallel graph either by two series edges (adding a vertex) or by two parallel edges is series-parallel. In [FJ90] it is shown that every series-parallel graph is $\langle 2/3, 2\rangle$-separable and the separator can be found in $O(n)$ time. Thus Algorithm APPROX_SEP is a polynomial $O(\log n)$-approximation algorithm for broadcasting on a series-parallel graph.

THEOREM 5.11. *There exists a polynomial* $O(\log n)$*-approximation scheme for broadcasting on a series-parallel graph.*

The last example is of the family of *bounded-face planar graphs.* The size of a face of a planar graph is the number of vertices in the face, counting multiple visits when traversing the boundary (cf. [Mil86]). The following theorem is shown in [Mil86].

THEOREM 5.12 [Mil86]. *Every planar graph with bounded-face size is* $\langle 2/3, O(\sqrt{n})\rangle$*-separable, and the separator can be chosen to be a simple cycle or a single vertex.*

A linear time algorithm to find the separating cycle or vertex is also given in [Mil86]. We use this to derive the following theorem.

THEOREM 5.13. *There exists an* $O(n^{1/4}/\sqrt{\log n})$*-approximation scheme for broadcasting on bounded-face planar graphs.*

*Proof.* Use the algorithm of [Mil86] to compute the separators needed for Algorithm APPROX_SEP. At each step of Algorithm APPROX_SEP, if the separator is a cycle, instead of broadcasting to all the vertices of the cycle separator, choose arbitrarily a "starting" vertex in the cycle and give it an index 1, while giving indices to

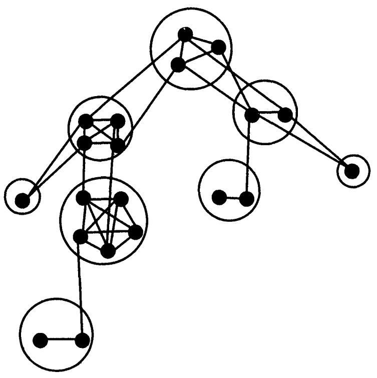


FIG. 3. *A tree of cliques (TOC) G. The large circles represent the vertices $\tilde{V}$ of the corresponding tree $T(G)$.*

the rest of the vertices in increasing order, according to their clockwise position.

Broadcast the message to every vertex whose index is congruent to 1 mod $\lceil n^{1/4} \cdot \sqrt{\log n} \rceil$, in every separating cycle. The number of recipients of the message in each cycle is $O(n^{1/4}/\sqrt{\log n})$. This is done as in Algorithm APPROX_SEP by using the technique for MWC problems. After the corresponding cycle vertices get the message, they can clearly inform the rest of the cycle vertices in no more than $O(n^{1/4} \cdot \sqrt{\log n})$ time units. Finally note that informing the vertices of the first cycle, i.e., the cycle separator of the graph itself, can be done in similar way. The broadcast time using this scheme is no more than

$$\sum_{i=1}^{\log_{3/2} n} O\left(\left(\frac{2}{3}\right)^i \cdot \frac{n^{1/4}}{\sqrt{\log n}}\right) \cdot b(v) + \sum_{i=1}^{\log_{3/2} n} \left(\frac{2}{3}\right)^i \cdot n^{1/4}\sqrt{\log n}$$
$$+ O(n^{1/4}\sqrt{\log n}) + \text{Diam}(G) = O(n^{1/4}/\sqrt{\log n}) \cdot b(v). \qquad \square$$

This is slightly better than the result given for general graphs in Theorem 4.5 (in the worst case).

## 6. Broadcasting in a tree of cliques.

In this section we present a broadcast scheme for graphs that are in a "tree of clusters" form. We illustrate this method by giving an approximation scheme for a special kind of graph family called *trees of cliques,* generalizing the family of trees.

### 6.1. Trees of cliques.

DEFINITION 6.1 (see Fig. 3). *A graph $G = (V, E)$ is a* tree of cliques (TOC) *if*

1. *the vertex set $V$ can be decomposed into a disjoint union of sets $C_1, \dots, C_k$ such that each $C_i$ induces a clique (i.e., a complete graph) in $G$, and*

2. *the auxiliary graph $T(G) = (\tilde{V}, \tilde{E})$ whose vertices are $\tilde{V} = \{C_1, \dots, C_k\}$ and whose edges are*

$$\tilde{E} = \{(C_i, C_j) : \textit{ there is an edge } (v_i, v_j) \in E, \textit{ for } v_i \in C_i, v_j \in C_j\}$$

*is a tree.*

To broadcast a message from a vertex $v$ in a TOC, we use the following idea. In order to deliver the message between vertices of different cliques (i.e., from cliques to their clique children), we use the techniques developed for MVWC problems. It follows that the total broadcast complexity spent while delivering a message between cliques is bounded by $O(b(v))$. We can achieve an efficient method, since there is an efficient method for message delivery in a clique. We then develop an alternative method for delivering the message inside the cliques. In this method, every vertex participates in the message delivery in its clique only for a small (fixed) number of rounds and is thus free sooner to help in sending the message down the tree to its clique children. Using this method we establish some improved bounds in restricted cases.

**6.2. The broadcast scheme.** Let $G$ be a TOC. The notions of child, parent, height, and so on are defined in $G$ as in the tree $T(G)$; specifically, the parent of a clique $C$ is denoted by $p(C)$, the height of a rooted TOC $G$ is denoted by $h(G)$, the subtree rooted at a given clique $C$ is denoted by $G_C$, and $T(G)_C$ is defined accordingly.

Let $G$ be a rooted TOC. The *parent index, $PI(C)$*, of a nonroot clique $C$ in $G$ is defined to be the number of vertices in $C$ that are connected to at least one vertex in the parent clique $p(C)$. The parent index of the root is defined to be 1. Similarly the *child index*, $\mathrm{CI}(C)$, of a nonleaf clique $C$ is the number of vertices in $C$ that are connected to at least one of the vertices of the children of $C$.

A TOC $G$ is *parent $c$-restricted* if it is possible to root $G$ at a clique $C$ such that $PI(C') \leq c$ for every clique $C'$. Note that the fact that a TOC is parent $c$-restricted does not preclude the possibility that *every* vertex in $P(C)$ will have an arbitrarily large (total) number of adjacent vertices in the clique children. A *child $c$-restricted* TOC is defined similarly. Note that if a TOC is child $c$-restricted, there are no more than $c$ vertices in $p(C)$ that can inform the vertices in its clique children. It follows that it seems easier to approximate the broadcast problem on a TOC if it is child $c$-restricted than if it is parent $c$-restricted.

We next give a broadcast scheme on a TOC. We assume that the clique partition of the TOC is given. Before presenting the approximation algorithm, let us give two definitions. The first definition is of the *rank* of a clique $C$ in a rooted TOC $G$. This definition induces a definition of a controlling vertex $\tilde{F}(C)$ of $C$, such that $\tilde{F}(C) \in p(C)$, for any nonroot clique $C$. Recall that for a clique $C$ in the tree, $T_C$ is the subtree rooted by $C$.

DEFINITION 6.2. *Define* $\mathrm{rank}(C) = 0$ *for a leaf $C$ in $G$. Define inductively the rank of a nonleaf clique $C$ in $T(G)$ as follows.*

1. *Form the control graph $D_{C,T_C} = (V_1, V_2, A)$ of $C$ in $T(G)_C$.*
2. *Compute recursively the ranks of the children of $C$.*
3. *Set* $\mathrm{rank}(C) = \mathcal{W}(F)$, *where $F$ is the minimal function with respect to the MVWC problem resulting by the construction of step 1 taking the weight of a clique child vertex $C'$ to be $\omega(C') =$* $\mathrm{rank}(C')$.
4. *Define $\tilde{F}(C') = F(C')$, for every child $C'$ of $C$.*

This definition of rank tries to capture the minimal degrees needed for the cliques to control their clique children. It also identifies those vertices that dominate children cliques in the TOC. Denote

$$\mathrm{Dom}(C) = \{w \in C : \text{ there exists a child } C_i \text{ of } C \text{ such that } \tilde{F}(C_i) = w\}.$$

Our second definition concerns the *degree* of cliques in the TOC and attempts to capture the *number* of vertices there are in a subtree rooted at a clique $C$.

DEFINITION 6.3. *Let $G$ be a rooted TOC. The* degree *of a leaf $C$ in $G$ is* $\deg(C) = 0$. *The degree of a nonleaf clique $C$ in $T(G)$ is defined recursively as follows:*

1. *Compute the degrees of $C$'s children in $G$.*

2. *Let* $C_1, \ldots, C_k$ *denote* $C$*'s children in* $G$, *ordered by nonincreasing degrees, i.e.,* $\deg(C_i) \geq \deg(C_{i+1})$ *for every* $i$.

3. *Define* $\deg(C) = \max_i\{\lceil \log \lceil i/PI(C)\rceil )\rceil + \deg(C_i)\}$.

Let us now describe a scheme called the *Fibonacci method,* for message dissemination within a clique. In this scheme we try to save time in informing the vertices within the clique so that a vertex will be able to start sooner to deliver the message to its clique children. Let $v_1, \ldots, v_k$ be $k$ vertices in a clique $C$. We assume that $C$ contains an informed vertex $v_0$. Our goal is to broadcast the message from $v_0$ to $\{v_1, \ldots, v_k\}$. This is done as follows:

1. In the first two steps, $v_0$ sends the message to $v_1$ and $v_2$.
2. Now define a delivery scheme for $v_i, i \geq 2$, as follows: each vertex $v_i$ spends the first two rounds after it gets the message on delivering the message within the clique. The delivery scheme prefers vertices $v_m$ with lower index. In each round $j$, the $l$ vertices that are required to participate in the delivery within the clique in this round send the message to the next lowest index $l$ vertices among $\{v_1, \ldots, v_k\}$ that did not get the message yet.

For instance, in round 3, $v_1$ and $v_2$ send the message to $v_3$ and $v_4$; in round 4, $v_1, v_2, v_3, v_4$ send the message to $v_5, v_6, v_7, v_8$; and in round 5, $v_2, \ldots, v_8$, send the message to $v_9, \ldots, v_{15}$.

Let $G$ be a TOC, and let $C_0$ be a clique in $G$ and $v_0 \in C_0$ a distinguished vertex. The goal is to broadcast the message from $v_0$ to all the vertices in $G$. We next give two recursive approximation algorithms for the problem.

ALGORITHM APPROX_TOC$_{2a}$

Input: A TOC $G$ and a root clique $C_0$ and an informed vertex $v_0 \in C_0$.

1. Compute ranks and define a dominating vertex $\tilde{F}(C_i) \in C$ for any clique children $C_i$ of any clique $C \in G$ as indicated by Definition 6.2, using Algorithm MVWC.
2. As soon as a clique $C$ contains an informed vertex $v \in C, v$ sends the message to all the vertices in the clique $C$, using an optimal procedure for the clique.
3. Each vertex $w \in$ Dom$(C)$ starts sending the message to a single arbitrary vertex in each of the cliques it controls. The delivery is performed in nonincreasing order of ranks; i.e., if rank$(C') >$ rank$(C'')$, then $w$ sends the message to a vertex in $C'$ before it sends it to a vertex in $C''$.

Next, let us modify Algorithm APPROX_TOC$_{2a}$ to get Algorithm APPROX_TOC$_{2b}$. Instead of delivering the message within the clique using all the vertices through the entire process, as done in step 2 of Algorithm APPROX_TOC$_{2a}$, we use the Fibonacci delivery scheme. Thus in Algorithm APPROX_TOC$_{2b}$ step 2 is replaced by the following step:

2. Let $C$ be a clique in $G$ containing an informed vertex $v \in C$. Assume that Dom$(C) = \{v_1, \ldots, v_k\}$. For every vertex $v_i$, let $C_{\max}(v_i)$ be the clique dominated by $v_i$ with maximal degree. Assume without loss of generality that for every $i, \deg(C_{\max}(v_i)) \geq \deg(C_{\max}(v_{i+1}))$. Apply the Fibonacci delivery method within the clique, from the informed vertex $v$ to $\{v_1, \ldots, v_k\}$.

It is easy to see that when the execution of this modified version of Algorithm APPROX_TOC$_{2a}$ terminates, every clique contains at least one informed vertex (however, not necessarily all the vertices in all the cliques are informed, since in any clique a vertex is informed only if it controls a nonempty set of children cliques). Thus to terminate the broadcast, in every clique the informed vertices deliver (in parallel) the message to the rest of the vertices in the clique, using the optimal delivery procedure for the clique. We call this modified version of the algorithm Algorithm APPROX_TOC$_{2b}$.

**6.3. The broadcast complexity of the scheme.** We now analyze the broadcast complexity of the scheme. Before stating the next claim, which speaks about the Fibonacci delivery method, recall the sequence of Fibonacci numbers defined as follows: $z_1 = z_2 = 1, z_{i+2} = z_{i+1} + z_i$ for $i \geq 1$ (cf. [HW56]).

LEMMA 6.4. *Let $C$ be a clique, and suppose that $v, v_1, \ldots, v_k \in C$ and $v$ uses the Fibonacci method to send a message to $v_1, \ldots, v_k$. Then $i$ rounds after $v_1$ receives the message from $v$, there are exactly $z_{i+2}$ informed vertices in $\{v_1, \ldots, v_k\}$.*

*Proof.* The claim holds for $i = 1, i = 2$, and $i = 3$ by a direct inspection. Now assume it is true for $i \geq 3$ and that at times $i-2, i-1$, and $i$ the number of informed vertices in $\{v_1, \ldots, v_k\}$ is $z_i, z_{i+1}$, and $z_{i+2}$, respectively.

The number of vertices that delivered the message only once within the clique is $z_{i+1} - z_i$. The number of vertices that have not yet broadcast the message at all is $z_{i+2} - z_{i+1}$. Thus the total number of informed vertices in the next round is $z_{i+2} + (z_{i+2} - z_{i+1}) + (z_{i+1} - z_i) = z_{i+2} + z_{i+1} = z_{i+3}$. $\square$

Let us now study the time needed to inform $k$ vertices $v_1, \ldots, v_k$ in the Fibonacci method. Since $z_i = (((1+\sqrt{5})/2)^i - ((1-\sqrt{5})/2)^i)/\sqrt{5}$ (cf. [HW56]),

$$z_i \geq \frac{((1+\sqrt{5})/2)^i/\sqrt{5}) - 1}{\sqrt{5}}.$$

Thus the number of rounds before $v_i$ is informed in the Fibonacci scheme is no greater than $\lceil 1.441 \cdot \log i \rceil + 2$.

We now make the following claim.

LEMMA 6.5. *For any tree of cliques $G = (V, E)$ rooted at a clique $C_0$,* $\text{rank}(C_0) \leq b(C_0, G)$.

*Proof.* We prove the lemma by induction on the height of the TOC. If $h(T(G)) = 0$ the claim holds trivially since $\text{rank}(C) = 0$. Assume the claim for height $k$. Consider a tree of height $k+1$. Assume that the children of $C_0$ in $T(G)$ are $C_1, \ldots, C_l$. Consider an optimal scheme $S$ for broadcasting from the base set $C$. For each clique $C_i$ choose a vertex $v_i \in V$ that is among the first vertices that transmit the message to a vertex in $C_i$. Since the clusters $C_i$ are independent, all of the selected vertices $v_i$ are in $C_0$. We have defined a function $F'$ from the children of $C$ to the vertices of $C$. Denote the subtree corresponding to $C_i$ by $T_i$ and the corresponding subgraph of $G$ by $G_i$.

Assume that $v$ is first to deliver the message to the cliques $C'_1, \ldots, C'_j$, that $F'(C'_i) = v$ for every $i$, and without loss of generality that the cliques $C'_i$ are arranged by nonincreasing order of ranks. Further, assume that $v$ delivers the message to a vertex in $C'_i$ at time $t_i$. We claim that

$$\max_i\{t_i + b(C'_i, G_i)\} \leq b(C, G),$$

since $t_i$ is the first time that a subset of the vertices of $C'_i$ receive the message. Thereafter they must deliver the message to the rest of the vertices of $G_i$, and this clearly takes at least $b(C'_i, G_i)$ time units. Since $h(G_i) \leq k$ for every $i$, by the induction hypothesis,

$$\max_i\{t_i + \text{rank}(C'_i)\} \leq b(C, G).$$

By Lemma 5.3,

$$\max_i\{i + \operatorname{rank}(C'_i)\} \leq b(C, G).$$

Since this is true for any vertex $v, \mathcal{W}(F') \leq b(C, G)$. Thus it follows from Definition 3.1 plus the fact that $\operatorname{rank}(C_0)$ is the weight of the minimal function that $\operatorname{rank}(C) \leq b(C_0, G)$. □

LEMMA 6.6. *Let $G$ be a TOC rooted at $C_0$ and $v_0$ a vertex in $C_0$. Then* $\deg(C) \leq b(v)$.

*Proof.* Consider an optimal scheme $S$ for broadcasting from $v$ in $G$. At the first round to where a vertex of a clique $C'$ in $G$ receives the message, there are at most $q = PI(C')$ informed vertices in $C'$. Note that the vertices of $T_{C'}$ cannot receive the message through any alternative route other than via the vertices adjacent to $p(C')$; it follows by the doubling argument of Fact 2.5(1) that for any $l > q$, it will take at least $\lceil \log(l/q) \rceil$ rounds until $l$ vertices of $T_{C'}$ are informed. Suppose that $C'$ has $l$ children $C_1, \ldots, C_l$ ordered by nonincreasing order of degrees. For any $l, 1 \leq l \leq j$, the first time that all the first $l$ cliques $C_1, \ldots, C_l$ contain an informed vertex is at least $t_0 + \lceil \log l/q \rceil$. The rest of the proof follows in a straightforward way by induction on $h(G)$. □

We are ready to analyze the complexity of the broadcast scheme. Let us divide the delays encountered by a message before it reaches a vertex in some leaf clique $C'$ into the following two possible types:

1. Delays encountered when a predecessor clique $C''$ of $C'$ delivers the message to another subtree rather than the one containing $C'$, and

2. Delays encountered by the message when it is being broadcast within a predecessor clique of $C'$.

We bound delays of the first type by proving the following (where $v_0$ is the originator of the message.)

LEMMA 6.7. *There are no more than $b(v_0) - d(C_0, C')$ delays of the first type, where $d(C_0, C')$ is the distance between $C_0$ and $C'$ in the tree $T(G)$.*

*Proof.* The proof follows by Lemma 6.5 and straightforward induction on $h(T(G))$. □

We now consider second-type delays. Let us first analyze the number of such delays in Algorithm APPROX_TOC$_{2a}$. In this case, the number of type 2 delays encountered by a message before it reaches a leaf $C'$ is bounded by $\sum_i \lceil \log C_i \rceil$, where $C_i$ is the $i$th clique in the path connecting $C$ and $p(C')$ in $T(G)$. If there are $h$ cliques in the path, then the number of type 2 delays is bounded by $h \cdot \log(n/h)$.

Since both $\log n$ and $h$ are lower bounds on the broadcast time, the worst case is when $h = \log n$, and in this case the number of type 2 delays is bounded by $\log n \cdot (\log n - \log\log n)$. Thus we have the following theorem.

THEOREM 6.8. *Algorithm* APPROX_TOC$_{2a}$ *is an additive* $\log n \cdot (\log n - \log\log n)$*-approximation scheme for broadcasting in a* TOC.

Let us now analyze the situation in Algorithm APPROX_TOC$_{2b}$. Let $C_1, \ldots, C_l$ be the children of $C$, ordered by nonincreasing order of degrees, and let $\operatorname{Dom}(C) = \{v_1, \ldots, v_k\}$. Assume without loss of generality that the vertices $v_i$ are ordered such that $\deg(C_{\max}(v_i)) \geq \deg(C_{\max}(v_{i+1}))$ for every $i$. Since $C_i$ is dominated by one of the first $i$ vertices, the number of type 2 delays encountered by the message before it is sent to $C_i$ is no more than

$$\begin{aligned} 1 + \lceil 1.441 \cdot \log i \rceil + 2 + 2 &= (\lceil 1.441 \cdot \log i \rceil - \lceil 1.441 \cdot \log PI(C_i) \rceil) \\ &\quad + \lceil 1.441 \cdot \log PI(C_i) \rceil + 5 \\ &= O\left(\frac{\log i}{PI(C_i)}\right) + O(\log PI(C_i)). \end{aligned}$$

Thus the next claim follows from Lemma 6.6 and by induction on $h(G)$. Suppose that the $i$th clique in the path in $T(G)$ from the root to a leaf $C'$ is $C_i$.

CLAIM 6.9. *The number of second-type delays encountered by the message before it reaches $C'$ in Algorithm* APPROX_TOC$_{2b}$ *is bounded by*

$$O(\deg(C)) + \sum_i \lceil \log PI(C_i) \rceil)\rceil + O(\mathrm{Diam}(v)) \le O(b(v)) + O\left(\sum_i \log PI(C_i)\right).$$

For example, consider a parent $c$-restricted TOC $G$ for a constant $c$. It follows that Algorithm APPROX_TOC$_{2b}$ is a *constant* approximation scheme for broadcasting in such a TOC. (If the goal is to broadcast the message from a vertex $v'$ that is not in $C$, where $C$ is the clique that determines the fact that $G$ is $c$-restricted, simply deliver the message to a vertex $v$ in $C$, by a shortest path, and then use the APPROX_TOC$_{2b}$ scheme to broadcast from $v$. The fact that $b(v)$ and $b(v')$ differ by at most $\mathrm{Diam}(G)$ guarantees that the scheme is still an $O(\log c)$-approximation scheme.) As one can easily check, the scheme is also an $O(\log c)$-approximation scheme in the case of a child $c$-restricted TOC. We summarize this discussion in the following theorem.

THEOREM 6.10. *Algorithm* APPROX_TOC$_{2b}$ *is a* $\min\{O(\log c_1), O(\log c_2)\}$-*approximation scheme for broadcasting in a child $c_1$-restricted, parent $c_2$-restricted* TOC.

Note that similar methods can be used to broadcast in a more general class of a "tree of clusters" graphs as long as there exists a fast approximation scheme for broadcasting in the clusters.

**Note added in proof.** Recently, a new approximation algorithm was presented for the minimum broadcast time problem [R94]. That algorithm has (*multiplicative*) approximation ratio $O(\log^2 n/\log\log n)$. Hence the new algorithm improves on the result of Theorem 4.5 for graphs whose broadcast time is $O(\sqrt{n}\log\log n/\log^2 n)$ or smaller. For graphs with larger broadcast time, our $\sqrt{n}$-*additive* algorithm still yields better approximation.

## REFERENCES

[AHU74] A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[Bol85] B. BOLLOBAS, *Random Graphs*, Academic Press, New York, 1985.

[Far80] A. M. FARLEY, *Minimum time line broadcast network*, Networks, 10 (1980), pp. 59–70.

[FH78] A. FARLEY AND S. HEDETNIEMI, *Broadcasting in grid graphs*, in Proceedings of the Ninth Southern Conference on Combinatorics, Graph Theory, and Computing, 1978, pp. 275–288.

[FJ90] G. FREDERICKSON AND R. JANARDAN, *Space-efficient message routing in c-decomposable networks*, SIAM J. Comput., 19 (1990), pp. 164–181.

[FP80] A. M. FARLEY AND A. PROSKUROWSKI, *Gossiping in grid graphs*, J. Combin. Inform. System Sci., 15 (1980), pp. 161–162.

[GJ79] 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, CA, 1979.

[GRE84] J. R. GILBERT, D. J. ROSE, AND A. EDENBRANT, *A separator theorem for chordal graphs*, SIAM J. Alg. Discrete Methods, 5 (1984), pp. 306–313.

[Har69] F. HARARY, *Graph Theory*, Addison-Wesley, Reading, MA, 1969.

[HHL88] S. HEDETNIEMI, S. HEDETNIEMI, AND A. LIESTMAN, *A survey of gossiping and broadcasting in communication networks*, Networks, 18 (1988), pp. 319–349.

[HMS72] A. HAJNAL, E. C. MILNER, AND E. SZEMEREDI, *A cure for the telephone disease*, Canad. Math. Bull., 15 (1972), pp. 447–450.

[HW56] G. H. HARDY AND E. M. WRIGHT, *An Introduction to the Theory of Numbers*, Oxford University Press, London and New York, 1956.

[Mil86] G. MILLER, *Finding small simple cycle separators for 2-connected planar graphs*, J. Comput. System Sci., 32 (1986), pp. 265–279.

[R94] R. RAVI, *Rapid rumor ramification: Approximating the minimum broadcast time*, Proc. 35th IEEE Symp. on Foundations of Computer Science, 1994, pp. 202–213.

[SCH81] P. J. SLATER, E. J. COCKAYNE, AND T. HEDETNIEMI, *Information dissemination in trees*, SIAM J. Comput., 10 (1981).

[SW84] P. SCHEUERMAN AND G. WU, *Heuristic algorithms for broadcasting in point-to-point computer networks*, IEEE Trans. Comput., 33 (1984), pp. 804–811.

SIAM J. DISC. MATH.
Vol. 8, No. 3, pp. 428–447, August 1995

# MULTIDIMENSIONAL DIVIDE-AND-CONQUER MAXIMIN RECURRENCES*

LAURENT ALONSO[†], EDWARD M. REINGOLD[‡], AND RENÉ SCHOTT[§]

**Abstract.** Bounds are obtained for the solution to the divide-and-conquer recurrence

$$M(n) = \max_{k_1+\cdots+k_p=n} (M(k_1) + M(k_2) + \cdots + M(k_p) + \min(f(k_1), \cdots, f(k_p))),$$

for nondecreasing functions $f$. Similar bounds are found for the recurrence with "min" replaced by "sum-of-all-but-the-max." Such recurrences appear in the analysis of various algorithms. The bounds follow from analyses of partition trees.

**Key words.** recurrence relations, divide and conquer, algorithmic analysis, partition trees

**AMS subject classifications.** 68Q25, 68R05, 05A20, 26A12, 05A17, 05C05

## 1. Introduction.

We consider the following four similar recurrences:

$$M(n) = \max_{k_1+\cdots+k_p=n,\ p\geq 2} \left( \sum_{i=1}^{p} M(k_i) + \min_{1\leq i\leq p} f(k_i) \right), \tag{1}$$

$$M(n) = \max_{k_1+\cdots+k_p=n,\ p\geq 2} \left( \sum_{i=1}^{p} M(k_i) + \operatorname*{sam}_{1\leq i\leq p} f(k_i) \right), \tag{2}$$

with $M(1)$ given, and

$$M(n) = \max_{k_1+\cdots+k_p=n} \left( \sum_{i=1}^{p} M(k_i) + \min_{1\leq i\leq p} f(k_i) \right), \tag{3}$$

$$M(n) = \max_{k_1+\cdots+k_p=n} \left( \sum_{i=1}^{p} M(k_i) + \operatorname*{sam}_{1\leq i\leq p} f(k_i) \right), \tag{4}$$

with $M(1), M(2), \ldots, M(p-1)$ given, where "sam" is the sum of all but the maximum, defined formally as

$$\operatorname*{sam}_{x\in S} f(x) = \sum_{x\in S} f(x) - \max_{x\in S} f(x).$$

Note that in recurrences (1) and (2), the maximum is over all partitions of $n$ into *at least* two parts, while in recurrences (3) and (4), the maximum is over all partitions of $n$ into *exactly* $p$ parts. Divide-and-conquer recurrence relations of these types, for

* Received by the editors June 19, 1992; accepted for publication (in revised form) June 6, 1994. This research was supported in part by the Institut National de la Recherche Informatique et en Automatique and National Science Foundation grant NSF INT 90-16958.

[†] Centre de Ressources Informatiques de Nancy, Institut National de la Recherche Informatique et en Automatique-Lorraine, Université de Nancy I, 54506 Vandoeuvre-lès-Nancy, France and École Normale Superieure, 45 rue d'Ulm, 75005 Paris, France.

[‡] Department of Computer Science, University of Illinois at Urbana-Champaign, 1304 W. Springfield Avenue, Urbana, Illinois 61801. The research of this author was supported in part by National Science Foundation grant CCR-93-20577.

[§] Centre de Ressources Informatiques de Nancy, Institut National de la Recherche Informatique et en Automatique-Lorraine, Université de Nancy I, 54506 Vandoeuvre-lès-Nancy, France.

various functions $f$, occur in a variety of problems in the analysis of algorithms (all-nearest-neighbors [7], tree-drawing algorithms [6], and so on). When $p = 2$, recurrence formulas (1)–(4) are identical; this case has been thoroughly investigated by Li and Reingold [4]. Our purpose is to obtain bounds for these recurrence formulas for general $p$, for nondecreasing $f$; in so doing, we sharpen one of the bounds in [4] and provide a solution to a problem left open there.[1]

In studying $M(n)$ as defined by recurrence formulas (1)–(4), we will use trees to represent the recursive evaluation. Let $\mathcal{T}(n)$ be the set of *partition trees*: ordered trees with $n-1$ internal nodes and $n$ external nodes (leaves) such that each internal node has at least two subtrees and such that the subtrees are in nondecreasing order, from left to right, by the number of leaves in the subtree. For a node $N$ of a partition tree $T$, we denote by $\#N$ the number of leaves in the subtree rooted at $N$. We define the functions $\hat{F}(T)$ and $F(T)$ by

$$F(T) = \sum_{\substack{\text{leftmost}\\ \text{nodes } N \text{ of } T}} f(\#N),$$

$$\hat{F}(T) = \sum_{\substack{\text{nonrightmost}\\ \text{nodes } N \text{ of } T}} f(\#N),$$

where a node is "leftmost" if it is the leftmost child of its parent and a node is "nonrightmost" if it is *not* the rightmost child of its parent.

If $f$ is nondecreasing, the formation rule for partition trees makes the relationship between recurrence formulas (1) and (2) and partition trees

$$M(n) = nM(1) + \max_{T\in\mathcal{T}(n)} F(T)$$

for recurrence (1) and

$$M(n) = nM(1) + \max_{T\in\mathcal{T}(n)} \hat{F}(T)$$

for recurrence (2). We will, therefore, be able to bound $M(n)$ by bounding $F(T)$ and $\hat{F}(T)$. Similar relationships hold for recurrences (3) and (4), respectively, but with the maxima taken over $p$-ary trees.

## 2. Recurrences (1) and (2).

Let $\mathcal{B}(n)$ be the set of binary partition trees, that is, partition trees in which every internal node has exactly two subtrees. Notice that for $T \in \mathcal{B}(N)$, $F(T) = \hat{F}(T)$. The following results tell us that we need only consider binary partition trees to bound $M(n)$ for recurrences (1) and (2).

LEMMA 2.1. *For recurrence* (1) *and $f$ nonnegative and nondecreasing,*

$$M(n) = nM(1) + \max_{T\in\mathcal{B}(n)} F(T).$$

*Proof.* Since $f$ is nondecreasing we know that

$$M(n) = nM(1) + \max_{T\in\mathcal{T}(n)} F(T).$$

[1] We note here that Li and Reingold [4] considered only the case when $f$ is nondecreasing, claiming that the (less interesting) nonincreasing case is easily handled by induction once a simple observation has been made. This claim is wrong, as discussed in Alonso [1].

We will prove, using a slight modification of Knuth's natural correspondence [3, p. 333], that to each tree $T \in \mathcal{T}(n)$ there corresponds a binary tree $B \in \mathcal{B}(n)$ such that

$$F(B) \geq F(T);$$

since $\mathcal{B}(n) \subset \mathcal{T}(n)$,

$$\max_{T \in \mathcal{B}(n)} F(T) \leq \max_{T \in \mathcal{T}(n)} F(T),$$

and we will be done.

We construct $B$ from $T$ inductively. If $T$ has only binary nodes, then $B = T$. Otherwise, $T$ has at least one internal node with three or more subtrees, as shown in Fig. 1. Replace this subtree with that shown in Fig. 2. Since $f$ is nonnegative, this

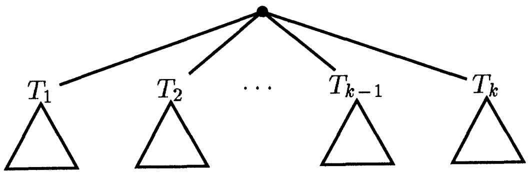


FIG. 1.

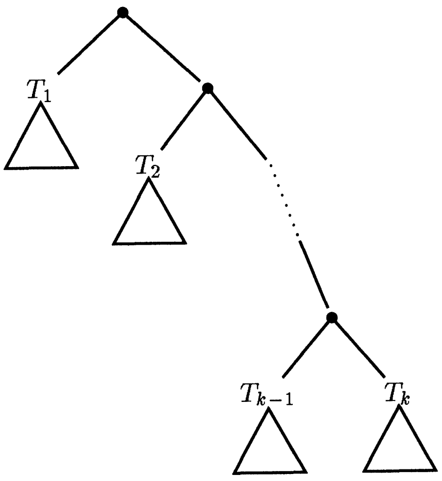


FIG. 2.

transformation does not decrease the value of $F$. □

COROLLARY 2.2. *For $f$ nondecreasing, recurrence* (1) *has the same solution as recurrence* (3) *with* $p = 2$. □

LEMMA 2.3. *For recurrence* (2) *and $f$ nondecreasing,*

$$M(n) = nM(1) + \max_{T \in \mathcal{B}(n)} F(T).$$

*Proof.* Since $f$ is nondecreasing, we know that

$$M(n) = nM(1) + \max_{T \in \mathcal{T}(n)} \hat{F}(T).$$

For any binary tree $B$, $\hat{F}(B) = F(B)$, since for binary trees the minimum is identical to the sum of all but the maximum. The same construction as in the previous lemma shows that for any $T \in \mathcal{T}(n)$, there corresponds a binary tree $B \in \mathcal{B}(n)$ such that $\hat{F}(T) = \hat{F}(B)$. Thus we have

$$F(B) = \hat{F}(B) = \hat{F}(T).$$

Since $\mathcal{B}(n) \subset \mathcal{T}(n)$,

$$\max_{B \in \mathcal{B}(n)} F(B) = \max_{T \in \mathcal{T}(n)} \hat{F}(T),$$

and the result follows. $\square$

COROLLARY 2.4. *For $f$ nondecreasing, recurrence* (2) *has the same solution as recurrence* (3) *with $p = 2$.* $\square$

THEOREM 2.5. *For $f$ nonnegative and nondecreasing in recurrence* (1), *and for $f$ nondecreasing in recurrence* (2), *the solution $M(n)$ satisfies*[2]

$$\begin{aligned} &nM(1) + \sum_{j=1}^{\lfloor \lg n \rfloor} \lfloor n/2^j \rfloor f(2^{j-1}) + \sum_{j=1}^{l-1} f\left(\sum_{i=1}^{j} 2^{k_i}\right) \\ &\le M(n) \\ &\le nM(1) + \sum_{j=1}^{\lfloor \lg n \rfloor} \lfloor n/2^j \rfloor f(2^j) + \sum_{j=1}^{l-1} f(2^{k_j+1}), \end{aligned}$$

*where $n = 2^{k_1} + 2^{k_2} + \cdots + 2^{k_l}$, $0 \le k_1 < k_2 < \cdots < k_l$, and $l \ge 1$.*

*Proof.* This follows directly from the last two corollaries, together with Corollary 9 in [4]. $\square$

THEOREM 2.6. *For $f$ nonnegative and nondecreasing in recurrence* (1), *and for $f$ nondecreasing in recurrence* (2), *the solution $M(n)$ satisfies*

$$\begin{aligned} &nM(1) + \sum_{i=0}^{\lfloor \lg n \rfloor - 1} \left( \left\lfloor \frac{n}{2^i} \right\rfloor - \left\lfloor \frac{n}{2^{i+1}} \right\rfloor \right) f(2^i) \\ &\le M(n) \\ &\le nM(1) + \sum_{i=1}^{\lfloor n/2 \rfloor} \left( \left\lfloor \frac{n}{i} \right\rfloor - \left\lfloor \frac{n}{i+1} \right\rfloor \right) f(i). \end{aligned} \tag{5}$$

*Proof.* This follows directly from the last two corollaries, together with Theorem 4.10 at the end of this paper. $\square$

Examples of these bounds applied to various functions $f$ can be found in Table 1 in [4].

## 3. Recurrence (3).

In bounding the growth of $M(n)$ as defined by recurrences (3) and (4), it is necessary to make some assumptions about the initial values $M(1)$, $M(2)$, $\ldots$, $M(p-1)$. For example, we could assume that $M$ is concave (or convex) on these values; without some such an assumption, the asymptotic behavior of $M$ would be obscured by idiosyncrasies arising from these initial values. To avoid such

[2] We use $\lg x$ for $\log_2 x$ throughout this paper.

difficulties, we assume that $M$ is defined *only* for $n$ such that $(p-1)|(n-1)$, that is, $n$ must be of the form $n = (p-1)l + 1$. This assumption is natural in the context of divide-and-conquer algorithms in which $O(p) = O(1)$ dummy elements are introduced to make the size of the input conform to the assumption. The assumption is also natural in the context of algorithms based on $p$-ary trees in which every node is either a leaf or has $p$ children; such trees have $(p-1)l + 1$ leaves.

The tree transformation technique of the previous section does not work for recurrences (3) and (4). Instead, for recurrence (3), we will use counting arguments to bound the number of leftmost nodes with a certain range of descendant leaves. Let $\mathcal{P}(n)$ be the set of $p$-ary trees with $n = (p-1)l + 1$ leaves and let

$$R_i(n) = \max_{T\in\mathcal{P}(n)} \sum_{\substack{\text{leftmost nodes } N \text{ in } T\\ \text{with } p^{i-1} < \#N \le p^i}} 1.$$

Thus $R_0(n)$ is the largest possible number of leftmost leaves in a $p$-ary tree with $n$ leaves and $R_1(n)$ is the largest possible number of internal nodes that are leftmost children of their parents and have exactly $p$ descendant leaves. $R_i(n) = 0$ for $i > \lfloor \log_p n \rfloor$, since a leftmost node with $p^{\lfloor \log_p n \rfloor} + 1$ or more descendant leaves has $p-1$ siblings to its right, each of which has at least as many descendant leaves, for a total of

$$p(p^{\lfloor \log_p n \rfloor} + 1) = p^{\lfloor \log_p n \rfloor + 1} + p > p^{\log_p n} = n$$

descendant leaves, which is impossible.

We need the following generalization of the well-known observation that a binary tree with $n$ external nodes contains $n-1$ internal nodes.

LEMMA 3.1. *A set of $k$ $p$-ary trees with a total of $I$ internal nodes contains a total of $I(p-1)+k$ external nodes.*

*Proof.* The $I$ internal nodes have a total of $Ip$ children, including the $I-k$ that are not the roots of the $k$ trees. The remaining $Ip - (I-k) = I(p-1) + k$ children must be external nodes. □

We can now express an upper bound for $F(T)$ (and hence $M(n)$) for nondecreasing functions $f$ in terms of $R$, since $f(p^i)$ is no smaller than the contribution of a node counted in $R_i(n)$. Thus

$$\begin{aligned} F(T) &= \sum_{\substack{\text{leftmost}\\ \text{nodes } N \text{ of } T}} f(\#N) \\ &= \frac{n-1}{p-1} f(1) + \sum_{\substack{\text{leftmost}\\ \text{nodes } N \text{ of } T}} [f(\#N) - f(1)], \end{aligned}$$

since by Lemma 3.1 there are $(n-1)/(p-1)$ internal nodes, each of which has a leftmost child. For $f$ nondecreasing, $f(x) - f(1)$ is a nonnegative, nondecreasing function of $x$, and hence

$$F(T) \le \frac{n-1}{p-1} f(1) + \sum_{i=0}^{\lfloor \log_p n \rfloor - 1} R_i(n)[f(p^i) - f(1)] + R_{\lfloor \log_p n \rfloor}(n)[f(\lfloor n/p \rfloor) - f(1)],$$

because $f(\#N) - f(1) \le f(p^i) - f(1)$ when $p^{i-1} < \#N \le p^i$, and because no leftmost node $N$ can have $\#N > \lfloor n/p \rfloor$. However, for $i = 0$, $f(p^i) = f(1)$, so we have

$$F(T) \le \frac{n-1}{p-1} f(1) + \sum_{i=1}^{\lfloor \log_p n \rfloor - 1} R_i(n)[f(p^i) - f(1)] + R_{\lfloor \log_p n \rfloor}(n)[f(\lfloor n/p \rfloor) - f(1)],$$

and hence

$$\begin{aligned}(6)\qquad M(n) &= nM(1) + \max_{T \in \mathcal{P}(n)} F(T) \\ &\le nM(1) + \frac{n-1}{p-1} f(1) + \sum_{i=1}^{\lfloor \log_p n \rfloor - 1} R_i(n)[f(p^i) - f(1)] \\ &\quad + R_{\lfloor \log_p n \rfloor}(n)[f(\lfloor n/p \rfloor) - f(1)]\end{aligned}$$

so that an upper bound on $R_i(n)$ will give an upper bound on $M(n)$.

Let $i > 0$. Given a tree $T \in \mathcal{P}(n)$ with $R_i(n)$ nodes $N$ for which $p^{i-1} < \#N \le p^i$, contract it by deleting all external nodes and all internal nodes whose leftmost child is *not* counted in $R_i(n)$, preserving any parent–child relationships among internal nodes that are not deleted. Then, add an external node for every missing child among the remaining nodes so that each node is properly $p$-ary. The result of this contraction is a set of $p$-ary trees that contain among them exactly $R_i(n)$ internal nodes and, by Lemma 3.1, at least $R_i(n)(p-1) + 1$ external nodes. By construction, each of these external nodes corresponds to a subtree of $T$ with at least $p^{i-1} + 1$ leaves (because *each* subtree of a node counted in $R_i(n)$ has at least as many descendant leaves as the leftmost subtree, which has at least $p^{i-1} + 1$). Thus there must have been at least $[R_i(n)(p-1) + 1](p^{i-1} + 1)$ leaves in $T$, and therefore

$$(7)\qquad n \ge [R_i(n)(p-1) + 1](p^{i-1} + 1),$$

or

$$(8)\qquad R_i(n) \le \left\lfloor \frac{n}{(p-1)(p^{i-1}+1)} - \frac{1}{p-1} \right\rfloor .$$

This bound can be strengthened when $i = 1$, since no subtree in a $p$-ary tree (except a leaf) can have fewer than $p$ leaves, so (7) becomes

$$n \ge [R_1(n)(p-1) + 1]p,$$

or

$$(9)\qquad R_1(n) \le \left\lfloor \frac{n}{p^2 - p} - \frac{1}{p-1} \right\rfloor .$$

Thus (6), (8), and (9) combine to give

$$\begin{aligned}(10)\ M(n) &\le nM(1) \\ &+ \left( \frac{n-1}{p-1} - \left\lfloor \frac{n}{p^2-p} - \frac{1}{p-1} \right\rfloor - \sum_{i=2}^{\lfloor \log_p n \rfloor} \left\lfloor \frac{n}{(p^{i-1}+1)(p-1)} - \frac{1}{p-1} \right\rfloor \right) f(1) \\ &+ \left\lfloor \frac{n}{p^2-p} - \frac{1}{p-1} \right\rfloor f(p) + \sum_{i=2}^{\lfloor \log_p n \rfloor - 1} \left\lfloor \frac{n}{(p^{i-1}+1)(p-1)} - \frac{1}{p-1} \right\rfloor f(p^i) \\ &+ \left\lfloor \frac{n}{(p^{\lfloor \log_p n \rfloor - 1}+1)(p-1)} - \frac{1}{p-1} \right\rfloor f(\lfloor n/p \rfloor).\end{aligned}$$

To obtain a lower bound for $M(n)$, we construct a tree $T_n \in \mathcal{P}(n)$ for which $F(T_n)$ is large. Let $n$ be of the form $(p-1)l+1$; the tree $T_n$ is defined recursively as follows. $T_1$ is the empty $p$-ary tree consisting of a single leaf. Given $n > 1$, let $m = \lfloor \log_p n \rfloor$ and $r = n - p^m$; $T_n$ is formed by combining $p-1$ copies of $T_{p^{m-1}}$ on the left with a copy of $T_{p^{m-1}+r}$ on the right in Fig. 3. We have

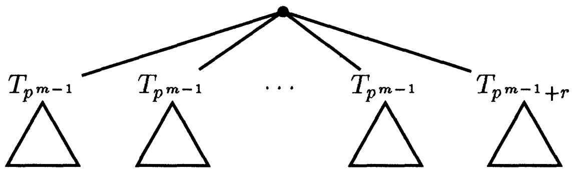


FIG. 3.

$$M(n) \geq nM(1) + F(T_n), \tag{11}$$

so we need to compute $F(T_n)$.

Let $S_i(n)$ be the number of nodes $N$ in $T_n$ for which $\#N = p^i$; clearly, when $i \leq m$,

$$S_i(p^m) = p^{m-i}.$$

Also, let $\hat{S}_i(n)$ be the number of nodes $N$ in $T_n$ for which $\#N = p^i$ but that are not on the rightmost boundary of $T_n$. We prove by induction on $n$ that for $n \geq p^i$,

$$\hat{S}_i(n) = (p-1)\left\lfloor \frac{n-p^i}{(p-1)p^i} \right\rfloor. \tag{12}$$

As the basis, observe that for $n = p^i$ the formula correctly gives $\hat{S}_i(n) = 0$. Now suppose $n > p^i$ and let $m = \lfloor \log_p n \rfloor$ and $r = n - p^m$. Since $n > p^i$, we know $m \geq i$. For $m = i$, (12) correctly gives $\hat{S}_i(n) = 0$, so, assume $m > i$, and we have

$$\begin{aligned}
\hat{S}_i(n) &= \hat{S}_i(p^m + r) \\
&= (p-1)S_i(p^{m-1}) + \hat{S}_i(p^{m-1}+r) \\
&= (p-1)p^{m-1-i} + (p-1)\left\lfloor \frac{p^{m-1}+r-p^i}{(p-1)p^i} \right\rfloor \\
&= (p-1)\left\lfloor p^{m-1-i} + \frac{p^{m-1}+r-p^i}{(p-1)p^i} \right\rfloor \\
&= (p-1)\left\lfloor \frac{n-p^i}{(p-1)p^i} \right\rfloor,
\end{aligned}$$

as desired.

Let $v_i$ be the highest node along the right boundary of $T_n$ such that $p^i \leq \#v_i < p^{i+1}$. We have

$$n = \#v_i + \hat{S}_i(n)p^i,$$

because the leaves of $T_n$ appear in subtrees of $v_i$ or in one of the $\hat{S}_i(n)$ subtrees of size $p^i$. So (12) gives us

$$\#v_i = n - \hat{S}_i(n)p^i$$

$$
\begin{aligned}
&= n - (p-1)\left\lfloor \frac{n-p^i}{(p-1)p^i} \right\rfloor p^i \\
&= n - (p-1)\left( \frac{n-p^i}{(p-1)p^i} - \left\{ \frac{n-p^i}{(p-1)p^i} \right\} \right) p^i \\
&= p^i + \left\{ \frac{n-p^i}{(p-1)p^i} \right\} (p-1)p^i,
\end{aligned}
$$

where $\{x\} = x - \lfloor x \rfloor$ is the fractional part of $x$.

Now any node $u$ of $T_n$ *not* on the right boundary of $T_n$—and hence any leftmost child of a node—has $\#u = p^i$ for some $i$. Hence we can write

$$
(13) \qquad F(T_n) = \sum_{i=0}^{\lfloor \log_p n \rfloor} \left( \begin{array}{c} \text{number of leftmost nodes} \\ N \text{ in } T_n \text{ with } \#N = p^i \end{array} \right) f(p^i),
$$

and the coefficient of $f(p^i)$ splits into two parts—the leftmost children whose parent is not on the right boundary of $T_n$ and the remaining leftmost children (that are in the subtree rooted at $v_{i+1}$). A node not on the right boundary of $T_n$ has $p$ equal-sized children, so the parent of such a node with $p^i$ descendant leaves must have $p^{i+1}$ descendant leaves. Thus there are $\hat{S}_{i+1}(n)$ such parent nodes and hence that same number of leftmost nodes $N$ in $T_n$ with $\#N = p^i$ and the parent of $N$ not on the right boundary of $T_n$. The remaining leftmost nodes $N$ in $T_n$ with $\#N = p^i$ have their parents on the right boundary of $T_n$; therefore, these parent nodes each have $p$ children, $p-1$ of which are not on the right boundary. Each of those $p-1$ non-right-boundary children is a node with $p^i$ descendant leaves—we know there are $\hat{S}_i(\#v_{i+1})$ such nodes, by definition of $\hat{S}_i$, so we have a total of $\hat{S}_i(\#v_{i+1})/(p-1)$ such parent nodes on the right boundary, each of which has a leftmost child $N$ with $\#N = p^i$. Therefore

$$
\begin{aligned}
(14) \quad & \left( \begin{array}{c} \text{number of leftmost nodes} \\ N \text{ in } T_n \text{ with } \#N = p^i \end{array} \right) \\
&= \hat{S}_{i+1}(n) + \frac{\hat{S}_i(\#v_{i+1})}{p-1}, \\
&= (p-1)\left\lfloor \frac{n-p^{i+1}}{(p-1)p^{i+1}} \right\rfloor + \left\lfloor \frac{p^{i+1} + \left\{ \frac{n-p^{i+1}}{(p-1)p^{i+1}} \right\} (p-1)p^{i+1} - p^i}{(p-1)p^{i+1}} \right\rfloor \\
&= (p-1)\left\lfloor \frac{n-p^{i+1}}{(p-1)p^{i+1}} \right\rfloor + \left\lfloor \left\{ \frac{n-p^{i+1}}{(p-1)p^{i+1}} \right\} p + 1 \right\rfloor \\
&= \left\lfloor \frac{n}{p^{i+1}} + \left\{ \frac{n-p^{i+1}}{(p-1)p^{i+1}} \right\} \right\rfloor,
\end{aligned}
$$

because

$$
(p-1)\lfloor x \rfloor + \lfloor \{x\}p + 1 \rfloor = \lfloor (p-1)\lfloor x \rfloor + \{x\}p + 1 \rfloor = \lfloor (p-1)x + \{x\} + 1 \rfloor.
$$

Combining (11), (13), and (14) gives

$$
(15) \qquad M(n) \geq nM(1) + \sum_{i=1}^{\lfloor \log_p n \rfloor} \left\lfloor \frac{n}{p^i} + \left\{ \frac{n-p^i}{(p-1)p^i} \right\} \right\rfloor f(p^{i-1}),
$$

which we combine with (10) to give the following theorem.

THEOREM 3.2. *For $f$ nondecreasing, the function defined by recurrence* (3) *for $n$ of the form $(p-1)l+1$ with $M(1)$ given satisfies*

$$\begin{aligned}
&nM(1) + \sum_{i=1}^{\lfloor \log_p n \rfloor} \left\lfloor \frac{n}{p^i} + \left\{ \frac{n-p^i}{(p-1)p^i} \right\} \right\rfloor f(p^{i-1}) \\
&\le M(n) \\
&\le nM(1) \\
&\quad + \left( \frac{n-1}{p-1} - \left\lfloor \frac{n}{p^2-p} - \frac{1}{p-1} \right\rfloor - \sum_{i=2}^{\lfloor \log_p n \rfloor} \left\lfloor \frac{n}{(p^{i-1}+1)(p-1)} - \frac{1}{p-1} \right\rfloor \right) f(1) \\
&\quad + \left\lfloor \frac{n}{p^2-p} - \frac{1}{p-1} \right\rfloor f(p) + \sum_{i=2}^{\lfloor \log_p n \rfloor - 1} \left\lfloor \frac{n}{(p^{i-1}+1)(p-1)} - \frac{1}{p-1} \right\rfloor f(p^i) \\
&\quad + \left\lfloor \frac{n}{(p^{\lfloor \log_p n \rfloor - 1}+1)(p-1)} - \frac{1}{p-1} \right\rfloor f(\lfloor n/p \rfloor). \qquad \Box
\end{aligned}$$

For example, when $f(x) = x$, Theorem 3.2 tells us that

$$\frac{1}{p} n \log_p n - O(n) \le M(n) \le \frac{p}{p-1} n \log_p n + O(n).$$

Similarly, when $f(x) = \log_p x$ we get

$$\left( M(1) + \frac{1}{(p-1)^2} \right) n - O(1) \le M(n) \le \left( M(1) + \frac{3}{(p-1)^2} + \frac{1}{p(p-1)^3} \right) n + O(1).$$

We can compare the upper and lower bounds on $M(n)$ in general for $f$ positive. Let $U(n)$ and $L(n)$ be, respectively, the upper and lower bounds in Theorem 3.2. For convenience, assume $M(1) = 0$ since the $M(1)$ term occurs with the identical coefficient in both $U(n)$ and $L(n)$. Then, from (10) and (15) we have

$$\begin{aligned}
U(n) &\le \sum_{i=0}^{\lfloor \log_p n \rfloor - 1} \frac{n}{(p^{i-1}+1)(p-1)} f(p^i) + \frac{n}{(p^{\lfloor \log_p n \rfloor - 1}+1)(p-1)} f(\lfloor n/p \rfloor), \\
L(n) &\ge \sum_{i=0}^{\lfloor \log_p n \rfloor - 1} \left\lfloor \frac{n}{p^{i+1}} \right\rfloor f(p^i).
\end{aligned}$$

Using

$$\left\lfloor \frac{n}{p^{i+1}} \right\rfloor \ge \frac{n}{2p^{i+1}},$$

we obtain

$$\begin{aligned}
U(n) &\le \frac{2p^2}{p-1} L(n) + \frac{n f(\lfloor n/p \rfloor)}{(p^{\lfloor \log_p n \rfloor}+1)(p-1)} \\
&\le \frac{2p^2}{p-1} L(n) + \frac{p^2}{p-1} f(p^{\lfloor \log_p n \rfloor - 1}) \frac{f(\lfloor n/p \rfloor)}{f(p^{\lfloor \log_p n \rfloor - 1})} \\
&\le \frac{2p^2}{p-1} L(n) + \frac{p^2}{p-1} L(n) \frac{f(\lfloor n/p \rfloor)}{f(p^{\lfloor \log_p n \rfloor - 1})},
\end{aligned}$$

and thus

$$\frac{U(n)}{L(n)} \le \frac{p^2}{p-1}\left(2 + \frac{f(\lfloor n/p \rfloor)}{f(p^{\lfloor \log_p n \rfloor - 1})}\right).$$

For $n = p^m$ this can be improved to

$$\frac{U(n)}{L(n)} \le \frac{p^2}{p-1}.$$

**4. Recurrence (4).** As in the previous section, we assume that $M$ is defined only when $(p-1)|(n-1)$, that is, only for $n$ of the form $n = (p-1)l + 1$.

A lower bound for $M(n)$ as given by recurrence (4) follows directly from our analysis of $F(T_n)$ in the previous section—in that analysis we counted the *leftmost* children of a node; here we need to count the *nonrightmost* children, and hence $p-1$ times our value for $F(T_n)$ gives a bound for $\hat{F}(T_n)$. Thus (15) becomes

$$\begin{aligned} (16) \qquad M(n) &= nM(1) + \max_{T \in \mathcal{P}(n)} \hat{F}(T) \\ &\ge nM(1) + \hat{F}(T_n) \\ &= nM(1) + (p-1)F(T_n) \\ &\ge nM(1) + (p-1) \sum_{i=1}^{\lfloor \log_p n \rfloor} \left\lfloor \frac{n}{p^i} + \left\{ \frac{n-p^i}{(p-1)p^i} \right\} \right\rfloor f(p^{i-1}). \end{aligned}$$

To obtain an upper bound on $M(n)$ as given by recurrence (4), we follow the strategy used in the previous section and use counting arguments to bound the number of nonrightmost nodes with a certain range of descendant leaves. Let

$$L_i(T) = \sum_{\substack{\text{nonrightmost nodes} \\ N \text{ in } T \text{ with } \#N > i}} 1,$$

(the root of the tree is considered a rightmost node) and let

$$\hat{L}_i(n) = \max_{T \in \mathcal{P}(n)} L_i(T).$$

Notice that $\mathcal{P}(n)$ is nonempty only for $n \equiv 1 \pmod{p-1}$; thus $\hat{L}_i(n)$ is defined only when $(p-1)|(n-1)$. We need the value of $\hat{L}_i(n)$ in what follows; to compute it we first observe the following lemma.

LEMMA 4.1. *The number of leaves in any tree* (*not necessarily of fixed arity*) *is one more than the number of nonrightmost nodes.*

*Proof.* The proof comes from simple induction on the height of the tree. □

It follows from this lemma that $\hat{L}_0(n) = n-1$. Furthermore, we use this lemma to prove the following theorem.

THEOREM 4.2. *For $n > l(p-1)+1$, $n \equiv 1 \pmod{p-1}$,*

$$\hat{L}_{l(p-1)+1}(n) = \hat{L}_{l(p-1)+2}(n) = \cdots = \hat{L}_{l(p-1)+p-1}(n) = \left\lfloor \frac{n}{l(p-1)+p} \right\rfloor - 1.$$

*Proof.* Notice that the last of these terms, with $l = -1$, correctly gives $\hat{L}_0(n) = n-1$. We have

$$\hat{L}_{l(p-1)+1}(n) = \hat{L}_{l(p-1)+2}(n) = \cdots = \hat{L}_{l(p-1)+p-1}(n),$$

because $(p-1)|(\#N-1)$ for any node $N$ in $T \in \mathcal{P}(n)$.

Take any tree $T \in \mathcal{P}(n)$, label each node $N$ with $\#N$, and remove all nodes $N$ of $T$ labeled $l(p-1)+1$ or less; we thus obtain a tree $T'$ with $L_{l(p-1)+1}(T)$ nonrightmost nodes. It follows from Lemma 4.1 that $T'$ has $L_{l(p-1)+1}(T)+1$ leaves. Each of these leaves represents a subtree of $T$ that has at least $(l+1)(p-1)+1 = l(p-1)+p$ leaves—otherwise that node would have been removed (the next larger possible number of leaves is $(l+1)(p-1)+1$). Thus

$$(l(p-1)+p)(L_{l(p-1)+1}(T)+1) \le n,$$

or

$$L_{l(p-1)+1}(T) \le \left\lfloor \frac{n}{l(p-1)+p} \right\rfloor - 1.$$

By the definition of $\hat{L}_{l(p-1)+1}(n)$, then,

$$\hat{L}_{l(p-1)+1}(n) \le \left\lfloor \frac{n}{l(p-1)+p} \right\rfloor - 1.$$

To prove that this value is a lower bound on $\hat{L}_{l(p-1)+1}(n)$, we construct a tree $T$ with $n$ leaves such that

$$L_{l(p-1)+1}(T) \ge \left\lfloor \frac{n}{l(p-1)+p} \right\rfloor - 1.$$

Let

$$u = \left\lfloor \frac{n}{l(p-1)+p} \right\rfloor$$

and

$$v = n \bmod (l(p-1)+p),$$

so that we have $u \ge 0$, $v \ge 0$,

$$n-1 = u(l+1)(p-1) + (u+v-1),$$

and hence $u+v \equiv 1 \pmod{p-1}$ because $n \equiv 1 \pmod{p-1}$. Thus $\mathcal{P}(u+v)$ is not empty; let $T' \in \mathcal{P}(u+v)$. Replace each of the rightmost $u$ leaves of $T'$ by any tree from $\mathcal{P}(l(p-1)+p)$ to obtain a tree $T \in \mathcal{P}(n)$. When $T$ is subjected to the pruning process described at the beginning of the previous paragraph, the result is a tree with at least $u$ leaves, each representing a subtree of $T$ that has at least $l(p-1)+p$ leaves. Hence by Lemma 4.1, $T$ has at least $u-1$ nonrightmost nodes $N$ with $\#N > l(p-1)+p$. □

Now for our analysis of (4). There are exactly $L_{i-1}(T) - L_i(T)$ nonrightmost nodes $N$ in $T$ with $\#N = i$ and so we have

$$\hat{F}(T) = \sum_{i \ge 1} [L_{i-1}(T) - L_i(T)] f(i). \tag{17}$$

This sum is actually finite because $L_i(T) = 0$ when $i > \lfloor (n-p)/2 \rfloor + 1$ and $T \in \mathcal{P}(n)$. If $v$ is the nonrightmost node, aside from the root, with the largest number of descendant leaves, then $v$ has a right sibling with at least as many descendant leaves and at least $p-2$ siblings with at least one descendant leaf each; thus

$$2\#v + p - 2 \leq n,$$

or

$$\#v \leq \left\lfloor \frac{n-p+2}{2} \right\rfloor,$$

so that

$$\#v \leq \left\lfloor \frac{n-p}{2} \right\rfloor + 1$$

for all nonrightmost nodes, as claimed.

We can use equation (17) and Theorem 4.2 to obtain an upper bound on $\hat{F}(T)$.

THEOREM 4.3. *Given an increasing sequence of integers $\alpha_0 = 0 < \alpha_1 < \cdots < \alpha_k$, $\alpha_k > \lfloor (n-p)/2 \rfloor + 1$, satisfying $(p-1) | (\alpha_i - 1)$, $1 \leq i \leq k$, and a corresponding sequence of functions $L_i^*(n)$, $0 \leq i \leq k$, satisfying, for all $n \equiv 1 \pmod{p-1}$, $L_{i-1}^*(n) \geq L_i^*(n) \geq \hat{L}_{\alpha_i}(n)$, $L_0^*(n) = n-1$, and $L_k^*(n) = 0$, then for nondecreasing $f$,*

$$\begin{aligned}\hat{F}(T) &\leq \sum_{i=1}^{k} [L_{i-1}^*(n) - L_i^*(n)] f(\alpha_i) \\ &= \sum_{i=1}^{k-1} [L_{i-1}^*(n) - L_i^*(n)] f(\alpha_i) + L_{k-1}^*(n) f(\alpha_k),\end{aligned}$$

*for all $T \in \mathcal{P}(n)$.*

*Proof.* First we prove the result for integer-valued functions $L_i^*(n)$; later we show how to remove this restriction. We have, by the definition of $\hat{F}$,

$$\hat{F}(T) = \sum_{\substack{\text{nonrightmost} \\ \text{nodes } N \text{ of } T}} f(\#N).$$

This sum of $n-1$ terms can be written as

$$\hat{F}(T) = \sum_{i=1}^{j} [L_{\beta_{i-1}}(T) - L_{\beta_i}(T)] f(\beta_i),$$

where $\beta_1 < \beta_2 < \cdots < \beta_j$ are the values assumed by $\#N$ as $N$ ranges over the internal nodes of $T$; since $\#N$ assumes only values of the form $(p-1)l + 1$, each $\beta_i$ satisfies $(p-1)|(\beta_i - 1)$ and hence $\beta_{i-1} \leq \beta_i - (p-1)$. Thus we can write $\hat{F}(T)$ as a sum of $n-1$ terms

$$\hat{F}(T) = f(\beta_1) + \cdots + f(\beta_j). \tag{18}$$

(A term $f(\beta_i)$ can occur more than one time, of course.) The upper bound we want to prove has the same form, namely,

$$\sum_{i=1}^{k} [L_{i-1}^*(n) - L_i^*(n)] f(\alpha_i) = f(\alpha_1) + \cdots + f(\alpha_k), \tag{19}$$

also a sum of $\sum_{i=1}^{k}[L_{i-1}^*(n) - L_i^*(n)] = L_0^*(n) - L_k^*(n) = (n-1) - 0 = n-1$ terms. We compare the sums in (18) and (19) term by term, showing that the $t$th term of (18) is less than or equal to the $t$th term of (19).

On the right-hand side of (18) the $(n - L_{\beta_{i-1}}(T))$th to the $(n - L_{\beta_i}(T) - 1)$st terms are $f(\beta_i)$, while on the right-hand side of (19), the $(n - L_{i-1}^*(n))$th to the $(n - L_i^*(n) - 1)$st terms are $f(\alpha_i)$. Suppose the $t$th term of (18) is $f(\beta_{\hat{t}})$. Then

$$n - L_{\beta_{\hat{t}-1}}(T) \le t < n - L_{\beta_{\hat{t}}}(T).$$

However, $L_{\beta_{\hat{t}-1}}(T) = L_{\beta_{\hat{t}}-(p-1)}(T)$, because, by the definition of $\beta_i$ there are no nodes $N$ satisfying $L_{\beta_{\hat{t}-1}}(T) < \#N < L_{\beta_{\hat{t}}}(T)$, but $\beta_{\hat{t}-1} \le \beta_{\hat{t}} - (p-1) < \beta_{\hat{t}}$. Thus

$$n - L_{\beta_{\hat{t}}-(p-1)}(T) \le t.$$

Since $\hat{L}_i(n) \ge L_i(T)$ for any $T \in \mathcal{P}(n)$, we have

$$n - \hat{L}_{\beta_{\hat{t}}-(p-1)}(n) \le t.$$

Let $u$ be the least index for which $\alpha_u > \beta_{\hat{t}} - (p-1)$; hence $\alpha_u \ge \beta_{\hat{t}}$. Such an index $u$ exists because $L_k^*(n) = 0$ and $T$ has, by the definition of $\beta_i$, a node $N$ with $\#N \ge \beta_{\hat{t}} - (p-1)$. However $L_{u-1}^*(n) \ge \hat{L}_{\alpha_{u-1}}(n)$ by hypothesis and $\alpha_{u-1} \le \beta_{\hat{t}} - (p-1)$, so $\hat{L}_{\alpha_{u-1}}(n) \ge \hat{L}_{\beta_{\hat{t}}-(p-1)}(n)$ and hence $L_{u-1}^*(n) \ge \hat{L}_{\beta_{\hat{t}}-(p-1)}(n)$. Thus

$$n - L_{u-1}^*(n) \le t,$$

and therefore the $t$th term of (19) is $f(\alpha_t) \ge f(\alpha_u) \ge f(\beta_{\hat{t}})$ (because $f$ is nondecreasing), which is what we wanted to prove.

We now show how to reduce the non-integer-valued case to the integer-valued case. Let the functions $L_i^*(n)$ be given and define

$$\Lambda_i^*(n) = \lfloor L_i^*(n) \rfloor.$$

We have

$$L_i^*(n) \ge \hat{L}_{\alpha_i}(n),$$

but the $\hat{L}_{\alpha_i}(n)$ are integer valued, so

$$\Lambda_i^*(n) \ge \hat{L}_{\alpha_i}(n).$$

We can thus apply our theorem, which yields

$$\hat{F}(T) \le \sum_{i=1}^{k}[\Lambda_{i-1}^*(n) - \Lambda_i^*(n)]f(\alpha_i),$$

for all $T \in \mathcal{P}(n)$. However,

$$\sum_{i=1}^{k}[\Lambda_{i-1}^*(n) - \Lambda_i^*(n)]f(\alpha_i) \le \sum_{i=1}^{k}[L_{i-1}^*(n) - L_i^*(n)]f(\alpha_i),$$

because

$$\begin{aligned}
&\sum_{i=1}^{k}[L_{i-1}^*(n) - L_i^*(n)]f(\alpha_i) - \sum_{i=1}^{k}[\Lambda_{i-1}^*(n) - \Lambda_i^*(n)]f(\alpha_i)\\
&= \sum_{i=1}^{k}[L_{i-1}^*(n) - \Lambda_{i-1}^*(n) - L_i^*(n) + \Lambda_i^*(n)]f(\alpha_i)\\
&= \sum_{i=1}^{k}[\{L_{i-1}^*(n)\} - \{L_i^*(n)\}]f(\alpha_i)\\
&= \{L_0^*(n)\}f(\alpha_1) + \sum_{i=1}^{k-1}[f(\alpha_i) - f(\alpha_{i-1})]\{L_i^*(n)\} - \{L_k^*(n)\}f(\alpha_k)\\
&= \sum_{i=1}^{k-1}[f(\alpha_i) - f(\alpha_{i-1})]\{L_i^*(n)\}\\
&\geq 0,
\end{aligned}$$

since $L_0^*(n) = n-1$, $L_k^*(n) = 0$, and $f$ is nondecreasing. □

This theorem has several interesting corollaries. First, there is a $p$-dimensional analogue of Corollary 9 in [4].

COROLLARY 4.4. *For $f$ nondecreasing, the function defined by recurrence* (4) *for $n$ of the form $(p-1)l+1$ with $M(1)$ given satisfies*

$$\begin{aligned}
M(n) \leq nM(1) + n(p-1) &\sum_{i=1}^{\lfloor \log_p((n-p+2)/2)\rfloor} \frac{f(p^i)}{p^i}\\
&+ \left(\frac{n}{p^{\lfloor \log_p((n-p+2)/2)\rfloor}} - 1\right) f\left(\left\lfloor \frac{n-p+2}{2}\right\rfloor\right).
\end{aligned}$$

*Proof.* Apply Theorem 4.3 with $k = \lfloor \log_p((n-p+2)/2)\rfloor$, $\alpha_0 = 0$, $\alpha_i = p^i$, $1 \leq i < k$, $\alpha_k = (n-p+2)/2$, $L_i^*(n) = n/p^i - 1$, $1 \leq i < k$ and $L_k^*(n) = 0$. □

Next, a much tighter upper bound is given in the following corollary.

COROLLARY 4.5. *For $f$ nondecreasing, the function defined by recurrence* (4) *for $n$ of the form $(p-1)l+1$ with $M(1)$ given satisfies*

$$\begin{aligned}
M(n) \leq nM(1) + n &\sum_{i=0}^{\lfloor (l-3)/2\rfloor} \left(\frac{1}{(p-1)i+1} - \frac{1}{(p-1)i+p}\right) f((p-1)i+1)\\
&+ \left(\frac{n}{(p-1)\lfloor (l-1)/2\rfloor + 1} - 1\right) f\left((p-1)\left\lfloor \frac{(l-1)}{2}\right\rfloor + 1\right).
\end{aligned}$$

*Proof.* Apply Theorem 4.3 with $k = \lfloor \frac{l+1}{2}\rfloor = \lfloor \frac{n+p-2}{2(p-1)}\rfloor$, $\alpha_i = (i-1)(p-1)+1$ and $L_i^*(n) = n/(\alpha_i + p - 1) - 1$, for $1 \leq i < k$, $\alpha_0 = 0$ and $L_0^*(n) = n-1$, and $\alpha_k = \lfloor \frac{n-p+2}{2}\rfloor$ and $L_k^*(n) = 0$. □

For $p = 2$, the bound in Corollary 4.5 becomes

$$M(n) \leq nM(1) + n \sum_{i=1}^{\lfloor n/2\rfloor - 1} \frac{f(i)}{i(i+1)} + O(f(\lfloor n/2\rfloor)), \tag{20}$$

a major improvement over the upper bound given in Corollary 9 in [4]. For example, when $f(x) = x$ and $M(1) = 0$, we know (see [2, Eq. 2.50], for example) that $M(n) = \frac{1}{2}n \lg n + O(n)$. The upper bound (20) gives $M(n) \leq n \ln n + O(n) \approx 0.693\ldots n \lg n + O(n)$, while the result in [4] gives only $M(n) \leq n \lg n + O(n)$. When $f(x) = \lg x$ and $M(1) = 0$, (20) gives $M(n) \leq 1.137\ldots n + O(\log n)$, while the result in [4] gives only $M(n) \leq 2n + O(\log n)$.

When $f(x) = \lceil \lg x \rceil$ we know from [4] that $M(n) = nM(1) + n - \lceil \lg n \rceil - 1$, and we can use summation by parts (see, for example, [2, Eq. 4.65] or [3, Ex. 1.2.7-10]) with (16) and (20) to obtain

$$\begin{aligned} n\,M(1) + n + O(\log^2 n) \leq M(n) &\leq nM(1) + n\sum_{i=0}^{\infty} \frac{1}{2^i + 1} + O(\log n) \\ &\approx nM(1) + 1.2645\ldots n + O(\log n). \end{aligned}$$

Most interesting is the case $f(x) = \lfloor \lg x \rfloor$. We know from [4] that $M(n) = nM(1) + n - \lfloor \lg n \rfloor - \nu(n)$, where $\nu(n) = O(\log n)$ is the number of 1-bits in the binary representation of $n$. Using summation by parts with (16) and (20), we obtain the sharp result that $M(n) = nM(1) + n + O(\log^2 n)$.

COROLLARY 4.6. *For $f$ nondecreasing, the function defined by recurrence* (4) *for $n$ of the form $(p-1)l + 1$ with $M(1)$ given satisfies*

$$M(n) \leq nM(1) + \sum_{i=0}^{\lfloor (l-1)/2 \rfloor} \left( \left\lfloor \frac{n}{(p-1)i+1} \right\rfloor - \left\lfloor \frac{n}{(p-1)i+p} \right\rfloor \right) f((p-1)i+1).$$

*Proof.* Apply Theorem 4.3 with $k = \lfloor \frac{l+1}{2} \rfloor = \lfloor \frac{n+p-2}{2(p-1)} \rfloor$, $\alpha_i = (i-1)(p-1)+1$ and $L_i^*(n) = \hat{L}_{\alpha_i}(n) = \lfloor \frac{n}{(i-1)(p-1)+p} \rfloor - 1$ for $1 \leq i \leq k$, and $\alpha_0 = 0$ and $L_0^*(n) = n - 1$. □

The upper bound in Corollary 4.6 is sharper than those in Corollaries 4.4 and 4.5; in fact, it is sharper than *any* other upper bound of the same form. Let $n$ be fixed and consider the partial order on the set $\mathcal{U}(n)$ of upper bounds for $M(n)$ of the form

$$V(f) = nM(1) + \sum_{i=1}^{k} v_i f(i) \tag{21}$$

that hold for all nondecreasing functions $f$. Upper bounds $V$ and $W$ are comparable, $V \prec W$, if $V(f) \leq W(f)$ for all nondecreasing functions $f$; $V$ and $W$ are incomparable if there exist nondecreasing functions $f$ and $g$ such that $V(f) < W(f)$ and $V(g) > W(g)$.

LEMMA 4.7. *Let*

$$V(f) = nM(1) + \sum_{i=1}^{k} v_i f(i)$$

*and*

$$W(f) = nM(1) + \sum_{i=1}^{k} w_i f(i).$$

*Then $V \prec W$ if and only if for all $j$, $1 \leq j \leq k$,*

$$\sum_{i=j}^{k} v_i \leq \sum_{i=j}^{k} w_i. \tag{22}$$

*Proof.* If $V \prec W$, then (22) follows by considering the step function

$$f_j(x) = \begin{cases} 0, & x < j, \\ 1, & x \geq j. \end{cases}$$

On the other hand, suppose that (22) holds. To prove that $V \prec W$, we must show that $V(f) \leq W(f)$ for all nondecreasing functions $f$. Let $\hat{f}(x) = f(x) - f(1)$; $\hat{f}(x)$ is a nonnegative, nondecreasing function of $x$ and, moreover,

$$W(f) - V(f) = W(\hat{f}) - V(\hat{f}),$$

so we need only prove that $V(f) \leq W(f)$ for nonnegative, nondecreasing functions $f$. Such a function can be written as a linear combination of the step functions $f_j(x)$,

$$f(x) = \sum_{j=1}^{k} c_j f_j(x) + h(x),$$

where $h(x) = 0$ when $x$ is an integer, $1 \leq x \leq k$, and the $c_i$ are nonnegative. Because $V$ only uses $f$ at integers, we have $V(f) = V(f - h)$, and we have

$$\begin{aligned} V(f) &= V(f-h) \\ &= nM(1) + \sum_{i=1}^{k} v_i[f(i) - h(i)] \\ &= nM(1) + \sum_{i=1}^{k} v_i \sum_{j=1}^{k} c_j f_j(i) \\ &= nM(1) + \sum_{i \geq j} v_i c_j, \end{aligned}$$

since $f_j(i)$ is 1 if $i \geq j$, and 0 otherwise. Thus,

$$V(f) = nM(1) + \sum_{j=1}^{k} c_j \sum_{i=j}^{k} v_i,$$

and so by (22),

$$\begin{aligned} V(f) &\leq nM(1) + \sum_{j=1}^{k} c_j \sum_{i=j}^{k} w_i \\ &= W(f) \end{aligned}$$

by a similar argument. □

LEMMA 4.8. *Let*

$$V(f) = nM(1) + \sum_{i=1}^{k} v_i f(i).$$

*Then, for* $1 \leq j \leq k$, $\sum_{i=j}^{k} v_i \geq \hat{L}_{j-1}(n)$.

*Proof.* Suppose $\sum_{i=j}^{k} v_i < \hat{L}_{j-1}(n)$. Consider the step function

$$f_j(x) = \begin{cases} 0 & \text{if } x < j, \\ 1 & \text{otherwise.} \end{cases}$$

We have

$$\begin{aligned} M(n) &= nM(1) + \max_{T \in \mathcal{P}(n)} \hat{F}(T) \\ &= nM(1) + \max_{T \in \mathcal{P}(n)} \sum_{\substack{\text{nonrightmost} \\ \text{nodes } N \text{ of } T}} f_j(\#N) \\ &= nM(1) + \max_{T \in \mathcal{P}(n)} \sum_{\substack{\text{nonrightmost nodes} \\ N \text{ in } T \text{ with } \#N > j-1}} 1 \\ &= nM(1) + \max_{T \in \mathcal{P}(n)} L_{j-1}(T) \\ &= nM(1) + \hat{L}_{j-1}(n) \\ &> nM(1) + \sum_{i=j}^{k} v_i \\ &= V(f_j), \end{aligned}$$

contradicting the fact that $V$ is an upper bound. □

THEOREM 4.9. *The upper bound of Corollary* 4.6,

$$V(f) = nM(1) + \sum_{i=0}^{\lfloor (l-1)/2 \rfloor} \left( \left\lfloor \frac{n}{(p-1)i+1} \right\rfloor - \left\lfloor \frac{n}{(p-1)i+p} \right\rfloor \right) f((p-1)i+1),$$

*is the minimum element of the partial order $\mathcal{U}(n)$; that is, $V$ is in $\mathcal{U}(n)$ and is less than or equal to any other element in $\mathcal{U}(n)$.*

*Proof.* The upper bound $V$ is of the form (21) with $k = (p-1)\lfloor (l-1)/2 \rfloor + 1$, $l = (n-1)/(p-1)$,

$$v_i = \begin{cases} \hat{L}_{i-1}(n) - \hat{L}_i(n) & \text{if } (p-1) | (i-1), \\ 0 & \text{otherwise,} \end{cases}$$

so $V$ is in $\mathcal{U}(n)$.

Given any element $W \in \mathcal{U}(n)$, $W(f) = nM(1) + \sum_{i=1}^{k} w_i f(i)$, pad the shorter of $V$ and $W$ with zeroes so the two are the same length. By Lemma 4.8, for $1 \le j \le k$,

$$\begin{aligned} \sum_{i=j}^{k} w_i \ge \hat{L}_{j-1}(n) \\ &= \sum_{i=j}^{k} [\hat{L}_{i-1}(n) - \hat{L}_i(n)] \\ &= \sum_{i=j}^{k} v_i, \end{aligned}$$

so $V \prec W$ by Lemma 4.7. □

Combining all of these results yields the following theorem.

THEOREM 4.10. *For $f$ nondecreasing, the function defined by recurrence* (4) *for $n$ of the form $(p-1)l+1$ with $M(1)$ given satisfies*

$$\begin{aligned}
&nM(1) + (p-1) \sum_{i=1}^{\lfloor \log_p n \rfloor} \left\lfloor \frac{n}{p^i} + \left\{ \frac{n-p^i}{(p-1)p^i} \right\} \right\rfloor f(p^{i-1}) \\
&\leq M(n) \\
&\leq nM(1) + \sum_{i=0}^{\lfloor (l-1)/2 \rfloor} \left( \left\lfloor \frac{n}{(p-1)i+1} \right\rfloor - \left\lfloor \frac{n}{(p-1)i+p} \right\rfloor \right) f((p-1)i+1) \\
&\leq nM(1) + n \sum_{i=0}^{\lfloor (l-3)/2 \rfloor} \left( \frac{1}{(p-1)i+1} - \frac{1}{(p-1)i+p} \right) f((p-1)i+1) \\
&\qquad + \left( \frac{n}{(p-1)\lfloor (l-1)/2 \rfloor + 1} - 1 \right) f\left( (p-1) \left\lfloor \frac{(l-1)}{2} \right\rfloor + 1 \right) \\
&\leq nM(1) + n(p-1) \sum_{i=1}^{\lfloor \log_p \frac{n-p+2}{2} \rfloor} \frac{f(p^i)}{p^i} + \left( \frac{n}{p^{\lfloor \log_p \frac{n-p+2}{2} \rfloor}} - 1 \right) f\left( \left\lfloor \frac{n-p+2}{2} \right\rfloor \right).
\end{aligned}$$

*Proof.* All of the inequalities except that between the two larger upper bounds follow immediately from our preceding discussion. Let $W(f)$ be the largest of the three upper bounds and let $V(f)$ be the second largest of the three upper bounds. We must show that $V \prec W$ in order to prove the theorem. Write $V(f)$ in the form

$$V(f) = \sum_{i=1}^{\lfloor \frac{n-p+2}{2} \rfloor} (v_i - v_{i+1}) f(i)$$

and $W(f)$ in the form

$$W(f) = \sum_{i=1}^{\lfloor \frac{n-p+2}{2} \rfloor} (w_i - w_{i+1}) f(i),$$

with

$$v_i = \frac{n}{j(p-1)+1} - 1$$

when $(j-1)(p-1)+1 < i \leq j(p-1)+1$,

$$w_i = \frac{n}{p^{j-1}} - 1$$

when $p^{j-1} < i \leq p^j$, and

$$v_{\lfloor \frac{n-p+4}{2} \rfloor} = w_{\lfloor \frac{n-p+4}{2} \rfloor} = 0.$$

Since $v_i \leq w_i$ for each $i$, it follows that $V \prec W$ by Lemma 4.7. □

For example, when $p = 2$, the bounds of Theorem 4.10 simplify to (5). When $f(x) = x$, we use the lower bound and the middle of the three upper bounds in Theorem 4.10 to find that

$$\frac{p-1}{p} n \log_p n + O(n) \leq M(n) \leq n \ln n + O(n).$$

Similarly, when $f(x) = \log_p x$, we obtain

$$\begin{aligned}&\left(M(1) + \frac{1}{p-1}\right) n + O(\log^2 n)\\ &\le M(n)\\ &\le \left(M(1) + \frac{1}{p-1} + \frac{1}{(p-1)\ln p}\right) n + O(\log n),\end{aligned}$$

when $f(x) = \lceil \log_p x \rceil$, we obtain

$$\begin{aligned}&\left(M(1) + \frac{1}{p-1}\right) n + O(\log^2 n)\\ &\le M(n)\\ &\le \left(M(1) + \frac{3}{2p-2}\right) n + O(\log n),\end{aligned}$$

and when $f(x) = \lfloor \log_p x \rfloor$, we obtain the sharper result that

$$M(n) = \left(M(1) + \frac{1}{p-1}\right) n + O(\log^2 n).$$

We can compare the upper and lower bounds in Theorem 4.10 on $M(n)$ for $f$ positive. Let $U(n)$ be the largest of the three upper bounds in Theorem 4.10 and $L(n)$ be the lower bound in Theorem 4.10. For convenience, assume $M(1) = 0$, since the $M(1)$ term occurs with the identical coefficient in both $U(n)$ and $L(n)$. It then follows from Corollary 4.4 and (16) that the same calculations that we did for recurrence (3) lead to

$$\frac{U(n)}{L(n)} \le 2p \left(1 + \frac{f(\lfloor \frac{n-p+2}{2} \rfloor)}{f(p^{\lfloor \log_p (n-p) \rfloor - 1})}\right)$$

for recurrence (4), because

$$L(n) \ge \frac{n(p-1)}{2p} \sum_{i=0}^{\lfloor \log_p n \rfloor - 1} \frac{f(p^i)}{p^i}$$

and

$$U(n) \le n(p-1) \sum_{i=1}^{\lfloor \log_p \frac{n-p+2}{2} \rfloor} \frac{f(p^i)}{p^i} + pf\left(\left\lfloor \frac{n-p+2}{2} \right\rfloor\right).$$

**5. Conclusions.** It is worth noting that, in contradistinction to the binary case explored in [4], even as strong a property as the concavity of $f$ is insufficient to determine the exact location of the maximum in recurrences (3) and (4). For example, in recurrence (3) with $p = 3$ and $f$ nondecreasing and concave, direct calculation gives unique values for $M(n)$, $3 \le n \le 53$, $n$ odd, but gives

$$\begin{aligned}M(55) &= 55M(1) + \max\{19f(1) + 5f(3) + f(5) + 2f(9),\\ &\qquad 19f(1) + 6f(3) + f(9) + f(13)\}\\ &= 55M(1) + 19f(1) + 5f(3) + f(9) + \max\{f(5) + f(9),\ f(3) + f(13)\},\end{aligned}$$

which is indeterminate given only that $f$ is nondecreasing and concave. For $f(x) = x$, $f(3)+f(13)$ is larger while for $f(x) = \ln x$, $f(5)+f(9)$ is larger; both of these functions are nondecreasing and concave. In recurrence (4) with $p = 3$ and $f$ nondecreasing and concave, direct calculation gives unique values for $M(n)$, $3 \leq n \leq 13$, $n$ odd, but gives

$$\begin{aligned} M(15) &= 15M(1) + \max\{11f(1) + 2f(3) + f(7),\ 10f(1) + 4f(3)\} \\ &= 15M(1) + 10f(1) + 2f(3) + \max\{f(1) + f(7),\ 2f(3)\}, \end{aligned}$$

which is similarly indeterminate.

Thus we leave it as an open problem to find general conditions on $f$ under which the exact location of the maximum in recurrences (3) and (4) is determined. Such a condition would likely involve the signs of the differences $\Delta f$, $\Delta^{(2)} f, \ldots, \Delta^{(p)} f$, just as the case $p = 2$ involves conditions on the signs of $\Delta f$ (that is, $f$ nondecreasing) and $\Delta^{(2)} f$ (that is, $f$ concave or convex).

## REFERENCES

[1] L. ALONSO, *Structures arborescentes*, Ph.D. thesis, Université de Paris 11, Orsay, France, 1992.

[2] D. H. GREENE AND D. E. KNUTH, *Mathematics for the Analysis of Algorithms*, 3rd ed., Birkhäuser, Boston, 1990.

[3] D. E. KNUTH, *The Art of Computer Programming, Volume* 1: *Fundamental Algorithms*, 2nd ed., Addison–Wesley, Reading, MA, 1973.

[4] Z. LI AND E. M. REINGOLD, *Solution of a divide-and-conquer maximin recurrence*, SIAM J. Comp., 18 (1989), pp. 1188–1200.

[5] E. M. REINGOLD AND J. S. TILFORD, *Tidier drawings of trees*, IEEE Trans. Software Engineering, 7 (1981), pp. 223–228.

[6] J. S. TILFORD, *Tree-drawing algorithms*, Tech. report UIUCDCS-R-81-1055, Department of Computer Science, University of Illinois, Urbana, IL, 1981; much (but not all) of the contents of this report appears in [5].

[7] P. VAIDYA, *An $O(n \log n)$ algorithm for the all-nearest-neighbors problem*, Discrete Comput. Geom., 4 (1989), pp. 101–115.

SIAM J. DISC. MATH.
Vol. 8, No. 3, pp. 448–463, August 1995

# P-COMPONENTS AND THE HOMOGENEOUS DECOMPOSITION OF GRAPHS*

BEVERLY JAMISON[†] AND STEPHAN OLARIU[‡]

**Abstract.** In this paper we introduce and investigate the notion of p-connectedness. As it turns out, this concepts leads naturally to a unique tree representation for arbitrary graphs: the leaves of this tree are the p-connected components along with weak vertices, that is, vertices of the graph that belong to no p-connected component. We then show how to refine this decomposition to obtain a new decomposition that extends the well-known modular decomposition.

**Key words.** graph decomposition, p-connectedness, graph algorithms, structural graph theory

**AMS subject classifications.** 05, 68

**1. Introduction.** It is a standard paradigm to model problems arising in communications, very large-scale integration (VLSI) design, database design, network protocol design, and other areas of computer science and engineering by graphs in the hope that the resulting graph problems can be solved quickly. A powerful tool for obtaining efficient solutions to graph problems is the *divide-and-conquer* paradigm, one of whose manifestations is graph decomposition.

An increasingly popular approach to graph decomposition involves associating with a given graph $G$ a rooted tree $T(G)$ whose leaves are subgraphs of $G$ (e.g., vertices, edges, cliques, stable sets, cutsets) and whose internal nodes correspond to certain prescribed graph operations. Of particular interest are classes of graphs $G$ for which the following conditions hold:

- $T(G)$ can be obtained *efficiently*, that is, in time polynomial in the size of $G$;
- $T(G)$ is unique up to labeled-tree isomorphism.

Tree representations satisfying the conditions mentioned above have been obtained for several classes of graphs, including cographs [4], interval graphs [3], chordal graphs [6], [15], maximal outerplanar graphs [1], traveling salesman problem (TSP) digraphs [10], $P_4$-reducible graphs [7], $P_4$-extendible graphs [8], and $P_4$-sparse graphs [9], among many others.

A well-known form of graph decomposition is the *modular* decomposition (also called *substitution* decomposition). The modular decomposition has been discovered independently by researchers in many areas. The reader is referred to Möhring and Rademacher [13], where some applications are discussed.

The purpose of this paper is to introduce and investigate a new graph-theoretic concept that we refer to as the p-connectedness of a graph. On one hand, the p-connectedness generalizes the usual connectedness. On the other hand, this concept leads naturally to a structure theorem for general graphs which, in turn, suggests a unique tree representation for arbitrary graphs: the leaves of this tree are the p-connected components and the weak vertices, that is, vertices of the graph that belong to no p-connected component. By refining our first result, we obtain a new decomposition for arbitrary graphs that we call the *homogeneous decomposition.* As with the modular decomposition, we produce a unique decomposition tree for arbitrary

* Received by the editors March 18, 1991; accepted for publication (in revised form) May 19, 1994.

[†] Department of Computer Science, Marymount University, Arlington, Virginia 22347.

[‡] Department of Computer Science, Old Dominion University, Norfolk, Virginia 23529-0162. The research of this author was supported in part by National Science Foundation grant CCR-9407180.

graphs; however, our decomposition can be seen as a natural extension of the modular decomposition, since it goes further in decomposing graphs that are prime with respect to the modular decomposition.

The principle contribution of this paper, as we see it, is the structure theorem mentioned above, which is the backbone of the whole paper, since all our results follow directly or indirectly from it. This new structure theorem sheds a new light on the modular decomposition and, at the same time, suggests a natural way of extending it.

The paper is organized as follows: §2 presents the basic definitions and introduces some new terminology; §3 studies separable p-components, a key ingredient in our decomposition; §4 presents a structure theorem for arbitrary graphs in terms of p-connected components; §5 gives our first decomposition scheme for general graphs, which we call the primeval decomposition, together with a corresponding unique tree representation; §6 discusses the homogeneous decomposition of graphs and a second tree representation for general graphs; finally, §7 summarizes the results and presents open problems.

**2. Basics and terminology.** All the graphs in this work are finite, with no loops or multiple edges. We assume familiarity with standard graph-theoretical terminology compatible with Bondy and Murty [2]. At the same time, to specify our results, we define and use some new terms.

Let $G$ be an arbitrary graph. For a vertex $x$ of $G$, we let $N(x)$ denote the set of vertices of $G$ that are adjacent to $x$. Adjacency is assumed to be non-reflexive, and so $x \notin N(x)$; we let $G_S$ stand for the subgraph of $G$ induced by $S$. Occasionally, to simplify the notation, we blur the distinction between a set $S$ of vertices and the graph $G_S$ that it induces and use the same symbol for both.

If a vertex $x$ is nonadjacent to a vertex $y$, we say that $x$ *misses* $y$ (similarly, $y$ *misses* $x$). A vertex $z$ is said to *distinguish* vertices $u$ and $v$ whenever $z$ misses precisely one of $u$, $v$. We let $P_k$ stand for the chordless path on $k$ vertices. In a $P_4$ with vertices $a$, $b$, $c$, $d$ and edges $ab$, $bc$, $cd$, the vertices $a$ and $d$ are referred to as *endpoints* while $b$ and $c$ are termed *midpoints.* Let $S$ induce a $P_4$ in $G$; a vertex $u$ outside $S$ is said to have a *partner* in $S$ if $u$ belongs to a $P_4$ involving three vertices from $S$.

Call a subset $C$ of vertices of $G$ *p-connected* if for every partition of $C$ into nonempty disjoint sets $A$ and $B$, some $P_4$ in $G$ contains vertices from both $A$ and $B$. If $C$ is maximal with this property, then $C$ is called a *p-connected component* of $G$ or simply a *p-component.* For further reference we make the following simple observation.

*Observation* 0. Let $G$ be an arbitrary graph. The following statements hold:

(0.1) $G$ admits a unique partition into p-components;

(0.2) the p-components are closed under complementation;

(0.3) every p-component is a connected subgraph of $G$ and $\overline{G}$.

(To settle (0.1), consider the binary relation $R$ on the vertex set of $G$ defined by writing $xRy$ if and only if $x$ and $y$ belong to a common p-connected set $C$. It is easy to confirm that $R$ is an equivalence relation. The p-components of $G$ are precisely the equivalence classes under $R$, hence the uniqueness. (0.2) follows from the fact that the $P_4$'s are closed under complementation; (0.3) follows from the definition of p-components together with (0.2).)

Let $S$ be a set of vertices in $G$; for a vertex $u$ outside $S$, write

- $u \in T(S)$ whenever $u$ misses no vertex in $S$;
- $u \in I(S)$ whenever $u$ misses all the vertices in $S$;

- $u \in P(S)$ whenever $u$ misses some, but not all, vertices in $S$ and $u$ has no partner in $S$.

For further reference, we take note of the following simple observations.

*Observation* 1. Let $S$ induce a $P_4$ in $G$. A vertex $u$ outside $S$ belongs to a $P_4$ involving a midpoint (endpoint) of $S$ whenever one of the following conditions is satisfied:

(1.1) $u$ distinguishes the midpoints of $S$;
(1.2) $u$ distinguishes the endpoints of $S$;
(1.3) $u$ is adjacent to both endpoints and misses both midpoints;
(1.4) $u \in T(S)$ misses a vertex in $P(S)$;
(1.5) $u \in I(S)$ is adjacent to a vertex in $P(S)$;
(1.6) $u \in T(S)$ distinguishes adjacent vertices in $I(S)$;
(1.7) $u \in I(S)$ distinguishes nonadjacent vertices in $T(S)$.

(To justify (1.1)–(1.3), note that $u$ induces a $P_4$ with three vertices in $S$; (1.5) follows from the easy observation that every vertex in $P(S)$ distinguishes an adjacent midpoint–endpoint pair in $S$. Consequently, if some vertex $u$ in $P(S)$ is adjacent to some vertex $v$ in $I(S)$, then vertices $u$ and $v$, together with suitably chosen midpoint–endpoint pair $s$, $s'$ in $S$, induce a $P_4$. (1.4) follows from a mirror argument. Finally, (1.6) and (1.7) follow directly from the definition of the sets involved.)

Let $B$ be an arbitrary p-component in a graph $G$. Note that the maximality implied by the definition guarantees that every vertex outside $B$ belongs to precisely one of the sets $T(B)$, $P(B)$, $I(B)$. Next, we make some observations about the relationships between vertices in $T(B)$, $P(B)$, and $I(B)$ for an arbitrary p-component $B$.

*Observation* 2. The following statements must hold:

(2.1) for every set $S$ inducing a $P_4$ in $B$, no vertex in $P(B)$ misses an odd number of vertices in $S$ or is adjacent to endpoints only;
(2.2) no vertex in $I(B)$ is adjacent to a vertex in $P(B)$;
(2.3) every vertex in $T(B)$ is adjacent to all vertices in $P(B)$;
(2.4) no vertex in $I(B)$ distinguishes nonadjacent vertices in $T(B)$;
(2.5) no vertex in $T(B)$ distinguishes adjacent vertices in $I(B)$.

(This follows trivially from Observation 1, together with the assumption that $B$ is maximally p-connected.)

## 3. Separable p-components.

**3. Separable p-components.** A p-component $B$ is said to be *separable* if it partitions into nonempty, disjoint sets $B_1$ and $B_2$ such that every $P_4$ with vertices from both $B_i$'s has its midpoints in $B_1$ and its endpoints in $B_2$. For convenience, we denote this partition as $(B_1,B_2)$. Separable p-components play a crucial role in our decomposition theorem. We therefore investigate some of their basic properties. In this context, to simplify the exposition, we refer to a $P_4$ with vertices from both $B_i$'s as *crossing*. Note that separable p-components are closed under complementation, since a $P_4$ is crossing in both $G$ and $\overline{G}$; consequently, in $\overline{G}$ the partition $(B_1,B_2)$ becomes $(B_2,B_1)$. The reader is referred to Fig. 1 for an illustration of this concept.

*Observation* 3. A p-component $B$ is separable whenever $P(B)$ is nonempty; furthermore, for every vertex $p$ in $P(B)$, $(B_1(p),B_2(p))$ is a partition of $B$, with $B_1(p) = N(p) \cap B$ and $B_2(p) = B \setminus B_1(p)$.

(Since $B$ is a p-component, there must exist a $P_4$ in $B$ with vertices from both $B_i(p)$'s. Let $\pi$ be an arbitrary such $P_4$. (2.1) guarantees that the endpoints are in $B_2(p)$ and the midpoints are in $B_1(p)$. Since $\pi$ was arbitrary, the conclusion follows.)

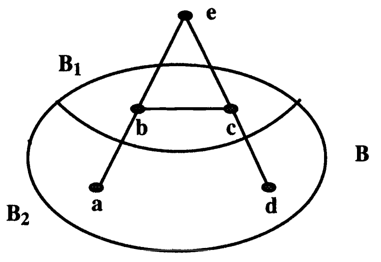


FIG. 1. *Illustrating separable p-components.*

Our first result states a fundamental property of separable p-components that is used repeatedly in the remainder of this paper.

THEOREM 1. *In a separable p-component, every vertex belongs to a crossing $P_4$.*

*Proof.* Consider an arbitrary separable p-component $B$ with a partition $(B_1,B_2)$ as described above. Since separable p-components are closed under complementation, we need only prove that every vertex in $B_2$ belongs to a crossing $P_4$.

If the statement is false, then we find a nonempty subset $A$ of $B_2$ such that no vertex in $A$ is on a crossing $P_4$. Write $C = B_2 \setminus A$. Note that since $B$ is separable, the p-connectedness of $B$ implies that $C \neq \emptyset$. Similarly, the p-connecteness of $B$ implies the existence of a $P_4$ $\pi$ with vertices from both $A$ and $C \cup B_1$. Since no vertex in $A$ is on a crossing $P_4$, the vertices of $\pi$ belong to $A \cup C$. Without loss of generality, we let $\pi$ be induced by $\{u, v, w, z\}$; furthermore, we let $u$ denote a vertex of $\pi$ that belongs to $C$. Note that the definition of $C$ guarantees that some set $S = \{u, b, c, d\}$ induces a *crossing* $P_4$ having $u$, $d$ as endpoints and $b$, $c$ as midpoints.

We claim that there is no labeling of the vertices of $\pi$ for which the edges are $uv$, $vw$, and $wz$, with $u$, $w$ in $C$ and $v$, $z$ in $A$. (To see this, note that since $z$ is adjacent to $w$ but not to $u$, $w$ and $d$ must be distinct vertices; otherwise by (1.2) $z$ would belong to a crossing $P_4$. Next, by (1.1)–(1.3), $v \in T(S)$. Because $z$ misses $u$, (1.2) and (1.4) imply that $z \in I(S)$. Now, however, either $zwvb$ or $zwbu$ is a crossing $P_4$, depending on whether or not $w$ misses $b$.)

It is easy to confirm that for a suitable labeling of the vertices in $\pi$, we have the following:

- $u \in C$, $v \in A$, with $u$ and $v$ nonadjacent;
- $w$ adjacent to both $u$ and $v$;
- $z$ misses $w$ and distinguishes $u$ and $v$.

We claim that

$$v \in I(S) \text{ and } w \in T(S). \qquad (3)$$

(First, if $w \notin T(S)$, then since $w$ cannot be a midpoint of any crossing $P_4$ it must be the case that $w$ is adjacent to $b$ and misses both $c$ and $d$; put differently, $wbcd$ is a crossing $P_4$ having $w$ as one of its endpoints. Note that since $v$ is adjacent to $w$, (1.1)–(1.3) imply that $v \in T(\{w, b, c, d\})$. Now, however, $uwvc$ is a crossing $P_4$, a contradiction. Thus $w \in T(S)$.

Note further that by (1.2), $v$ belongs to $P(S) \cup I(S)$. We may assume that $v$ belongs to $P(S)$; otherwise there is nothing to prove. Specifically, $v$ is adjacent to $b$ and $c$ and misses $u$ and $d$. We want to show that this assumption leads to a contradiction.

To see this, consider first the case where $z$ misses $u$ (and therefore $z$ is adjacent to $v$). We must have $z$ adjacent to $b$; otherwise $zvbu$ would be a crossing $P_4$, contradicting that $v \in A$. Now, however, either $ubzd$ or $zbwd$ is a crossing $P_4$ with one midpoint in $B_2$, depending on whether or not $z$ is adjacent to $d$. Either possibility contradicts that $B$ is separable.

Next, if $z$ misses $v$, $z$ must be adjacent to $u$. Note that $z$ misses $c$; otherwise $uzcv$ would be a crossing $P_4$ involving $v$. Now, however, we have reached a contradiction: $zuwc$ is a crossing $P_4$ with one of its midpoints in $B_2$, contradicting that $B$ is separable. Thus (3) must be true.)

We now continue with the proof of Theorem 1. Note that $z$ cannot miss $u$ and be adjacent to $v$, otherwise, by (3), either $vzbu$ or $zvwb$ is a crossing $P_4$, depending on whether or not $z$ is adjacent to $b$. Therefore, it must be the case that $z$ misses $v$ and is adjacent to $u$. Note that $z$ misses $c$; otherwise by (3), $zcwv$ would be a crossing $P_4$ involving $v$. However, this implies that $zuwc$ is a crossing $P_4$ with $u$ as one of its midpoints, contradicting that $B$ is separable. This completes the proof of Theorem 1. □

THEOREM 2. *If a p-component is separable, then its partition is unique.*

*Proof.* Suppose that Theorem 2 is false; let $(B_1,B_2)$ and $(B'_1,B'_2)$ be distinct partitions of a p-component $B$. By replacing $G$ with its complement $\overline{G}$, if necessary, we can ensure that

$$B_1 \cap B'_2 \neq \emptyset.$$

Let $b$ be an arbitrary vertex in $B_1 \cap B'_2$. By Theorem 1, $b$ belongs to a crossing $P_4$ with respect to the partition $(B_1,B_2)$. Since $b \in B_1$, this $P_4$ can be written as $abcd$ with $a$, $d$ in $B_2$ and $c$ in $B_1$.

Similarly, since $b \in B'_2$, Theorem 1 guarantees that $b$ belongs to a crossing $P_4$ with respect to the partition $(B'_1,B'_2)$. Furthermore, notice that this $P_4$ can be written as $buvw$ with $b$, $w$ in $B'_2$ and $u$, $v$ in $B'_1$. It is immediate that, since $b \in B_1$, all the vertices $u$, $v$, $w$ must be in $B_1$; otherwise we violate the partition $(B_1,B_2)$. Note that since $u$ belongs to $B'_1$ and $c$ belongs to $B'_2$, $u$ and $c$ are distinct vertices. We claim that

(4) $w$ is not adjacent to $c$.

(Otherwise $w$ must be adjacent to $a$ and miss $d$, or else the $P_4$'s $wcba$ or $dwab$ would violate the partition $(B_1,B_2)$. $v$ thus must be adjacent to $c$ and $a$, for if not, then the $P_4$'s $vwcb$ or $vwab$ would violate the partition $(B'_1,B'_2)$. Now, however, either $dvab$ or $dcva$ violates the partition $(B'_1,B'_2)$, depending on whether or not $v$ is adjacent to $d$. Thus $w$ misses $c$, as claimed.)

Note that by (4), together with (1.1) and (1.3),

$$w \text{ belongs to } I(\{a,b,c,d\}).$$

Similarly, since $c$ is adjacent to $b$ but not to $w$, it must be the case that

$$c \text{ is adjacent to } u \text{ but not to } v$$

and that

$$v \text{ belongs to } I(\{a,b,c,d\});$$

Otherwise we violate the partition $(B_1', B_2')$.

However, we have reached a contradiction: either *vuba* or *wvua* violates the partition $(B_1, B_2)$, depending on whether or not $u$ misses $a$. With this, the proof of Theorem 2 is complete. □

Theorem 2 implies the following result that is used repeatedly in our subsequent arguments.

COROLLARY 2.1. *Let $B$ be a p-component with $P(B)$ nonempty. For every pair of vertices $p$, $p'$ in $P(B)$, $N(p) \cap B = N(p') \cap B$.*

*Proof.* By Observation 3, $B$ is separable and both $(B_1(p), B_2(p))$ and $(B_1(p), B_2(p))$ are partitions of $B$. By Theorem 2, these partitions must coincide. The conclusion follows. □

**4. The structure theorem.** We are now in a position to state the following structure theorem for arbitrary graphs that provides the foundation of our decomposition scheme.

THEOREM 3. *For an arbitrary graph $G$, precisely one of the following conditions is satisfied:*

(i) *$G$ is disconnected;*
(ii) *$\overline{G}$ is disconnected;*
(iii) *$G$ is p-connected;*
(iv) *there is a* unique *proper separable* p-*component $H$ of $G$ with a partition $(H_1, H_2)$ such that every vertex outside $H$ is adjacent to all vertices in $H_1$ and misses all vertices in $H_2$.*

*Proof.* We need only show that if (i), (ii), and (iii) are not satified, then (iv) must be satisfied. For this purpose, we assume that both $G$ and $\overline{G}$ are connected and that $G$ alone is not p-connected. Since both $G$ and $\overline{G}$ are connected, a result of Seinsche [14] guarantees that $G$ contains a $P_4$. This, in turn, implies that $G$ must contain at least one p-component.

Choose a p-component $B$ in $G$ such that

$$(5) \qquad |P(B)| \text{ is as large as possible.}$$

We claim that

$$(6) \qquad \text{both } T(B) \text{ and } I(B) \text{ are empty.}$$

If precisely one of the sets $T(B)$ and $I(B)$ is nonempty, then by (2.2) and (2.3) combined, either $G$ or $\overline{G}$ is disconnected, contrary to our assumption. Hence, if (6) is false, then both $T(B)$ and $I(B)$ are nonempty.

We now need the following technical lemma. (Recall that the conditions of Theorem 3 still hold.)

LEMMA 4. *Let $B$ be a p-component in $G$ with both $T(B)$ and $I(B)$ nonempty. If no vertex in $T(B)$ is adjacent to all the vertices in $I(B)$, then $T(B) \cup I(B)$ contains a p-component $B'$, with $P(B) \subset P(B')$.*

*Proof.* Choose a vertex $t$ in $T(B)$ such that

$$(7) \qquad |N(t) \cap I(B)| \text{ is as large as possible.}$$

We claim that

$$\begin{gathered}\text{if a vertex } x \text{ in } T(B) \text{ is nonadjacent to a vertex in some}\\ \text{component } Z \text{ of } I(B), \text{ then } x \text{ is adjacent to no vertices in } Z.\end{gathered}\tag{8}$$

(This follows by a combination of (2.5) and the connectedness of $Z$.)

Since, by assumption, no vertex in $T(B)$ is adjacent to all the vertices of $I(B)$, (8) guarantees the existence of a component $Z'$ of $I(B)$ such that $t$ is adjacent to no vertices in $Z'$. The connectedness of $G$, together with (2.2), guarantees that some vertex $z'$ in $Z'$ is adjacent to some vertex $t'$ in $T(B)$.

Our choice of $t$, expressed in (7), implies the existence of a vertex $z$ in some component $Z$ distinct from $Z'$ such that $z$ distinguishes $t$ and $t'$. (2.4) guarantees that $t$ and $t'$ are adjacent, and thus the set $\{t, t', z, z'\}$ induces a $P_4$ in $G$. Let $B'$ stand for the p-component containing $\{t, t', z, z'\}$.

(2.2) and (2.3), combined, imply that

$$P(B) \subset P(B').$$

To see that the inclusion is strict, note that by the definition of $T(B)$ and $I(B)$, every vertex in $B$ belongs to $P(B')$. With this, the proof of Lemma 4 is complete. □

We now continue the proof of Theorem 3. If no vertex of $T(B)$ is adjacent to all the vertices of $I(B)$, then Lemma 4 guarantees the existence of a p-component $B'$ with $P(B) \subset P(B')$, contradicting our choice of $B$ in (5).

It must be the case, therefore, that some vertex $t$ in $T(B)$ is adjacent to all the vertices in $I(B)$. Let $F$ stand for the connected component of the subgraph of $\overline{G}$ induced by $T(B)$, containing $t$. Note that by (2.4), every vertex in $F$ is adjacent to all the vertices in $I(B)$. Now, however, by the definition of $T(B)$, together with (2.3), it follows that $\overline{G}$ is disconnected (since every vertex in $F$ is adjacent to all the vertices in $V \setminus F$), a contradiction. Thus (6) must be true.

Since, by assumption, $G$ itself is not a p-component, (6) guarantees that

$$V = B \cup P(B), \text{ with } P(B) \neq \emptyset.$$

By Observation 3, $B$ is separable. Theorem 2 guarantees the uniqueness of the separation. By Corollary 2.1, every vertex in $P(B)$ has the same set of neighbors in $B$. The proof of Theorem 3 is thus complete. □

**5. The primeval decomposition.** Our first decomposition scheme for general graphs, which we call the *primeval decomposition*, relies on a number of graph operations that we present next. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be disjoint graphs. Define

- $G_1 \textcircled{0} G_2 = (V_1 \cup V_2, E_1 \cup E_2)$ and
- $G_1 \textcircled{1} G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup \{xy \mid x \in V_1,\ y \in V_2\})$.

It is easy to confirm that operations $\textcircled{0}$ and $\textcircled{1}$ reflect the conditions (i) and (ii), respectively, in Theorem 3.

Operation $\textcircled{2}$, defined below reflects condition (iv) in Theorem 3. More precisely, let $G_1 = (V_1, E_1)$ be a graph such that $V_1$ is p-connected and separable with a partition $(V_1^1, V_1^2)$, and let $G_2 = (V_2, E_2)$ be an arbitrary graph disjoint from $G_1$. Define

$$G_1 \textcircled{2} G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup \{xy \mid x \in V_1^1,\ y \in V_2\}).\tag{9}$$

Note that the ② operation is well defined and admits a unique inverse: given an arbitrary graph G that satisfies condition (iv) in Theorem 3, the graphs $G_1$ and $G_2$, featured in (9), are uniquely determined.

To specify our results, we call a vertex of a graph $G$ *weak* if it is involved in no p-component of $G$. As it turns out, all graphs are constructible from atomic subgraphs by means of the operations, ⓪, ①, and ② that we just defined. More precisely, we have the following result.

THEOREM 5. *Every graph $G$ is either p-connected or it can be obtained uniquely from its p-components and weak vertices by a finite sequence of operations ⓪, ①, and ②.*

*Proof.* We proceed by induction on the size of $G = (V, E)$. If we assume the statement true for all graphs with fewer vertices than $G$, we need only prove that the statement holds for $G$ itself.

For this purpose, we assume that $G$ is not p-connected. Note that if $G$ or $\overline{G}$ is disconnected, then $G$ arises from two of its proper induced subgraphs by a ⓪ or ① operation, and the conclusion is guaranteed by the induction hypothesis. Finally, by Theorem 3, if both $G$ and $\overline{G}$ are connected and if $G$ itself is not p-connected, then there exists a unique separable p-component $H$ of $G$ and a partition $(H_1,H_2)$ of $H$, such that every vertex in $V \setminus H$ is adjacent to all the vertices in $H_1$ and misses all the vertices in $H_2$. Now, however, it is obvious that $G$ arises *uniquely* from the graphs $G_H$ and $G_V \setminus H$ by a ② operation. The proof of Theorem 5 is thus complete. □

Theorems 3 and 5 suggest a natural way of associating with every graph $G$ a unique tree $T(G)$ called the *primeval tree* of $G$. To anticipate this we notice that the leaves of $T(G)$ are precisely the p-components of $G$, along with weak vertices of $G$; an internal node $\lambda$ of $T(G)$ is labeled $i$ $(0 \leq i \leq 2)$ whenever the subgraph $H$ of $G$ corresponding to the subtree $T'$ of $T(G)$ rooted at $\lambda$ arises from two of its proper induced subgraphs by an ⓘ operation.

We are now in a position to describe the formal construction of the primeval tree of an arbitrary graph $G$. The details are spelled out by the following recursive procedure.

```
**Procedure** Build_Primeval_Tree($G$);
{Input: an arbitrary graph $G = (V, E)$;
Output: the primeval tree $T(G)$ corresponding to $G$.}
begin
    if| $V$ |= 1 or $G$ is p-connected then
        return the tree $T$ having $G$ as its unique vertex;
    else if $G$ is disconnected then begin
        let $G_1, G_2, \ldots, G_p$ $(2 \leq p)$ be the components of $G$;
        let $T_1, T_2, \ldots, T_p$ be the corresponding primeval trees rooted at $r_1, r_2, \ldots, r_p$;
        return the tree $T(G)$ obtained by adding $r_1, r_2, \ldots, r_p$ as children of a 0-
        node
        end
    else if $\overline{G}$ is disconnected then begin
        let $\overline{G}_1, \overline{G}_2, \ldots, \overline{G}_p$ $(2 \leq p)$ be the components of $\overline{G}$;
        let $T_1, T_2, \ldots, T_p$ be the corresponding primeval trees rooted at $r_1, r_2, \ldots, r_p$;
        return the tree $T(G)$ obtained by adding $r_1, r_2, \ldots, r_p$ as children of a 1-
        node
        end
    else {now $G$ satisfies condition (iv) in Theorem 3} begin
```

```
            write $G = G_1 \textcircled{2} G_2$ as in (9);
            let $T_1$, $T_2$ be the corresponding primeval trees rooted at $r_1$ and $r_2$;
            return the tree $T(G)$ obtained by adding $r_1$, $r_2$ as children of a 2-node
        end {if};
end; {Build_Primeval_Tree}
```

By Theorems 3 and 5, it follows immediately that the primeval tree associated with a graph $G$ is unique up to labeled tree isomorphism. Furthermore, the primeval tree of an arbitrary graph $G$ can be obtained in polynomial time in the number of vertices in $G$. To see that this is the case, observe that detecting whether $G$ (respectively, $\overline{G}$) is connected can be done trivially using depth-first search; detecting the p-components of $G$ can be done in a similar way. Finally, in case condition (iv) in Theorem 3 holds, the unique subgraphs featured in (9) can be obtained in polynomial time using the previous observations.

**6. The homogeneous decomposition.** A *module* $M$ in a graph $G = (V, E)$ is a set of vertices of $G$ that cannot be distinguished by vertices in $V \setminus M$. Note that, in particular, $G$ itself is a module. One form of graph decomposition, commonly referred to as *modular decomposition* (also *substitution decomposition* [12]), partitions a graph $G$ into subgraphs each of which is a module in a subgraph of $G$. The modular decomposition produces a tree that describes the submodules of $G$. It is folklore that a number of NP-complete problems can be solved efficiently if a decomposition into modules is available.

A proper subset $Y$ with at least two vertices of $G$ will be referred to as *homogeneous* (also a *nontrivial module* [12]) if every vertex outside $Y$ is adjacent to either all or none of the vertices in $Y$. Note that every homogeneous set is a module, but not vice versa: in particular, the graph itself is not considered to be a homogeneous set. Möhring [12] calls a graphs that has no nontrivial module *prime* with respect to the modular decomposition.

We propose a graph decomposition scheme that results in a unique tree representation for arbitrary graphs. This will be obtained by refining the primeval tree representation developed in the previous section. The leaves of this new tree are weak vertices along with prime subgraphs of the original graph.

Homogeneous sets (nontrivial modules) play a central role in our decomposition theorem. We, therefore investigate their properties in the context of separable p-components.

LEMMA 6. *Let $B$ be a separable p-component with a partition $(B_1, B_2)$. If the subgraph of $B$ induced by $B_2$ is disconnected, then every component of $B_2$ with at least two vertices is a homogeneous set in $B$.*

*Proof.* Let $Z$ be an arbitrary component with at least two vertices of the subgraph of $B$ induced by $B_2$. For later reference, note that since $B_2$ is disconnected, $B_2 \setminus Z \subseteq I(Z)$.

If $Z$ fails to be homogeneous, then some vertex in $B_1$ is adjacent to some, but not all, of the vertices in $Z$. We propose to show that this assumption leads to a contradiction.

To begin, we claim that if this is the case, then

$$P(Z) \text{ must be nonempty.}$$

(Otherwise, some vertex in $B_1$ belongs to a $P_4$ involving three vertices from $Z$, violating the partition $(B_1, B_2)$.)

Next, we note that

(10) no vertex in $P(Z)$ is adjacent to a vertex in $I(Z)$.

(Let $u$ be a vertex in $P(Z)$ that is adjacent to a vertex $v$ in $I(Z)$. By the connectedness of $Z$, $u$ distinguishes adjacent vertices $z$, $z'$ in $Z$. Now, however, the $P_4$ induced by $\{u, z, z', v\}$ has a midpoint in $B_2$, violating the partition $(B_1,B_2)$.)

Note further that (10) along with the fact that $B_2$ is disconnected guarantees that $T(Z)$ is nonempty: otherwise, $B$ would be disconnected, contradicting (0.3). We claim that

(11) every vertex in $P(Z)$ is adjacent to all vertices in $T(Z)$.

(Let a vertex $p$ in $P(Z)$ miss a vertex $v$ in $T(Z)$. By Theorem 1, there exists a crossing $P_4$ $uvwz$ containing $v$. Trivially, the endpoints $u$ and $z$ are in $B_2$ while the midpoints are in $B_1$. Since $v$ misses $z$, it must be the case that $z$ belongs to $B_2 \setminus Z \subseteq I(Z)$; therefore, by (10), $p$ misses $z$. Observe that $u$ belongs to $Z$: otherwise, $p$ misses $u$ and, for an arbitrary neighbor $x$ of $p$ in $Z$, $pxvu$ is a $P_4$ with one of its midpoints in $B_2$. Furthermore, since $u$ belongs to $Z$, $w$ cannot belong to $T(Z)$ (because $w$ misses $u$). In addition, since $w$ is adjacent to $z$, (10) guarantees that $w$ belongs to $I(Z)$. Now, however, we have reached a contradiction: with $x$ as before, $\{p, x, v, w\}$ induces a $P_4$, violating the partition $(B_1,B_2)$.)

Furthermore, we claim that

(12) no vertex in $I(Z)$ distinguishes nonadjacent vertices in $T(Z)$.

(Let a vertex $u$ in $I(Z)$ distinguish nonadjacent vertices $v$, $v'$ in $T(Z)$. Now, hoewver, for every choice of a vertex $x$ in $Z$, $\{u, v, v', x\}$ induces a $P_4$ with one of its midpoints in $B_2$, a contradiction.)

Finally, note that

(13) no vertex in $T(Z)$ distinguishes adjacent vertices in $I(Z)$.

(Let a vertex $u$ in $T(Z)$ distinguish adjacent vertices $v$, $v'$ in $I(Z)$, and let $x$ be an arbitrary vertex in $Z$. Clearly, the $P_4$ induced by $\{u, x, v, v'\}$ is crossing, and so exactly one of the vertices $v$, $v'$ must belong to $I(Z) \cap B_2$. Now, however, for an arbitrary choice of the vertex $p$ in $P(Z)$, (10) and (11) guarantee that $\{p, u, v, v'\}$ induces a $P_4$ that violates the partition $(B_1,B_2)$.)

To complete the proof of Lemma 6, we note that

(14) no $P_4$ in $B$ contains vertices from both $Z \cup P(Z)$ and $B \setminus (Z \cup P(Z))$.

(Let $uvwz$ be such a $P_4$. First, we argue that this $P_4$ cannot have vertices from $B_1$ only. To see this, note that by (10) and (11), such a $P_4$ must have either

- $u$, $w \in T(Z)$, $v \in P(Z)$, $z \in I(Z)$, or
- $w$, $z \in I(Z)$, $u \in P(Z)$, $v \in T(Z)$.

However, the first case is invalidated by (12) while the second contradicts (13). Therefore, the $P_4$ $uvwz$ is crossing.

If none of the endpoints belongs to $Z$, then by (10), neither $v$ nor $w$ belongs to $P(Z)$, contradicting that the $P_4$ is crossing. Similarly, if both endpoints are in $Z$, then by definition, none of the midpoints can be in $T(Z) \cup I(Z)$, contradicting that the $P_4$ is crossing.

Therefore, we may assume without loss of generality that $z$ belongs to $Z$ while $u$ belongs to $I(Z) \cap B_2$. (10) implies that $v \notin P(Z)$. Since $v$ misses $z$, we have $v \in I(Z)$. By (10), together with the fact that $w$ is adjacent to $z$, it follows that $w \in T(Z)$. Now, however, $w$, $u$, $v$ contradict (13), and the conclusion follows.)

(14) now spells out the desired contradiction: $B$ is not p-connected. This completes the proof of Lemma 6. □

THEOREM 7. *Let $G$ be an arbitrary graph and $B$ denote a separable p-component in $G$ with a partition $(B_1, B_2)$. The subgraph of $G$ (respectively, $\overline{G}$) induced by $B_2$ (respectively, $B_1$) is disconnected. Furthermore, every component of the subgraph of $G$ (respectively, $\overline{G}$) induced by $B_2$ (respectively, $B_1$) with at least two vertices is a homogeneous set in $G$.*

*Proof.* To begin, we claim that

(15) $B_2$ is not p-connected.

(Suppose, to the contrary, that $B_2$ is p-connected. Note that every vertex in $B_1$ belongs to exactly one of the sets $T(B_2)$, $P(B_2)$, $I(B_2)$, for otherwise some vertex in $B_1$ along with three vertices in $B_2$ induces a $P_4$ that violates the partition $(B_1, B_2)$.

By definition, no vertex in $T(B_2)$ (respectively, $I(B_2)$) can belong to a $P_4$ in $G$ involving vertices from $B_2$. Theorem 1 guarantees that $T(B_2)$ and $I(B_2)$ are empty. Hence, $B_1 \subseteq P(B_2)$. However, Corollary 2.1 implies that no $P_4$ contains vertices from both $B_i$'s, a contradiction. Thus (15) must be true.)

Next, we claim that

(16) the subgraph of $\overline{G}$ induced by $B_2$ is connected.

To justify (16), let $X$ and $Y$ be distinct connected components of the subgraph of $\overline{G}$ induced by $B_2$, and let $x$, $y$ be arbitrary vertices in $X$ and $Y$, respectively. By (0.2) and (0.3), $B$ induces a connected subgraph of $\overline{G}$. Consequently, we find a path

$$(P)\ x = z_0,\ z_1, \ldots,\ z_t = y$$

in $\overline{B}$ joining $x$ and $y$. By taking $t$ as small as possible, we ensure that the path $(P)$ is *chordless.*

Let $z_i$ $(1 \le i \le t-1)$ be the first vertex on the path $(P)$ that belongs to $B_1$: since $X$ and $Y$ are distinct components of $B_2$, such a vertex must exist. By our choice of $i$, $z_{i-1}$ belongs to $X \subset B_2$.

The fact that $(P)$ is chordless guarantees that if $i \le t-2$, then the set $\{z_{i-1}, z_i, z_{i+1}, z_{i+2}\}$ induces crossing a $P_4$ in $G$ with at least one midpoint in $B_2$, a contradiction.

Thus, $i = t-1$. A similar argument shows that if $i \ge 2$, then the set $\{z_{i+1}, z_i, z_{i-1}, z_{i-2}\}$ induces a $P_4$ in $G$, violating the partition $(B_1, B_2)$. Consequently,

$$t = 2 \text{ and } z_1 \text{ belongs to } B_1.$$

We claim that (in $\overline{G}$)

(17) $z_1$ misses no vertex in $X \cup Y$.

(Otherwise, without loss of generality, $z_1$ distinguishes adjacent (in $\overline{G}$) vertices $x'$ and $x''$ in $X$. Now, however, $\{z_1, x', x'', y\}$ induces a $P_4$ with three vertices in $B_2$, contradicting that $B$ is separable.)

By Theorem 1, $z_1$ belongs to a crossing $P_4$. In $\overline{G}$, this $P_4$ reads $z_1uvw$ with endpoints $z_1$ and $w$ in $B_1$ and midpoints $u$ and $v$ in $B_2$. Since $u$ and $v$ are adjacent in $\overline{G}$, they must belong to the same component $Z$ of the subgraph of $\overline{G}$ induced by $B_2$.

Since $z_1$ distinguishes $u$ and $v$, (17) guarantees that $Z$ is distinct from both $X$ and $Y$. Now, however, $\{x, z_1, u, v\}$ induces a $P_4$ that violates the partition $(B_1,B_2)$. Thus (16) must be true, as claimed.

We now continue with the proof of Theorem 7. If $B_2$ induces a connected subgraph of $B$, then by virtue of Theorem 3 applied to $B_2$, together with (15) and (16), it must be the case that

$$B_2 = H \cup P(H) \tag{18}$$

such that $H$ is a separable p-component of $B_2$, with a partition $(H_1,H_2)$, with $P(H) \neq \emptyset$, and such that every vertex in $P(H)$ is adjacent to all the vertices in $H_1$ and misses all the vertices in $H_2$.

Note further that every vertex in $B_1$ belongs to precisely one of the sets $T(H)$, $P(H)$, $I(H)$; otherwise, some vertex in $B_1$ belongs to a crossing $P_4$ involving three vertices from $H \subset B_2$, which contradicts the fact that $B$ is separable. Clearly, $B_1 \cap (T(H) \cup I(H))$ is nonempty; otherwise by Corollary 2.1, every vertex in $B \setminus H$ is adjacent to the same vertices in $H$, which contradicts the fact that $B$ is p-connected.

Let $v$ be an arbitrary vertex in $B_1 \cap (T(H) \cup I(H))$. Since $B$ is separable, Theorem 1 guarantees that $v$ belongs to a crossing $P_4$ $uvwz$ with the endpoints $u$, $z$ in $B_2$ and the midpoints $v$, $w$ in $B_1$.

First, assume that

$$v \text{ belongs to } B_1 \cap T(H).$$

Since $v$ misses $z$, $z$ cannot belong to $H$; by (18), $z$ belongs to $P(H)$. Now, however, by (1.4), $v$, $z$, together with two vertices from $H$, induce a crossing $P_4$ in $B$, which contradicts the fact that that $B$ is separable.

Next, assume that

$$v \text{ belongs to } B_1 \cap I(H).$$

Note that $u$ must belong to $P(H)$: by definition, $v$ is adjacent to no vertices in $H$. Now, however, (1.5) guarantees that the vertices $u$ and $v$, along with two vertices from $P(H)$, induce a crossing $p$, which contradicts the fact that $B$ is separable.

Therefore, $B_2$ must induce a disconnected subgraph of $G$. Lemma 6 guarantees that every component of $B_2$ containing at least two vertices is a homogeneous set in $B$. In addition, no vertex outside $B$ can distinguish vertices in such a component: this is trivially true for vertices in $T(B)$ and $I(B)$. By Corollary 2.1, it is also true of vertices in $P(B)$.

A mirror argument shows that every component with at least two vertices of the subgraph of $\overline{G}$ induced by $B_1$ is a homogeneous set in $G$. With this, the proof of Theorem 7 is complete. $\square$

A graph $G = (V, E)$ is termed a *split graph* [5] if there is a partition of the vertex set $V$ into nonempty, disjoint sets $C$ and $S$ such that $S$ is stable and $C$ is a complete set. A homogeneous set $Y$ is *maximal* in a subgraph $H$ of $G$ if no homogeneous set of $H$ properly contains $Y$. Consider shrinking every maximal homogeneous set of a p-component $B$ to one *representative* vertex: the p-component $C(B)$ obtained in this way is referred to as the *characteristic* p-component of $B$. Theorem 7 implies the following simple result, whose proof is trivial. Figure 2 features a separable p-component $B$ along with its characteristic p-component.

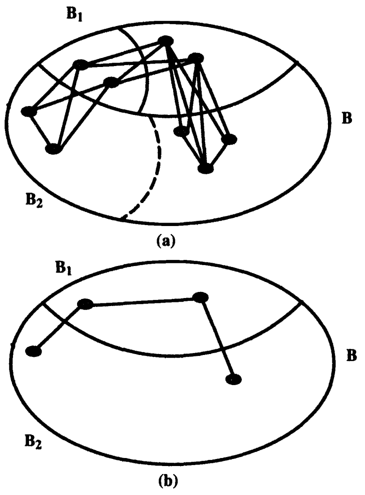


FIG. 2. *Illustrating characteristic p-components.*

COROLLARY 7.1. *A p-component $B$ is separable if and only if the characteristic p-component $C(B)$ of $B$ induces a split graph.*

To obtain the homogeneous decomposition of graphs, we augment the primeval decomposition discussed in §5 with two graph operations ③ and ④, which we define next.

Let $G_0=(V_0 \cup \{y_0, y_1, \ldots, y_t\}, E_0)$, $G_1=(V_1, E_1), \ldots,$ and $G_t = (V_t, E_t)$ be arbitrary graphs. The graph $G=(V,E)$ is said to arise from $G_0$ and $(G_1, \ldots, G_t)$ by a ③ operation if the following are true:

- $V = \bigcup_{i=0}^{t} V_i$;
- $E=(E_0 \setminus \{xy_i \mid x \in V_0 \cup \{y_0, y_1, \ldots, y_t\}, i = 1, 2, \ldots, t\}) \cup \bigcup_{i=1}^{t} E_i \cup E' \cup E''$ with $E'$ obtained by making every vertex in $V_i$ $(i = 1, 2, \ldots, t)$ adjacent to all vertices in $V_0$ adjacent to $y_i$ and $E''$ obtained by making every vertex in $V_i$ $(i = 1, 2, \ldots, t)$ adjacent to all vertices in $V_j$ if and only if $y_i y_j \in E_0$.

Loosely speaking, $G$ is obtained by replacing every vertex $y_i$ $(i = 1, 2, \ldots, t)$ in $G_0$ by the graph $G_i$. To make sure that the operation ③ admits a unique inverse, we need to make the technical assumption that every graph $G_i$ remembers the identity of the vertex $y_i$ it has replaced.

Similarly, Corollary 7.1 suggests a natural way of decomposing characteristic p-components. To see this, recall [5], [6] that a graph is a split graph if and only if both the graph and its complement are triangulated (chordal). In addition, it is well known (see [6, pp. 92–93]) that every triangulated graph admits a (not necessarily unique) tree representation $T$. The nodes of this tree are the maximal cliques of the graph; for every node $v$ in the graph, the subset of nodes corresponding to the maximal cliques that contain $v$ is a subtree of the tree $T$.

In the case of a split graph that is the characteristic p-component, as in Corollary 7.1, it is easy to exhibit a tree representation that is *unique*. To clarify this, let the vertices of the split graph partition into nonempty, disjoint sets $K$ and $S$, inducing a clique and a stable set, respectively. Note that $K$ is a maximal clique in the graph; for

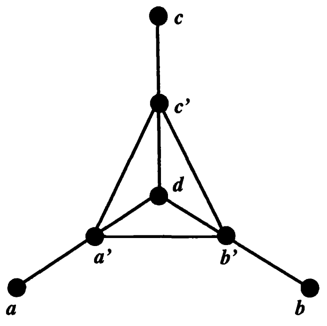


FIG. 3. *A graph G.*

every vertex $s$ in $S$, $s \cup N(s)$ is a maximal clique as well. The unique tree representation is obtained by placing $K$ at the root and, for all $s$ in $S$, having the clique $s \cup N(s)$ as a child of the root. The operation described above is captured by a ④ operation in our overall decomposition.

To summarize our findings, we state a result that extends Theorem 5. (The proof is similar to that of Theorem 5 and therefore omitted.)

THEOREM 8. *Every graph G can be obtained uniquely from its weak vertices and characteristic p-components by a finite sequence of operations ⓪, ①, ②, and ③.*

Next, we describe the formal construction of the homogeneous tree of an arbitrary graph $G$ by the following recursive procedure.

```
Procedure Build_Homogeneous_Tree($G$);
{Input: an arbitrary graph $G = (V, E)$;
Output: the homogeneous tree $HT(G)$ corresponding to $G$;}
begin
    if $|V| = 1$ then
        return the tree $HT$ having $G$ as its unique vertex;
    else if $G$ is p-connected then begin
        let $C(G)$ be the characteristic p-component of $G$;
        decompose $C(G)$ by a ④ operation and let $T'$ be the corresponding
        tree rooted at a 4-node $\alpha$;
        let $Y_1, Y_2, \ldots, Y_t$ be the maximal homogeneous sets of $G$;
        let $T_1, T_2, \ldots, T_p$ be the homogeneous trees
        corresponding to $Y_1, Y_2, \ldots, Y_t$, rooted at $r_1, r_2, \ldots, r_p$;
        return the tree $HT(G)$ obtained by adding $\alpha, r_1, r_2, \ldots, r_p$ as
        children of a 3-node;
        {Comment: it is assumed that the root $r_i$ of the homogeneous tree
        corresponding to $Y_i$ stores $y_i$}
        end
    else if $G$ is disconnected then begin
        let $G_1, G_2, \ldots, G_p$ $(2 \leq p)$ be the components of $G$;
        let $T_1, T_2, \ldots, T_p$ be the corresponding homogeneous trees rooted at $r_1, r_2,$
        $\ldots, r_p$;
        return the tree $HT(G)$ obtained by adding $r_1, r_2, \ldots, r_p$ as children of a
        0-node
        end
    else if $\overline{G}$ is disconnected then begin
```

```
        let $\overline{G}_1, \overline{G}_2, \ldots, \overline{G}_p$ $(2 \le p)$ be the components of $\overline{G}$;
        let $T_1, T_2, \ldots, T_p$ be the corresponding homogeneous trees rooted at $r_1, r_2,$
        $\ldots, r_p$;
        return the tree $HT(G)$ obtained by adding $r_1, r_2, \ldots, r_p$ as children of a
        1-node
        end
    else {now $G$ satisfies condition (iv) in Theorem 3} begin
        write $G = G_1 \circledtwo G_2$ as in (9);
        let $T_1$, $T_2$ be the corresponding homogeneous trees rooted at $r_1$ and $r_2$;
        return the tree $HT(G)$ obtained by adding $r_1$, $r_2$ as children of a 2-node
    end {if};
end; {Build_Homogeneous_Tree}
```

By Theorems 3, 5, and 8 and the previous discussion combined, it follows that the homogeneous tree associated with a graph $G$ is unique up to labeled tree isomorphism. Furthermore, just as the primeval tree, the homogeneous tree of an arbitrary graph $G$ can be obtained in polynomial time in the number of vertices in $G$.

To illustrate the differences between the modular decomposition and the proposed homogeneous decomposition the reader is referred to Figs. 3–5. To wit, the graph featured in Fig. 3 is prime with respect to modular decomposition, and this is reflected in the corresponding modular decomposition tree (see Fig. 4): it consists of the root and has every vertex in the graph as a leaf.

In contrast, the homogeneous decomposition starts by identifying $a, a', b, b', c, c'$ as a separable p-component $B$ of the graph and the vertex $d$ as belonging to $P(B)$. Furthermore, the graph induced by $a, a', b, b', c, c'$ is a split graph, and thus we decompose it again as discussed above. The net result is demonstrated in Fig. 5.

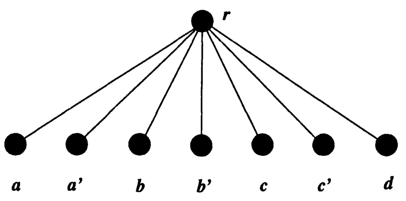


FIG. 4. *The modular decomposition of G.*

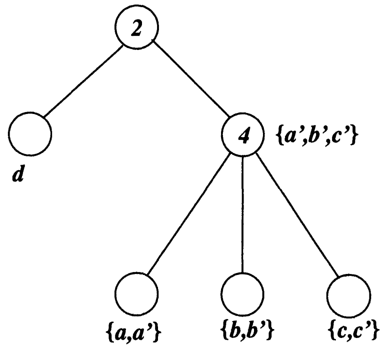


FIG. 5. *The homogeneous decomposition of G.*

**7. Conclusions and open problems.** In this paper, we have introduced and investigated the notion of p-connectedness of a graph. As it turns out, this concept leads naturally to a unique tree representation for arbitrary graphs: the leaves of this tree are the p-connected components along with weak vertices, that is, vertices of the graph that belong to no p-connected component.

We then showed how to refine this first decomposition to obtain a graph decomposition that constitutes a natural extension of the well-known modular decomposition. We argued that both of decompositions can be obtained in polynomial time in the size of the graph. Given the efficient algorithm to obtain the modular decomposition [11], it is natural to ask for a similar algorithm for the homogeneous decomposition. We pose this as an open problem.

## REFERENCES

[1] T. Beyer, W. Jones, and S. Mitchell, *A linear algorithm for isomorphism of maximal outerplanar graphs*, report CS-TR-78-1, Dept. of Computer Science, Univ. of Oregon, Eugene, OR, 1978.

[2] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, North–Holland, Amsterdam, 1976.

[3] K. S. Booth and G. S. Lueker, *A linear time algorithm for deciding interval graph isomorphism*, J. Assoc. Comput. Mach., 26 (1979), pp. 183–195.

[4] D. G. Corneil, Y. Perl, and L. K. Stewart, *A linear recognition algorithm for cographs*, SIAM J. Comput., 14 (1985), pp. 926–934.

[5] S. Földes and P. L. Hammer, *Split graphs*, in Proc. 8th Southeastern Conf. on Combinatorics, Graph Theory, and Computing, Boca Raton, FL, 1977, pp. 311–315.

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

[7] B. Jamison and S. Olariu, *$P_4$-reducible graphs, a class of uniquely tree representable graphs*, Stud. Appl. Math., 81 (1989), pp. 79–87.

[8] ———, *On a Unique Tree Representation for P4-extendible Graphs*, Discrete Applied Mathematics, 34 (1991), pp. 151–164.

[9] ———, *A tree representation for $P_4$-sparse graphs*, Discrete Appl. Math., 35 (1992), pp. 115–129.

[10] E. L. Lawler, *Graphical algorithms and their complexity*, Math. Center Tracts, 81 (1976), pp. 3–32.

[11] R. M. McConnell and J. Spinrad, *Linear-time modular decomposition and efficient transitive orientation of comparability graphs*, in Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, 1994, pp. 536–544.

[12] R. H. Möhring, *Algorithmic aspects of comparability graphs and interval graphs*, in Graphs and Order, I. Rival., ed, Reidel, Dordrecht, Holland, 1985.

[13] R. H. Möhring and F. J. Rademacher, *Substitution decomposition and connections with combinatorial optimization*, Ann. Discrete Math., 19 (1984), pp. 257–356.

[14] D. Seinsche, *On a property of the class of n-colorable graphs*, J. Combin. Theory Ser. B, 16 (1974), pp. 191–193.

[15] Y. Shibata, *On the tree representation of chordal graphs*, J. Graph Theory, 12 (1988), pp. 421–428.

SIAM J. DISC. MATH.
Vol. 8, No. 3, pp. 464–483, August 1995

# THE COMPLEXITY OF THE HAJÓS CALCULUS*

TONIANN PITASSI† AND ALASDAIR URQUHART‡

**Abstract.** The *Hajós calculus* is a simple, nondeterministic procedure that generates the class of non-3-colorable graphs. Mansfield and Welsch posed the question of whether there exist graphs that require exponential-sized Hajós constructions. Unless NP $\neq$ coNP, there must exist graphs that require exponential-sized constructions, but to date, little progress has been made on this question, despite considerable effort. In this paper, we prove that the Hajós calculus generates polynomial-sized constructions for all non-3-colorable graphs if and only if extended Frege systems are polynomially bounded. Extended Frege systems are a very powerful family of proof systems for proving tautologies, and proving superpolynomial lower bounds for these systems is a long-standing, important problem in logic and complexity theory. We also establish a relationship between a complete subsystem of the Hajós calculus and bounded-depth Frege systems; this enables us to prove exponential lower bounds on this subsystem of the Hajós calculus.

**Key words.** graph constructions, complexity of propositional proof systems, 3-colorability

**AMS subject classifications.** 68R10, 03F20, 05C85

## 1. Introduction.

The Hajós calculus (or Hajós construction) is a simple, nondeterministic procedure for generating the class of graphs that are not $k$-colorable [Haj].[1] Mansfield and Welsh [MW] posed the problem of determining the complexity of this procedure; in particular, it is an open problem whether or not there exists a polynomial-sized Hajós construction for every non-3-colorable graph. Because graph 3-colorability is NP-complete, if there were polynomial-sized Hajós constructions of all non-3-colorable graphs, then NP = coNP, so we expect that the Hajós calculus is not polynomially bounded. However, there has been very little progress toward a proof of this conjecture, despite considerable effort.

The main result of this paper is a proof that the Hajós calculus is polynomially bounded if and only if extended Frege proof systems are polynomially bounded. This result links an open problem in graph theory to an important open problem in the complexity of propositional proof systems. It also shows that the complexity problem for the Hajós calculus is very difficult, since extended Frege systems are a very powerful class of proof systems for the propositional calculus and no techniques that appear adequate to prove superpolynomial lower bounds for them currently exist. In addition, we study a subsystem of the Hajós calculus, which is still powerful enough to generate all non-3-colorable graphs. Our results, together with recent lower bounds for bounded-depth Frege proofs [Ajt1], [PBI], [KPW], enable us to prove an exponential lower bound for this subsystem of the Hajós calculus.

In §2, we introduce graph calculus terminology and prove some facts about the Hajós calculus. In §3, we introduce propositional proof system terminology and prove some lemmas about extended Frege proof systems. In §4, we show how to transform

* Received by the editors November 16, 1992; accepted for publication (in revised form) May 6, 1994.

† Department of Computer Science, University of Pittsburgh, Pittsburgh, Pennsylvania 15260 (toni@cs.ucsd.edu). The research of this author was supported by a National Science Foundation postdoctoral fellowship and a University of California presidential fellowship.

‡ Departments of Philosophy and Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada (urquhart@cs.toronto.edu). The research of this author was supported by the Natural Sciences and Engineering Council of Ontario.

[1] This procedure was originally formulated due to its connection to the four-color conjecture. In particular, when $k = 4$, proving that all graphs generated by the Hajós calculus are not planar is equivalent to proving the four-color theorem.

the rules of the Hajós calculus to obtain a propositional proof system and, analogously, how to transform the extended Frege system into a graph calculus. We then show that there are efficient simulations between these two systems. In §5, we prove the main theorems, and finally in §6, we discuss the implications of these results for the graph-theoretical complexity problem.

## 2. Graph calculi and the Hajós calculus.

**2.1. Graph calculi.** The graphs in this paper are finite simple undirected graphs, that is, they contain no loops or multiple edges. The descriptions of the Hajós construction in the graph-theoretic literature assume that the construction operates on graphs where isomorphic copies of a graph are considered to be the same graph. However, it is more natural to think of operating on standard graphs, where each vertex is labeled with a particular positive integer, and therefore we view isomorphic graphs with different labels as different graphs. Accordingly, we define a graph $G$ to be a pair $(V, E)$, where $V$ is a finite set of positive integers and $E$ is a set of unordered pairs of elements of $V$; we write $e(G)$ for $E$ and $v(G)$ for $V$. Two graphs will be said to be *disjoint* if their vertex sets are disjoint. A $k$-coloring of a graph is an assignment of one of $k$ distinct colors to each of the vertices of the graph; the coloring is *proper* if no two adjacent vertices receive the same color, otherwise *improper.*

A *graph calculus* is a collection of initial graphs, together with a finite collection of rules that allow us to derive new graphs. We shall restrict our attention to calculi for the class of non-3-colorable graphs. Let $\mathcal{C}$ be a particular graph calculus. A *construction* of a graph $G$ in $\mathcal{C}$ is a sequence of graphs, ending with $G$, such that every graph in the sequence is either an initial graph or follows from previous graphs by one of the rules of $\mathcal{C}$. $\mathcal{C}$ is *sound* if every graph constructed by $\mathcal{C}$ is non-3-colorable. $\mathcal{C}$ is *complete* if every non-3-colorable graph can be constructed in $\mathcal{C}$. The Hajós calculus is an example of a sound and complete graph calculus. Let $\Gamma = \{G_1, G_2, \ldots, G_k\}$ be a set of graphs, and let $G$ be another graph, all with underlying vertex set $V$. Then $\Gamma$ *implies* $G$, written $\Gamma \Rightarrow G$, if for all 3-colorings  of $V$, if $c$ is an improper 3-coloring of every graph in $\Gamma$, then $c$ is also an improper 3-coloring of $G$. A graph calculus $\mathcal{C}$, is said to be *implicationally sound* if for all $\Gamma = \{G_1, G_2, \ldots, G_k\}$, $G$, if $G$ can be constructed in $\mathcal{C}$ from $\Gamma$, then $\Gamma \Rightarrow G$. A graph calculus $\mathcal{C}$, is said to be *implicationally complete* if for all $\Gamma = \{G_1, G_2, \ldots, G_k\}$, $G$ if $\Gamma \Rightarrow G$, then there is a construction in $\mathcal{C}$ of $G$ from the graphs in $\Gamma$. We will see in the subsequent section that the Hajós calculus is not implicationally sound but is implicationally complete. We will also define a complete subsystem of the Hajós calculus, $\mathcal{HC}^-$, which is implicationally sound.

The *size* of a graph $G$ is the number of its edges, that is, $|e(G)|$. The *size* of a graph calculus construction is the sum of the sizes of all graphs in the construction. The *length* of a graph calculus construction is the number of applications of rules in it. A graph calculus $\mathcal{C}$ is *polynomially bounded* if there exists a polynomial $p$ such that every non-3-colorable graph $G$ can be constructed in $\mathcal{C}$ with a construction of size at most $p(|e(G)|)$.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two graph calculus systems. Then $\mathcal{C}_1$ *p-simulates* $\mathcal{C}_2$ if there is a polynomial-time computable function $f$ so that for all graphs $G$, if $s$ is a graph construction of $G$ in $\mathcal{C}_2$, then $f(s)$ is a graph construction of $G$ in $\mathcal{C}_1$. $\mathcal{C}_1$ and $\mathcal{C}_2$ are *p-equivalent* if $\mathcal{C}_1$ *p*-simulates $\mathcal{C}_2$ and if $\mathcal{C}_2$ *p*-simulates $\mathcal{C}_1$.

**2.2. The Hajós calculus.** In this section, we will describe the Hajós calculus for $k = 3$; to obtain the Hajós calculus for a different value of $k$, simply substitute the complete graph, $K_{k+1}$, for the initial graph, $K_4$. The set of initial graphs in the Hajós calculus contains all graphs isomorphic to $K_4$. There are three rules for generating new graphs:

(1) (Vertex/edge introduction) Add (any number of) vertices and edges.

(2) (Join rule) Let $G_1$ and $G_2$ be disjoint graphs, $a_1$ and $b_1$ adjacent vertices in $G_1$, and $a_2$ and $b_2$ adjacent vertices in $G_2$. Construct the graph $G_3$ from $G_1 \cup G_2$ as follows. First, remove edges $[a_1, b_1]$ and $[a_2, b_2]$; then add an edge $[b_1, b_2]$; last, contract vertices $a_1$ and $a_2$ into a single vertex, named $a_1$.

(3) (Contraction rule) Contract two nonadjacent vertices into a single vertex, and remove any resulting duplicated edges; the single vertex can be either of the two original vertices.

Since we are using labeled rather than unlabeled graphs, it might seem necessary to include a duplication rule that allows the generation of any graph isomorphic to an already constructed graph. However, the duplication rule applied to a graph $G$ can be simulated by repeated applications of the contraction rule, applied to $G$ and a collection of isolated vertices. The simulating construction that results has size $O(|v(G)|^2)$, so the systems with and without the duplication rule are $p$-equivalent.

It is not too hard to see that any graph constructed by means of the above Hajós calculus procedure is not 3-colorable. Moreover Hajós showed [Haj] that the Hajós calculus is complete. The proof of this completeness theorem yields an algorithm that generates a Hajós construction for any non-3-colorable graph. The length of the construction, in the worst case, is exponential in the size of the graph. Mansfield and Welsh [MW] show that for a graph with less than $n^2/3$ edges, there is a Hajós construction of the graph with length at most $2^{n^2/3-|e(G)|+1} - 1$.

In earlier papers on the complexity of the Hajós calculus [MW], [HRT], a different definition of Hajós calculus is given. There, a Hajós construction of a graph $G$ is defined to be a construction of a subgraph of $G$ using only the rules (2) and (3). However, this alternative definition results in essentially the same complexity measure. (The unwritten proof of this result is due to Jason Brown.) Additionally, in earlier formulations, the size is taken to be the length rather than the size. However, the two measures are polynomially related for the formulation of the Hajós calculus employed here.

It will be convenient for our purposes to reformulate the Hajós calculus. The system $\mathcal{HC}$ has the same set of initial graphs as well as rules (1) and (3) of the Hajós calculus, but now rule (2) of the Hajós calculus is replaced by the following rule:

(2′) (Edge elimination rule) Let $G_1$ and $G_2$ be two graphs with common vertex set $\{v_1, \dots, v_n\}$ that are identical except that $G_1$ contains edges $[v_1, v_2]$ and $[v_2, v_3]$ and not $[v_1, v_3]$, whereas $G_2$ contains edges $[v_1, v_2]$ and $[v_1, v_3]$ and not $[v_2, v_3]$. Then from $G_1$ and $G_2$, we can construct the graph $G_3$, which is graph $G_1$ with edge $[v_2, v_3]$ removed.

CLAIM 1. *$\mathcal{HC}$ is p-equivalent to the Hajós calculus.*

*Proof.* Because the only rule that differs is rule (2), it suffices to show that the join rule of the Hajós calculus can be simulated by a construction in $\mathcal{HC}$, that the edge elimination rule (2′) of $\mathcal{HC}$ can be simulated by a construction in the Hajós calculus, and that in each case the series of simulating steps can be constructed in polynomial time.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two disjoint graphs, where $G_1$ has a distinguished edge $[a_1, b_1]$, $G_2$ has a distinguished edge $[a_2, b_2]$, and $G$ is the graph obtained from $G_1$ and $G_2$ by applying the join rule. To derive $G$ in $\mathcal{HC}$, we first apply rules (1) and (3) to obtain an isomorphic copy, $G_2^*$, of $G_2$ in which $a_2$ is replaced by

$a_1$. Then we apply rule (1) to $G_1$ to obtain a new graph, $G_1'$ with underlying vertex set $(V_1 \cup V_2) - \{a_2\}$; the edges of $G_1'$ are the edges of $G_1$ except for $[a_1, b_2]$, the edges of $G_2^*$, and the edge $[b_1, b_2]$. Similarly, we apply (1) to $G_2^*$ to obtain the new graph $G_2'$ where $G_2'$ has the same vertex set as $G_1'$, and the edge set is the same except that edge $[a_1, b_1]$ is replaced by edge $[a_1, b_2]$. We can now apply rule (2) of $\mathcal{HC}$ to $G_1'$ and $G_2'$ to obtain the desired graph, $G$.

Conversely, let $G_1$ and $G_2$ be two graphs that are identical except that $G_1$ contains edges $[v_1, v_2]$ and $[v_1, v_3]$ whereas $G_2$ contains edges $[v_1, v_2]$ and $[v_2, v_3]$. Let $G$ be the graph obtained by applying the edge elimination rule (2′). To derive $G$ in the Hajós calculus, first apply rules (1) and (3) to generate an isomorphic copy, $G_2^*$, of $G_2$ disjoint from $G_1$; if $v$ is a vertex of $G_2$, then $v^*$ is the corresponding vertex of $G_2^*$. Second, apply the join rule to $G_1$ and $G_2^*$ to obtain a new graph, $G'$, where vertices $v_3$ and $v_3^*$ are contracted to a single vertex, $v_3$, and that contains the edge $[v_1, v_2^*]$ but does not contain edges $[v_2^*, v_3^*]$ or $[v_1, v_3]$. $G'$ contains two copies of each vertex other than vertex $v_3$; by contracting vertices $v$ and $v^*$ (using rule (3)), we obtain the desired graph, $G$. □

DEFINITION. *Let the graph system $\mathcal{HC}^-$ be the system $\mathcal{HC}$ without the contraction rule* (3).

In [Be, pp. 352–353], it is shown that $\mathcal{HC}$ is complete. The proof actually shows the stronger result that $\mathcal{HC}^-$ is complete. The following theorem is a new and stronger form of completeness for $\mathcal{HC}$ than in the earlier proof by Hajós. The proof of this theorem follows from our more general Theorem 3 by setting $V'$ equal to $V$.

THEOREM 2. *$\mathcal{HC}^-$ is implicationally complete.*

DEFINITION. *Let $\Gamma = \{G_1, \ldots, G_q\}$, and let $G_1, \ldots, G_q$ and $G$ be graphs over vertices $V$, $|V| = n$. For $V' \subseteq V$, let $G^{V'}$ denote the subgraph of $G$ over the vertices in $V'$. The implication $\Gamma \Rightarrow G$ is $V'$-local if there exists a subset $V' \subseteq V$ such that* (1) *for all $i \leq q$, the set of edges in $G_i$ that do not lie inside $V'$ equals the set of edges of $G$ that do not lie inside $V'$, and* (2) *if $c$ is a 3-coloring to $V'$ and $c$ is a proper 3-coloring of $G^{V'}$, then $c$ is also a proper 3-coloring of $G_j^{V'}$, for some $j \leq q$. The implication will be called $V'$-local when the underlying vertex set is $V'$.*

THEOREM 3. *Let $\Gamma = \{G_1, \ldots, G_q\}$, and let $G_1, \ldots, G_q$ and $G$ be graphs over vertices $V$, $|V| = n$. If there exists $V' \subseteq V$, $|V'| \geq 4$, such that $\Gamma \Rightarrow G$ is $V'$-local, then there is a $\mathcal{HC}^-$ construction of $G$ from $\Gamma$ with size at most $O(2^{|V'|}n^2)$.*

*Proof.* Let $G_1, G_2, \ldots, G_q$ and $G$ be graphs over vertex set $V$, and let $G_1, \ldots, G_q \Rightarrow G$. A *triplet* is a subgraph consisting of three vertices and a single edge between two of the three vertices. For a fixed $V$, $V' \subseteq V$, $|V'| \geq 4$, $G_1, \ldots, G_q$, we will show that if $G$ is a graph over $V$ such that $G_1, \ldots, G_q \Rightarrow G$ and the implication is $V'$-local, then there exists a construction (in $|V|$) of $G$ from $G_1, \ldots, G_q$ of size $O(2^{|V'|}n^2)$.

The proof is by downward induction on the number of edges in $G^{V'}$. If all edges are present in $G^{V'}$, then $G^{V'}$ and, hence, $G$ must contain a $K_4$ subgraph and thus, can be constructed from $K_4$ by application of rule (1). For the inductive step, assume that we have proven the theorem for all graphs, $H$, with underlying vertices $V$, such that (1) $G_1, \ldots, G_q \Rightarrow H$, and the implication is $V'$-local, and (2) the number of edges in $H^{V'}$ is greater than $m$. Now fix $G$ such that $G_1, \ldots, G_q \Rightarrow G$, the implication is $V'$-local, and $G^{V'}$ has $m$ edges. There are two cases to consider: either $G^{V'}$ contains a triplet, or there are no triplets in $G^{V'}$. If $G^{V'}$ does not contain any triplets, then $G^{V'}$ is a complete $k$-partite graph, for some $k$. (This follows because if there are no triplets, then the relation "is not adjacent" is an equivalence relation.) If $k > 3$, then $G^{V'}$, and hence also $G$, contain a $K_4$ subgraph and can therefore be constructed from $K_4$ by rule (1). Otherwise $k \leq 3$. In this case, we will show that $G$ contains some $G_i$ as a subgraph. Label the maximal independent sets of $G^{V'}$ by $X$, $Y$, and $Z$. Assume for the sake of contradiction that $G$ does not contain any of $G_1, \dots, G_q$ as subgraphs. Then each $G_i$ must contain some edge that is contained in either $X$, $Y$, or $Z$. Now consider the 3-coloring of $G^{V'}$ that assigns to all vertices of $X$ the color red, to all vertices of $Y$ the color blue, and to all vertices of $Z$ the color green. Then this coloring properly colors $G^{V'}$ but does not properly color any of $G_1^{V'}, \dots, G_q^{V'}$. But this contradicts our assumption that $G_1, \dots, G_q \Rightarrow G$ is $V'$-local. Thus, some $G_i$ must be contained within $G$, and it follows that $G$ can be constructed from $G_i$ by adding edges.

The second case is when $G^{V'}$ contains at least one triplet. Consider a triplet, $T$, of $G^{V'}$, with vertices $v_1, v_2, v_3$ and edge $(v_1, v_2)$. Let $G^a$ be the graph $G$ plus the additional edge $(v_2, v_3)$, and let $G^b$ be the graph $G$ plus the additional edge $(v_1, v_3)$. Because $G_1, \dots, G_q \Rightarrow G$ and $G \Rightarrow G^a$, it follows that $G_1, \dots, G_q \Rightarrow G^a$. By this fact, and because $G^a$ has more than $m$ edges, we can apply the inductive hypothesis to obtain a construction of $G^a$ from $G_1, \dots, G_q$. Similarly, we can construct $G^b$ from $G_1, \dots, G_q$. Now, by the edge elimination rule (2′), we can construct $G$ from $G^a$ and $G^b$.

The number of intermediate graphs constructed is $O(2^{|V'|})$, and each is of size at most $O(|V|^2)$. Thus, the total size of the construction is $O(2^{|V'|}n^2)$. □

It is interesting to note that $\mathcal{HC}$ is not implicationally sound because rule (3) is not sound. However, $\mathcal{HC}^-$ is implicationally sound.

## 3. Propositional proof systems.

A *propositional proof system* is a collection of initial propositional formulas (axioms), together with a set of rules that allow us to derive new propositional formulas. The propositional proof systems we consider are systems for showing formulas to be tautologies. A propositional proof system is a *Frege system* if its axioms consist of all instances of a finite number of tautologies, and it has a finite number of inference rules of a certain form that are sound (i.e., preserve truth). We shall also consider *extended Frege systems*, in which it is possible to introduce new sentence letters as abbreviations for complex formulas. For a more in depth treatment of propositional proof systems and their relative efficiency, see [CR].

Let $\mathcal{P}$ be a fixed propositional proof system. We say that $\mathcal{P}$ is *sound* if all of its axioms are tautologies and its rules of inference preserve truth (for each truth assignment). If $\mathcal{P}$ is sound and $A_1, \dots, A_n$ and $B$ are formulas of $\mathcal{P}$, then a *proof of $B$ from $A_1, \dots, A_n$ in $\mathcal{P}$* is a sequence of formulas, each of which is either an axiom of $\mathcal{P}$, one of the formulas $A_1, \dots, A_n$, or derived from earlier formulas in the sequence by one of the rules of $\mathcal{P}$ and in which the last formula is $B$. We write $A_1, \dots, A_n \vdash_{\mathcal{P}} B$ if there is a proof of $B$ from $A_1, \dots, A_n$ in $\mathcal{P}$. The *size* of a proof is defined to be the number of occurrences of symbols in it. If $\Gamma \cup \{A\}$ is a set of formulas, then we write $\Gamma \models A$ if $A$ is a logical consequence of the set of formulas $\Gamma$. A proof system $\mathcal{P}$ is *complete* if every tautology has a proof in $\mathcal{P}$; $\mathcal{P}$ is *implicationally complete* if $\Gamma \vdash_{\mathcal{P}} B$ whenever $\Gamma \models B$.

Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two propositional proof systems that share a common language. The system $\mathcal{P}_1$ *p-simulates* $\mathcal{P}_2$ if there is a polynomial-time computable function $f$ so that for all formulas $A$, if $P$ is a proof in $\mathcal{P}_2$ of $A$, then $f(P)$ is a proof of $A$ in $\mathcal{P}_1$. If $\mathcal{P}_1$ $p$-simulates $\mathcal{P}_2$, then there is a polynomial $p$ so that if there is a proof of $A$ in $\mathcal{P}_2$ of size $m$, then there is also a proof of $A$ in $\mathcal{P}_1$ of size at most $p(m)$. Two proof systems are *p-equivalent* if they $p$-simulate each other.

Several of the proof systems we consider in this paper operate with formulas in disjunctive normal form. A *DNF formula* is a finite disjunction of a set of *terms*, each of which is a conjunction of a set of literals (variables or their negations). The complement of a literal $l$ is written as $\bar{l}$. A term will be written in the form $\bigwedge X$ where $X$ is a set of literals, or by juxtaposing literals, as in $l_1 l_2 \dots l_k$. Disjunctions are written as $\bigvee X$, or in the form $l_1 \vee \cdots \vee l_k$. If $X$ is a set of terms and $l$ a literal, then $Xl$ will stand for the set of all terms $Al$, where $A$ is a term in $X$.

If $A$ is a formula in conjunctive normal form, then $\sim A$ is defined to be the formula in disjunctive normal form equivalent to the negation of $A$, that is, the formula resulting from $A$ by replacing $\vee$ by $\wedge$, $\wedge$ by $\vee$, and literals by their complements. If $l, m, n$ are literals, then the notation $l \leftrightarrow (m \vee n)$ will be used as an abbreviation for $(\bar{l} \vee m \vee n) \wedge (\overline{m} \vee l) \wedge (\overline{n} \vee l)$ and $(A_1, \dots, A_k) \supset B$ as an abbreviation for $\sim A_1 \vee \cdots \vee \sim A_k \vee B$.

If $A, B$ are formulas, then $A[B/p]$ is defined to be the formula that results from $A$ by substituting $B$ for every occurrence of the variable $p$ in $A$. If $\Gamma$ is a set of formulas, then $\Gamma[B/p]$ is the set of all formulas $A[B/p]$ for $A$ in $\Gamma$.

Let $f$ be a DNF formula. Then the formula $f[1/p]$ denotes the formula obtained from $f$ by removing all terms that contain $\overline{p}$ and then removing all occurrences of $p$. If the resulting formula is the empty disjunction, then $f[1/p]$ is defined to be $(p \wedge \overline{p})$; if the resulting formula contains an empty conjunction, then $f[1/p]$ is defined to be $(p \vee \overline{p})$. Similarly, the formula $f[0/p]$ denotes the formula obtained from $f$ by removing all terms that contain $p$ and then removing all occurrences of $\overline{p}$. If the resulting formula is the empty disjunction, then $f[0/p]$ is defined to be $(p \wedge \overline{p})$; if the resulting formula contains an empty conjunction, then $f[0/p]$ is $(p \vee \overline{p})$. If $f$ is a *CNF* formula, then $f[1/p]$ and $f[0/p]$ are defined similarly. Last, if $\Gamma$ is a set of formulas, then $\Gamma[0/p]$ ($\Gamma[1/p]$) is the set of all formulas $A[0/p]$ ($A[1/p]$), for $A \in \Gamma$.

**3.1. A depth-2 Frege system.** The system $\mathcal{F}_2$ is a proof system for DNF formulas, where a term in a DNF formula will be viewed as an AND of an ordered set of literals. The proof system $\mathcal{F}_2$ has a single axiom schema, $p \vee \overline{p}$, and the following five rules. In applying the rules of $\mathcal{F}_2$ and the other systems that we consider the associative and commutative laws for disjunction will be applied tacitly. We will also assume the following generalized commutativity rule for conjunction: $(l_1 \wedge l_2 \wedge \cdots \wedge l_q) \vee B \Rightarrow (\pi(l_1) \wedge \pi(l_2) \wedge \cdots \wedge \pi(l_q)) \vee B$, where $\pi$ is a bijection from $\{l_1, \dots, l_q\}$ to $\{l_1, \dots, l_q\}$.

(R0) $A \vee A \vee B \Rightarrow A \vee B$ and $A \vee A \Rightarrow A$,
(R1) $A \Rightarrow A \vee B$,
(R2) $AB \vee C \Rightarrow A \vee C$,
(R3) $(A \vee C), (B \vee C) \Rightarrow AB \vee C$,
(R4) $A \vee l, B \vee \bar{l} \Rightarrow A \vee B$.

THEOREM 4. *$\mathcal{F}_2$ is implicationally complete for DNF formulas.*

*Proof.* We prove that if $\Gamma \models A$, then $\Gamma \vdash_{\mathcal{F}_2} A$ by induction on the number of variables in $\Gamma \cup \{A\}$. We will need the following claim, which is proven by induction on the length of the derivation of $B$ from $\Gamma \cup \{A\}$.

CLAIM 4A. *If $\Gamma, A \vdash_{\mathcal{F}_2} B$, then $\Gamma, A \vee C \vdash_{\mathcal{F}_2} B \vee C$.*

If there is exactly one variable in $\Gamma \cup \{A\}$ and if $\Gamma \models A$, then there are two cases to consider. (1) If $A$ is equivalent to 0, $p$, or $\neg p$, then it can be checked that in all cases, we can derive $A$ from $\Gamma$. (2) Otherwise, $A$ is equivalent to 1, in which case it is of the form $(pp...p) \vee (\overline{pp}...\overline{p}) \vee B$ and therefore follows from the axiom plus rules (R1) and (R3).

Assume that the theorem holds for $n$ variables, that $\Gamma \cup \{A\}$ contains $n+1$ variables, and $\Gamma \models A$. Let $p$ be a variable occurring in $A$. Then without loss of generality, $A$ can be written in the form $\bigvee Xpp...p \vee \bigvee Y\overline{p}\overline{p}...\overline{p} \vee \bigvee Z$, where $p$ and $\overline{p}$ do not occur in $X$, $Y$, or $Z$. By assumption,

$$\Gamma[1/p] \vdash_{\mathcal{F}_2} \bigvee X \vee \bigvee Z; \qquad \Gamma[0/p] \vdash_{\mathcal{F}_2} \bigvee Y \vee \bigvee Z.$$

By applying the rules (R2) and (R4), the set of formulas $\Gamma[1/p]$ can be deduced from $\Gamma \cup \{p\}$; similarly, $\Gamma[0/p]$ can be deduced from $\Gamma \cup \{\overline{p}\}$. Thus, by (R1) and (R3),

$$\Gamma, p \vdash_{\mathcal{F}_2} \bigvee Xp \vee \bigvee Z; \qquad \Gamma, \overline{p} \vdash_{\mathcal{F}_2} \bigvee Y\overline{p} \vee \bigvee Z.$$

Hence, by the preceding claim,

$$\Gamma, p \vee \overline{p} \vdash_{\mathcal{F}_2} \bigvee Xp \vee \bigvee Z \vee \overline{p}; \qquad \Gamma, p \vee \overline{p} \vdash_{\mathcal{F}_2} \bigvee Y\overline{p} \vee \bigvee Z \vee p.$$

Since $p \vee \overline{p}$ is an axiom, $\Gamma \vdash_{\mathcal{F}_2} \bigvee Xp \vee \bigvee Y\overline{p} \vee \bigvee Z$ by (R4) and (R0). Then by repeated application of (R0), (R2), and (R3), we have $\Gamma \vdash_{\mathcal{F}_2} A$. □

**3.2. A depth-$d$ Frege system.** By adding a few more rules to the depth-2 Frege system, $\mathcal{F}_2$, we can define a more general Frege system, $\mathcal{F}$, over the basis AND, OR and NOT. We define the rules of this system to be the rules (R0)–(R4) of $\mathcal{F}_2$, plus the following rules (R5)–(R9). The symbol $\Leftrightarrow$ indicates that the rule may be applied either from left to right ($\Rightarrow$) or from right to left ($\Leftarrow$). In all rules (R0)–(R9), the formulas $A$, $B$, and $C$ are arbitrary formulas over the basis $\vee$, $\wedge$, and $\neg$, where AND and OR are unbounded-fanin boolean functions and NOT is defined as usual.

(R5) $A \wedge (B_1 \vee \cdots \vee B_k) \Leftrightarrow (A \wedge B_1) \vee \cdots \vee (A \wedge B_k)$,
(R6) $A \vee (B_1 \wedge \cdots \wedge B_k) \Leftrightarrow (A \vee B_1) \wedge \cdots \wedge (A \vee B_k)$,
(R7) $\neg(A_1 \vee \cdots \vee A_k) \Leftrightarrow \neg A_1 \wedge \cdots \wedge \neg A_k$,
(R8) $\neg(A_1 \wedge \cdots \wedge A_k) \Leftrightarrow \neg A_1 \vee \cdots \vee \neg A_k$,
(R9) $\neg\neg A \Leftrightarrow A$.

The *depth* of a formula is the depth of the boolean tree that represents the formula. We define a depth-$d$ proof of a formula, $A$, to be a proof in the above Frege system, where each formula occurring in the proof has depth at most $d$. Alternatively, the *depth-$d$ Frege system*, $\mathcal{F}_d$, is the system $\mathcal{F}$, where each formula is restricted to have depth at most $d$.

**3.3. Proof systems with extension.** Given a proof system, a natural and powerful way to extend it is to add a rule by which a complex formula can be abbreviated by a single variable.

Let $\mathcal{S}$ be any standard Frege system where no restrictions are placed on the depth of formulas (i.e., the system $\mathcal{F}$). The *extended version of* $\mathcal{S}$, or $\mathcal{ES}$, is the system that results from $\mathcal{S}$ by allowing as steps in derivations lines of the form $p \leftrightarrow A$, where $p$ is a variable that does not appear in $A$, nor does $p$ appear in earlier lines, in the assumptions, or the conclusion of the derivation. Such a system is an *extended Frege system.* Cook and Reckhow [CR] show that any two extended Frege systems are $p$-equivalent, so the results of this paper do not depend on the formal details of a particular system.

In the case of the system $\mathcal{F}_2$, it is more convenient to introduce a form of the extension rule similar to that originally used by Tseitin [Tse] in the context of resolution systems. Let $V$ be a set of variables and $p_1, \ldots, p_k$ a set of new variables disjoint from those in $V$. Then $D_1, \ldots, D_k$ is an *extension sequence* for $V$ if for each $i$, $D_i$ is $p_i \leftrightarrow (l_1 \vee l_2 \vee \cdots \vee l_q)$, where $l_1, \ldots, l_q$ are literals that involve variables from $V \cup \{p_1, \ldots, p_{i-1}\}$. We define an $\mathcal{EF}_2$ *proof* of a DNF formula $A$ to be a proof in $\mathcal{F}_2$ of the formula $(D_1, \ldots, D_k) \supset A$, where $D_1, \ldots, D_k$ is an extension sequence for the variables occurring in $A$.

Although $\mathcal{EF}_2$ only allows proofs of formulas in DNF, $\mathcal{EF}_2$ can be considered as a general proof system for propositional formulas by using the idea of extension. Let $A$ be a propositional formula; for definiteness, let us take $A$ to contain only the connectives $\neg$ and $\vee$. Associate a literal $l_B$ with each subformula $B$ of $A$ so that (1) the variables in $A$ are associated with themselves; (2) if $B$ has the associated literal $l_B$, then $\neg B$ has $\overline{l_B}$ as its associated literal; and (3) distinct subformulas are associated with distinct literals. Then the *definitional set* for $A$, $\mathrm{Def}(A)$, is the set of all formulas of the form $l_{B \vee C} \leftrightarrow (l_B \vee l_C)$, where $B \vee C$ is a subformula of $A$. Then we define an $\mathcal{EF}_2$ proof of the formula $A$ to be a proof in $\mathcal{EF}_2$ of $(\bigwedge \mathrm{Def}(A) \supset l_A)$.

LEMMA 5. *The system $\mathcal{EF}_2$ p-simulates any extended Frege system.*

*Proof.* Let $\mathcal{ES}$ be a fixed extended Frege system, $d$ a derivation of $A$ in $\mathcal{ES}$, and $\mathrm{Def}(d)$ the union of all the definitional sets $\mathrm{Def}(B)$, where $B$ is a line of $d$. It can be shown by induction on the length of $d$ that for every line $B$ in $d$, there is a derivation in $\mathcal{F}_2$ of $\mathrm{Def}(d) \supset l_B$; hence there is a derivation $d'$ in $\mathcal{EF}_2$ of $A$. The new derivation $d'$ can be constructed in polynomial time from the old derivation $d$, so that $\mathcal{EF}_2$ $p$-simulates $\mathcal{ES}$. This proof is essentially the same as the proof by Cook and Reckhow [CR] of the corresponding result for extended resolution. □

**3.4. Systems with substitution rules.** Another natural and powerful rule used in many propositional proof systems is the rule of substitution: Given $A$ derive $A[B/p]$, where $p$ is a variable, $B$ any formula. This rule is unsound (except if $A$ is required to be a tautology), so systems including this rule no longer qualify as Frege systems. If $\mathcal{G}$ is a Frege system, then we write $\mathcal{SG}$ for the system which results from $\mathcal{G}$ by adding the rule of substitution. Such systems are called *Frege systems with substitution* by Cook and Reckhow [CR], who prove that any such system can $p$-simulate any extended Frege system.[2]

A weak form of the substitution rule is the rule that allows the substitution of the constants 0 and 1 for all occurrences of a propositional variable. This form of the rule we shall call the *$\{0,1\}$-substitution rule.* If $\mathcal{G}$ is a Frege system, we write $\mathcal{S}_{0,1}\mathcal{G}$ for the system that results by adding the rules: $A \Rightarrow A[0/p]$ and $A \Rightarrow A[1/p]$ to $\mathcal{G}$. If we do not have the constants 0 and 1 in our system, then $A[0/p]$ and $A[1/p]$ denote the formulas obtained from $A$ as defined earlier. The following lemma, due to Buss [Bu], states that in the context of a Frege system this weak form of the rule is (up to a polynomial) as strong as the unrestricted form of the rule. For completeness, we provide a proof of this lemma.

LEMMA 6. *The system $\mathcal{S}_{0,1}\mathcal{F}_2$ p-simulates $\mathcal{EF}_2$.*

*Proof.* By definition, an $\mathcal{EF}_2$ proof of a formula $A$ is a proof in $\mathcal{F}_2$ of $(D_1, \ldots, D_k) \supset A$, where $D_1, \ldots, D_k$ is an extension sequence for the set of variables in $A$. Thus it is sufficient to prove that if we are given a proof in $\mathcal{F}_2$ of $(D_1, \ldots, D_k) \supset A$, we can eliminate the antecedent of the conditional efficiently, using the {0,1}-substitution rule.

Let $D_k$ be $p_k \leftrightarrow (l_1 \vee l_2 \vee \cdots \vee l_q)$. Then from $(D_1, \ldots, D_k) \supset A$ by substituting

[2] Somewhat surprisingly, the converse simulation also holds, a fact first proved by Dowd [Do] and later in [KP].

1 for $p_k$ we can derive the formula $(D_1, \ldots, D_{k-1}, (l_1 \vee \cdots \vee l_q)) \supset A$; similarly, by substituting 0 for $p_k$ we obtain the formula $(D_1, \ldots, D_{k-1}, (\overline{l_1} \wedge \cdots \wedge \overline{l_q})) \supset A$. By repeated application of (R2), (R0), and (R4), we can then derive $(D_1, \ldots, D_{k-1}) \supset A$. The elimination of the hypothesis $D_k$ takes steps of total size $O(r)$, where $r$ is the length of the formula $(D_1, \ldots, D_k) \supset A$, so that the formula $A$ can be derived in $\mathcal{S}_{0,1}\mathcal{F}_2$ by adding extra steps of total size $O(kr)$. □

Another restricted form of the substitution rule is the renaming rule: given $A$ derive $A[p/q]$, where $p$, $q$ are propositional variables. If $\mathcal{G}$ is a Frege system, the system that results from $\mathcal{G}$ by adding the renaming rule will be denoted by $\mathcal{RG}$. It is a surprising result of Buss [Bu4] that in the context of Frege systems the renaming rule is as powerful as the $\{0,1\}$-substitution rule (up to a polynomial).

LEMMA 7. *The system $\mathcal{RF}_2$ p-simulates $\mathcal{S}_{0,1}\mathcal{F}_2$.*

*Proof.* The proof presented here is due to Steve Cook. We show how, given a proof of $A$ in $\mathcal{F}_2$ and $p$ a variable in $A$, we can derive $A[0/p]$ in $\mathcal{RF}_2$ in a relatively efficient way. Let $p_1, \ldots, p_k$ be the variables in $A$, and let $p_i$ be the variable to be eliminated. Let the length of $A$ be $r$. Then from $A$ we can derive $p_i \vee A[0/p_i]$ by (R0), (R1), and (R2); hence by $k-1$ applications of renaming we obtain the formulas $p_1 \vee A[0/p_i]$, $p_2 \vee A[0/p_i]$, $\ldots$, $p_k \vee A[0/p_i]$. Since $A$ is a tautology, we can derive the formula $\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_k \vee A[0/p_i]$ by a subsidiary derivation of length $O(kr)$. Now by $k$ applications of rule (R4), we obtain $A[0/p_i]$, as desired. The derivation of $A[0/p_i]$ from $A$ has size $O(r^2)$. An exactly similar process allows the derivation of the formula $A[1/p_i]$, so the proof of simulation is complete. □

*Note.* In this section, we have defined all of our proof systems to be depth-2 systems for proving DNF tautologies. However, in the following sections it is more convenient to think in terms of unsatisfiable formulas in CNF. It is easy to interpret our systems as refutation systems for CNF formulas by simply replacing each AND by an OR and each OR by an AND in each rule and axiom of the system. In what follows, we will view $\mathcal{EF}_2$, $\mathcal{RF}_2$, $\mathcal{F}$, and $\mathcal{F}_2$ as refutation systems for CNF formulas. In particular, the depth-2 refutation system, $\mathcal{F}_2$, for deriving unsatisfiable CNF formulas consists of one axiom, $p \wedge \overline{p}$, and the following rules:

(R0′) $A \wedge A \wedge B \Rightarrow A \wedge B$;
(R1′) $A \Rightarrow A \wedge B$;
(R2′) $(A \vee B) \wedge C \Rightarrow A \wedge C$;
(R3′) $(A \wedge C), (B \wedge C) \Rightarrow (A \vee B) \wedge C$;
(R4′) $A \wedge l, B \wedge \overline{l} \Rightarrow A \wedge B$.

In addition, we have the associativity and commutativity rules for AND as well as the following generalized rule of commutativity for OR: $(l_1 \vee \cdots \vee l_q) \wedge A \Rightarrow (\pi(l_1) \vee \cdots \vee \pi(l_q)) \wedge A$, where $\pi$ is a bijection from $\{l_1, \ldots, l_q\}$ to $\{l_1, \ldots, l_q\}$.

## 4. Simulation results.

Using the well-known reductions between CNF unsatisfiability and graph non-3-colorability, we can translate the rules and axioms of the Hajós calculus to obtain a propositional refutation system, and similarly, we can translate the rules and axioms of $\mathcal{EF}_2$ to obtain a graph calculus. We will first review the translations between propositional formulas and graphs. Then we show that the Hajós calculus can $p$-simulate $\mathcal{EF}_2$, translated into a graph calculus, and conversely, that $\mathcal{EF}_2$ can $p$-simulate the Hajós calculus, translated into a propositional refutation system.

### 4.1. Translation from propositional formulas to graphs.

We use a reduction from the theory of NP-completeness to show the similarity between graph calculus systems and propositional refutation systems. Garey, Johnson, and Stockmeyer

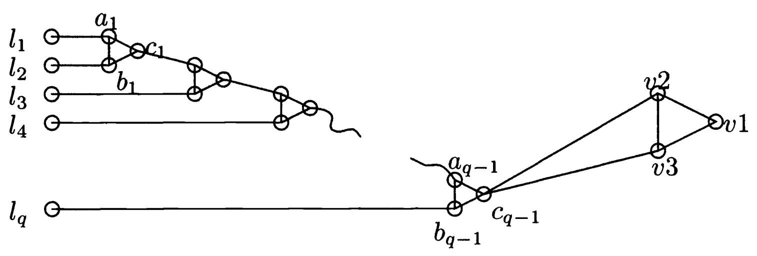


FIG. 1. *The subgraph $G_{C_i}$ corresponding to $C_i$.*

[GJS] showed that 3SAT is reducible to 3-colorability, by obtaining a polynomial-time, many–one transformation from 3CNF formulas to graphs. Here, we slightly modify their construction so that it is a transformation from arbitrary CNF formulas to graphs. Let $f$ be a CNF formula, $f = C_1 \wedge C_2 \wedge \cdots \wedge C_p$, with underlying variables $x_1, \ldots, x_n$. For notational convenience, let $C_i = (l_1 \vee l_2 \vee \cdots \vee l_q)$. Then Graph($f$) consists of the following nodes, $N$:

$$N = \{v1, v2, v3\} \cup \{xi, \overline{xi} \mid 1 \le i \le n\}$$
$$\cup \{a_j^i, b_j^i, c_j^i \mid 1 \le i \le p, 1 \le j \le q-1\}.$$

The set $E$ of edges for Graph($f$) is given by

$$E = \{[v1, v2], [v2, v3], [v1, v3]\} \cup \{[xi, \overline{xi}] | 1 \le i \le n\}$$
$$\cup \{[v3, xi], [v3, \overline{xi}] | 1 \le i \le n\}$$
$$\cup \{G_{C_i} \mid 1 \le i \le q\},$$

where for a particular clause, $C_i = (l_1 \vee l_2 \vee \cdots \vee l_q)$ in $f$, the subgraph $G_{C_i}$ is shown in Fig. 1. The superscript $i$ has been omitted for simplicity. When $C_i$ is just a single literal, $l$, then the construction for $G_{C_i}$ simply puts $l$ in place of $c_{q-1}$.

A key property of the graph $G$ is: for each $C_i = (l_1 \vee l_2 \vee \cdots \vee l_q)$, a proper 3-coloring to $l_1, \ldots, l_q$ can be extended to a proper 3-coloring to the underlying vertices of $G_{C_i}$ if and only if at least one of $l_1, \ldots, l_q$ has the same color as $v1$. The observation leads to the following lemma, which was proven in [GJS].

THEOREM 8. *$f$ is satisfiable if and only if* Graph($f$) *is 3-colorable.*

**4.2. Translation from graphs to propositional formulas.** Conversely, if $G$ is any graph, then there exists a propositional CNF formula, Form($G$) with the property that $G$ is 3-colorable if and only if Form($G$) is satisfiable. Let $G = (V, E)$ be a graph. We define a propositional statement, Form($G$), with underlying variables $\{R_x, B_x, G_x \mid x \in V\}$, which expresses "$G$ is 3-colorable" as follows:

$$\bigwedge_{x \in V} (\overline{R}_x \vee \overline{B}_x \vee \overline{G}_x) \cdot (R_x \vee B_x \vee G_x)$$
$$\cdot (R_x \vee \overline{B}_x \vee \overline{G}_x) \cdot (\overline{R}_x \vee \overline{B}_x \vee G_x) \cdot (\overline{R}_x \vee B_x \vee \overline{G}_x)$$
$$\bigwedge_{[x,y] \in E} (\overline{R}_x \vee \overline{R}_y) \cdot (\overline{G}_x \vee \overline{G}_y) \cdot (\overline{B}_x \vee \overline{B}_y).$$

LEMMA 9. *$G$ is 3-colorable if and only if* Form($G$) *is satisfiable.*

**4.3. The simulation lemmas.** By using the translations between propositional formulas and graphs described above, we can translate the Hajós calculus to obtain a new propositional refutation system. Define Form($\mathcal{HC}$) to be the propositional refutation system obtained by translating the rules and axioms of $\mathcal{HC}$. Specifically, Form($\mathcal{HC}$) has one axiom, Form($K_4$), and three rules, obtained by translating the three rules of $\mathcal{HC}$. Similarly, define Form($\mathcal{HC}^-$) to be the propositional refutation system obtained by translating the rules and axioms of $\mathcal{HC}^-$. Analogously, let Graph($\mathcal{F}_2$) and Graph($\mathcal{EF}_2$) be the graph calculus system obtained by translating the rules and axioms of $\mathcal{F}_2$ and $\mathcal{EF}_2$, respectively. Then we have the following simulation lemmas.

LEMMA 10. *$\mathcal{F}_2$ p-simulates* Form($\mathcal{HC}^-$).

*Proof.* Form($\mathcal{HC}^-$) has one axiom, Form($K_4$), and two rules. Because all rules and axioms of Form($\mathcal{HC}^-$) are sound, by the implicational completeness of $\mathcal{F}_2$ (Theorem 4), there exist $\mathcal{F}_2$ proofs of Form($K_4$) as well as the two rules of Form($\mathcal{HC}^-$). Furthermore, because each axiom and rule of Form($\mathcal{HC}^-$) involves only a constant number of symbols, the size of the proofs in $\mathcal{F}_2$ are polynomial in the size of the original axiom or rule instance in Form($\mathcal{HC}^-$). □

LEMMA 11. *$\mathcal{EF}_2$ p-simulates* Form($\mathcal{HC}$).

*Proof.* We will show that $\mathcal{RF}_2$ *p*-simulates Form($\mathcal{HC}$). Then because $\mathcal{EF}_2$ *p*-simulates $\mathcal{EF}$ and $\mathcal{EF}$ *p*-simulates $\mathcal{RF}_2$, it follows that $\mathcal{EF}_2$ *p*-simulates Form($\mathcal{HC}$). Because Form($\mathcal{HC}^-$) is a subsystem of Form($\mathcal{HC}$) and $\mathcal{F}_2$ is a subsystem of $\mathcal{RF}_2$, we only need to show how the one additional rule of $\mathcal{HC}$, the contraction rule, can be simulated by $\mathcal{RF}_2$. Since the contraction rule is just the renaming of the variables, it can be simulated by the renaming rule. □

LEMMA 12. *$\mathcal{HC}$ p-simulates* Graph($\mathcal{EF}_2$).

*Proof.* We will show that $\mathcal{HC}$ *p*-simulates Graph($\mathcal{RF}_2$). By Lemmas 6 and 7, $\mathcal{RF}_2$ *p*-simulates $\mathcal{EF}_2$; therefore Graph($\mathcal{RF}_2$) *p*-simulates Graph($\mathcal{EF}_2$), and it follows that $\mathcal{HC}$ *p*-simulates Graph($\mathcal{EF}_2$). Let $P$ be a construction of some non-3-colorable graph, $G$, in Graph($\mathcal{RF}_2$). We will prove that there exists a polynomial-sized construction of $G$, by induction on the number of graphs in $P$. If $P$ contains just one graph, then $P = G$, and $G$ must be the graph Graph($p \wedge \overline{p}$). In this case, it is easy to see that this graph contains a $K_4$ subgraph on the vertices $p$, $\overline{p}$, $v2$, and $v3$ and, hence, can be constructed easily in $\mathcal{HC}$. Otherwise, assume that we can simulate the first $k$ lines of $P$, by a polynomial-sized construction in $\mathcal{HC}$. The next graph, Graph($f'$), is either an axiom, or it follows from previous graphs by a rule. If Graph($f'$) is an axiom, then it can be constructed in $\mathcal{HC}$ as in the base case.

Otherwise, Graph($f'$) follows from previous graphs by a rule. We will show that in all cases, Graph($f'$) can be constructed efficiently in $\mathcal{HC}$. If $f'$ follows from $f$ by renaming, then Graph($f'$) can be constructed from Graph($f$) by rules (1) and (3). If $f' = A \wedge B$ follows from $f = A \wedge A \wedge B$ by (R0$'$), then Graph($f'$) can be constructed from Graph($f$) by repeated application of the contraction rule. (When $A$ is a single literal $l$, then Graph($f'$) is equal to Graph($f$), so there is nothing to do.) If $f'$ follows from a previous formula, $f$ by (R1$'$), then Graph($f'$) can be constructed from Graph($f$) by adding additional vertices and edges. Also, the associativity and commutativity laws for AND hold because Graph($(AB)C$) = Graph($A(BC)$), and also Graph($AB$) = Graph($BA$), up to renaming of vertices. The remaining cases (R2$'$), (R3$'$), (R4$'$), and the commutativity laws for OR are more complicated. The following definition and lemmas facilitate the proof of these cases.

DEFINITION. *Let $G$ be a graph with underlying vertices $V \cup \{l_1, l_2, \dots, l_k\} \cup \{v1, v2, v3\}$. Then we define the graph $G[l_i = 1]$ to be the graph $G$ plus the additional* *edges $[l_i, v2]$ and $[l_i, v3]$. Any proper 3-coloring of $G[l_i = 1]$ must color $l_i$ the same color as $v1$. Similarly, the graph $G[l_i = 2]$ equals $G$ plus the edges $[l_i, v1]$ and $[l_i, v3]$; and $G[l_i = 3]$ is $G$ plus the edges $[l_i, v1]$ and $[l_i, v2]$. More generally, the graph $G[l_1 = c_1, l_2 = c_2, \ldots, l_k = c_k]$ is equal to the graph $G'[l_k = c_k]$, where $G' = G[l_1 = c_1, \ldots, l_{k-1} = c_{k-1}]$. In what follows, setting $l_i = 2$ codes setting $l_i$ to false and setting $l_i = 1$ codes setting $l_i$ to true.*

LEMMA 13. *Let $f = AC$ be a CNF formula, where $A = (l_1 \vee l_2 \vee \cdots \vee l_q)$, and let $G = \text{Graph}(f)$. Then $G[l_1 = 2, l_2 = 2, \ldots, l_q = 2]$ has a polynomial-sized $\mathcal{HC}$ construction.*

*Proof.* The lemma is obvious when $q = 1$. For $q \geq 2$, recall that the graph $G = \text{Graph}(f)$ contains a subgraph, $G_C$, for each clause $C$ in $G$. In particular, the underlying subgraph corresponding to clause $A$, $G_A$ contains a "rail" of triangle graphs, with underlying vertices $\{l_1, \ldots, l_q\} \cup \{v1, v2, v3\} \cup \{a_i, b_i, c_i \mid 1 \leq i \leq q-1\}$. (See Fig. 1.)

Let $G' = G[l_1 = 2, l_2 = 2, \ldots, l_q = 2]$. First, we will generate the set of graphs $\{G'[c_1 = 2, c_2 = 2, \ldots, c_{k-1} = 2, c_k = 1], G'[c_1 = 2, c_2 = 2, \ldots, c_{k-1} = 2, c_k = 3] \mid 1 \leq k \leq q-1\}$, and also the graph $G'[c_1 = 2, c_2 = 2, \ldots, c_{q-1} = 2]$. Because each of these graphs is non-3-colorable, we have $(\emptyset \Rightarrow H)$, for all graphs $H$ described above. Further, the implication is $V'$*-local*, where $|V'| \leq 8$. Therefore, by Theorem 3, there exist polynomial-sized constructions for each of these graphs.

Second, we will construct the desired graph, $G'$, from the above graphs. We will show by downward induction on $k$ $(1 \leq k \leq q-1)$ that we can construct the graph $G'[c_1 = 2, c_2 = 2, \ldots, c_k = 2]$ from the previous graphs. The base case holds because the graph $G'[c_1 = 2, \ldots, c_{q-1} = 2]$ is in our initial set of graphs. Now fix $k < q-1$; then we want to construct the graph $H = G'[c_1 = 2, \ldots, c_k = 2]$. By the inductive hypothesis, assume that we have constructed the graph $H[c_{k+1} = 2]$. Additionally, we have the graphs $H[c_{k+1} = 1]$ and $H[c_{k+1} = 3]$ in our initial set of graphs. Applying the edge elimination rule three times, we can construct $H$ from $H[c_{k+1} = 1]$, $H[c_{k+1} = 2]$, and $H[c_{k+1} = 3]$. □

LEMMA 14. *Let $f = AC$, $f' = A'C$, $A = (l_1 \vee \cdots \vee l_q)$, and $A' = (\pi(l_1) \vee \cdots \vee \pi(l_q))$, where $\pi$ is a permutation of $\{l_1, \ldots, l_q\}$. Then for each $k$, $1 \leq k \leq q$, there exists a polynomial-sized construction of $G'[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$ from $G$, where $G = \text{Graph}(f)$, and $G' = \text{Graph}(f')$.*

*Proof.* Again, the lemma is obvious when $q = 1$. For $q \geq 2$, we first construct the graph $G[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$ from $G$ by adding edges. Because there exists a proper 3-coloring of $G_A$ consistent with $c(l_1) = c(v2), c(l_2) = c(v2), \ldots, c(l_k) = c(v1)$, and with $b_i$ colored $c(v3)$ for $i \geq k$, we can apply the contraction rule to rename the vertices labeled $a_i, b_i, c_i$, $1 \leq i \leq q-1$, to either $v1$, $v2$, or $v3$ to obtain the graph $G''[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$, where $G'' = G \setminus e(G_A)$. (By contracting $b_i$ to $v3$ for all $i \geq k$, we are assured that we introduce no new edges from $l_i$ for $i > k$ to $v1$ or $v2$.) Now by adding edges and vertices to $G''[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$, we can construct the desired graph $G'[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$. □

LEMMA 15. *Let $f = AC$, where $A = (l_1 \vee l_2 \vee \cdots \vee l_q)$, and let $G = \text{Graph}(f)$. Then there exists a polynomial-sized $\mathcal{HC}$ construction of $G$ from the set of graphs $\mathcal{G} = \{G[l_1 = 1], G[l_1 = 2, l_2 = 1], \ldots, G[l_1 = 2, \ldots, l_{q-1} = 2, l_q = 1], G[l_1 = 2, \ldots, l_q = 2]\}$.*

*Proof.* We will prove by downward induction on $k$, $0 \leq k \leq q$, that we can construct the graph $G[l_1 = 2, \ldots, l_k = 2]$. The base case $(k = q)$ holds because $G[l_1 = 2, \ldots, l_q = 2]$ is an initial graph. Now fix $k < q$. By the inductive hypothesis, assume that we have constructed the graph $G[l_1 = 2, \ldots, l_k = 2, l_{k+1} = 2]$; also, $G[l_1 = 2, \ldots, l_k = 2, l_{k+1} = 1]$ is an initial graph. Therefore, by the edge elimination rule, the graph $G[l_1 = 2, \ldots, l_k = 2]$ can be constructed from the graphs $G[l_1 = 2, \ldots, l_k = 2, l_{k+1} = 2]$ and $G[l_1 = 2, \ldots, l_k = 2, l_{k+1} = 1]$. The number of intermediate graphs generated is $O(q)$; therefore the construction of $G$ is polynomial sized. □

The following four claims complete the proof of Lemma 12.

CLAIM 16 (commutativity). *Let $f = AC$, $A = (l_1 \vee l_2 \vee \cdots \vee l_q)$, and let $f' = A'C$, $A' = (\pi(l_1) \vee \pi(l_2) \vee \cdots \vee \pi(l_q))$, where $\pi$ is a permutation of $\{l_1, \ldots, l_q\}$. Then there exists a polynomial-sized $\mathcal{HC}$ construction of $G' =$ Graph$(f')$ from $G =$ Graph$(f)$.*

*Proof.* By Lemma 13, there exists a polynomial-sized construction of $G'[l_1 = 2, l_2 = 2, \ldots, l_q = 2]$. By Lemma 14, there exists a polynomial-sized construction of $G'[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$ for all $1 \leq k \leq q$. Therefore, by Lemma 15, there exists a polynomial-sized construction of $G'$. □

CLAIM 17 (rule (R2′)). *Let $f = (A \vee B)C$, and let $f' = AC$. Then there exists a polynomial-sized $\mathcal{HC}$ construction of $G' =$ Graph$(f')$ from $G =$ Graph$(f)$.*

*Proof.* Let $A = (l_1 \vee l_2 \vee \cdots \vee l_q)$. Then by Lemma 13, there exists a polynomial-sized $\mathcal{HC}$ construction of $G'[l_1 = 2, l_2 = 2, \ldots, l_q = 2]$. By the same argument as in the proof of Lemma 14, there exists a polynomial-sized $\mathcal{HC}$ construction of $G''[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$ for all $1 \leq k \leq q$, where $G'' = G \setminus e(G_{A \vee B})$ from $G$. Now by adding edges and vertices to $G''[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$, we can construct the graphs $G'[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$, for all $1 \leq k \leq q$. Therefore, by Lemma 15, there exists a polynomial-sized $\mathcal{HC}$ construction of $G'$. □

CLAIM 18 (rule (R3′)). *Let $f_1 = AC$, $f_2 = BC$, $f' = (A \vee B)C$. Then there exists a polynomial-sized $\mathcal{HC}$ construction of $G' =$ Graph$(f')$ from $G_1 =$ Graph$(f_1)$ and $G_2 =$ Graph$(f_2)$.*

*Proof.* Let $A = (l_1 \vee l_2 \vee \cdots \vee l_m)$, and let $B = (l_{m+1} \vee \cdots \vee l_q)$. By Lemma 13, there exists a polynomial-sized $\mathcal{HC}$ construction of $G'[l_1 = 2, \ldots, l_q = 2]$. By the argument in the proof of Lemma 14, for all $k$, $1 \leq k \leq m$, there exists a polynomial-sized $\mathcal{HC}$ construction of $G'[l_1 = 2, \ldots, l_{k-q} = 2, l_k = 1]$ from $G_1$. Also, from $G_2$, we can easily construct $G_2[l_1 = 2, \ldots, l_m = 2]$ by adding edges; and then again by the proof of Lemma 14, for all $k$, $m + 1 \leq k \leq q$, there exists a polynomial-sized $\mathcal{HC}$ construction of $G'[l_1 = 2, \ldots, l_{k-1} = 2, l_k = 1]$ from $G_2[l_1 = 2, \ldots, l_m = 2]$. Thus, by Lemma 15, there exists a polynomial-sized $\mathcal{HC}$ construction of $G'$. □

CLAIM 19 (rule (R4′)). *Let $f_1 = A\bar{l}$, $f_2 = Bl$, $f' = AB$. Then there exists a polynomial-sized $\mathcal{HC}$ construction of $G' =$ Graph$(f')$ from $G_1 =$ Graph$(f_1)$ and $G_2 =$ Graph$(f_2)$.*

*Proof.* Given $G_1 =$ Graph$(A\bar{l})$, we can construct the graph $G_1' =$ Graph$(AB\bar{l})$ by adding vertices and edges. Similarly, from $G_2 =$ Graph$(Bl)$, we can construct the graph $G_2' =$ Graph$(ABl)$ by adding vertices and edges. Because $\{G_1', G_2'\} \Rightarrow G'$ and the implication is $V'$-local where $|V'|$ is constant, by Theorem 3, there exists a polynomial-sized construction of $G'$ from $G_1'$ and $G_2'$. □

**5. The main theorem.** In this section, we will prove that $\mathcal{EF}_2$ is polynomially bounded if and only if $\mathcal{HC}$ is polynomially bounded. We will first state and prove that if $\mathcal{HC}$ is polynomially bounded, then so is $\mathcal{EF}_2$ (Theorem 20). In order to prove this theorem, we begin with an unsatisfiable CNF formula $f$ and, using the fact that $\mathcal{HC}$ is polynomially bounded, obtain a short refutation of Form(Graph$(f)$). Since this doubly translated formula is not the same as the original formula $f$, we still need to show (Lemma 21) how to derive $f$ efficiently from Form(Graph$(f)$). Similarly, to prove the other direction of the equivalence, we need to show how to derive $G$

efficiently from Graph(Form($G$)); this is accomplished by Lemma 26.

THEOREM 20. *If $\mathcal{HC}$ is polynomially bounded, then so is $\mathcal{EF}_2$.*

*Proof.* Let $f$ be an unsatisfiable CNF formula, and assume that $\mathcal{HC}$ is polynomially-bounded. Then Graph($f$) is a non-3-colorable graph, and therefore, Graph($f$) has a polynomial-sized $\mathcal{HC}$ construction. Thus, there is also a short refutation of Form (Graph($f$)) in Form($\mathcal{HC}$). By Lemma 11, this implies that there are polynomial-sized $\mathcal{EF}_2$ refutations of Form(Graph($f$)). Because $\mathcal{EF}_2$ $p$-simulates $\mathcal{F}$ (by Lemma 5), it follows by the lemma below that we can derive a polynomial-sized refutation of $f$ in $\mathcal{EF}_2$. □

LEMMA 21. *Let $f$ be a CNF formula. Then from* Form(Graph($f$)), *we can derive a depth-*5 *refutation of $f$ in $\mathcal{F}$, of size polynomial in the length of $f$.*

*Proof.* First we will obtain a polynomial-sized extended Frege proof of $f$; then we will see how to remove the extensions. Let $f = C_1 \wedge C_2 \wedge \cdots \wedge C_q$ be a CNF formula, with underlying variables $\{x_1, \dots, x_n\}$. The proof follows the basic idea behind the [GJS] proof that if Graph($f$) is non-3-colorable, then $f$ is unsatisfiable. We will define a particular coloring for Graph($f$), based on the propositional variables $\{x_1, \dots, x_n\}$.

Let $C_i = (l_1 \vee l_2 \vee \cdots \vee l_q)$ be a clause in $f$, where each $l_k$ is a literal over $\{x_1, \dots, x_n\}$. We define the following extensions for all $1 \le j \le q-1$ and $1 \le k \le q$:

$$\begin{aligned}
&R_{l_k} \leftrightarrow l_k;\ B_{l_k} \leftrightarrow \overline{l_k};\ G_{l_k} \leftrightarrow 0;\\
&R_{\overline{l_k}} \leftrightarrow \overline{l_k};\ B_{\overline{l_k}} \leftrightarrow l_k;\ G_{\overline{l_k}} \leftrightarrow 0;\\
&R_{a^i_j} \leftrightarrow (\overline{l}_1 \wedge \cdots \wedge \overline{l_{j+1}});\ B_{a^i_j} \leftrightarrow (l_1 \vee \cdots \vee l_j) \wedge \overline{l_{j+1}};\\
&\quad G_{a^i_j} \leftrightarrow l_{j+1};\\
&R_{b^i_j} \leftrightarrow 0;\ B_{b^i_j} \leftrightarrow l_{j+1};\ G_{b^i_j} \leftrightarrow \overline{l_{j+1}};\\
&R_{c^i_j} \leftrightarrow (l_1 \vee \cdots \vee l_{j+1});\ B_{c^i_j} \leftrightarrow (\overline{l_1} \wedge \cdots \wedge \overline{l_{j+1}});\\
&\quad G_{c^i_j} \leftrightarrow 0;\\
&R_{v1} \leftrightarrow 1;\ B_{v1} \leftrightarrow 0;\ G_{v1} \leftrightarrow 0;\\
&R_{v2} \leftrightarrow 0;\ B_{v2} \leftrightarrow 1;\ G_{v2} \leftrightarrow 0;\\
&R_{v3} \leftrightarrow 0;\ B_{v3} \leftrightarrow 0;\ G_{v3} \leftrightarrow 1.
\end{aligned}$$

Now consider each clause in Form(Graph($f$)) after applying the above extensions. It can be checked that all clauses with the exception of clauses of the form $(\overline{B_{c^i_{q-1}}} \vee \overline{B}_{v2})$ can be simplified to "true." For each $i$, $1 \le i \le q$, the only clause remaining is $(\overline{B_{c^i_{q-1}}} \vee \overline{B}_{v2})$, which becomes $\overline{B_{c^i_{q-1}}} = C_i$ after applying the substitution. Combining all of the $C_i$'s, we can then derive $f$. Furthermore, it is not too hard to see that this derivation has depth 5 and is polynomial sized.

The following lemma shows that we can remove all extensions in the above proof, without increasing the size or depth by more than a constant factor. The main idea behind the lemma is that any logarithmic-depth, polynomial-sized circuit can be converted into a polynomial-sized formula.

DEFINITION. *Let $P$ be an extended Frege proof. Let the variables introduced by extension axioms be $A_1, \dots, A_m$, and assume that they are introduced in $P$ in this order. Each extension variable has an associated recursion depth, defined as follows. The recursion depth of $A_1$ is depth$(A_1)$. Suppose that $A_k$ is defined based on extension variables $A_1, \dots, A_{k-1}$. Let* max *be the maximum recursion depth of $A_1, \dots, A_{k-1}$. Then the recursion depth of $A_k$ is* max+depth$(A_k)$.

LEMMA 22. *Let $P$ be a depth-$d$ extended Frege proof of $f$ such that each extension variable has recursion depth that is no greater than $\lceil \log(|P|) \rceil$. Then there exists a Frege proof $P'$ of $f$ such that $|P'| \leq p(|P|)$, where $p$ is a fixed polynomial, independent of $f$, $P$, and $P'$.*

Because the recursion depth of the extension sequence defined above is 3, the above lemma implies that from Form(Graph($f$)) there exists a constant-depth, polynomial-sized refutation in $\mathcal{F}$ of $f$. This completes the proof of Lemma 21. □

COROLLARY 23. *If $\mathcal{HC}^-$ is polynomially bounded, then so is $\mathcal{F}_5$.*

*Proof.* The proof of this corollary is almost identical to the proof of Theorem 20, with $\mathcal{EF}_2$ replaced by $\mathcal{F}_5$, $\mathcal{HC}$ replaced by $\mathcal{HC}^-$, and Lemma 11 replaced by Lemma 10. □

Recently, it has been shown in [PBI], [KPW] that any depth-$d$ proof in $\mathcal{F}$ of the propositional pigeonhole principle, $PHP_n$, must have size at least $2^{\Omega(n^{6^{-d}})}$. This lower bound together with the above results give us the following corollary.

COROLLARY 24. *There exists a family of non-3-colorable graphs $\{G_n \mid n \in N\}$ such that for $n$ sufficiently large, any $\mathcal{HC}^-$ construction of $G_n$ has size at least $2^{n^\epsilon}$, for some $\epsilon$, $0 < \epsilon < 1$ .*

The converse to Theorem 20 is much more surprising, because $\mathcal{HC}$ is quite simple compared to the formulation of an extended Frege proof system.

THEOREM 25. *If $\mathcal{EF}_2$ is polynomially bounded, then so is $\mathcal{HC}$ .*

*Proof.* Let $G$ be a non-3-colorable graph, and assume that $\mathcal{EF}_2$ is polynomially bounded. Then Form($G$) is an unsatisfiable formula, and therefore Form($G$) has a polynomial-sized $\mathcal{EF}_2$ refutation. Thus there is also a polynomial-sized construction of Graph(Form($G$)) in Graph($\mathcal{EF}_2$). By Lemma 12, this implies that there are polynomial-sized constructions of Graph(Form($G$)) in $\mathcal{HC}$. Now by the lemma below, there is a polynomial-sized $\mathcal{HC}$ construction of $G$ from Graph(Form($G$)). □

LEMMA 26. *If $G$ is not 3-colorable, then there exists a polynomial-sized $\mathcal{HC}$ construction of $G$ from the graph* Graph(Form($G$)).

*Proof.* Let Graph(Form($G$)) be denoted by $H = (V_H, E_H)$. For a particular vertex $x$ in $G$, let the corresponding set of nodes in $H$, $\{R_x, B_x, G_x, \overline{R}_x, \overline{B}_x, \overline{G}_x\}$, be called $N(x)$. For each vertex $x$ in $G$, there is a set of five clauses in Form($G$) that express that $x$ is assigned to exactly one color. (Namely, they say that exactly one of the variables $R_x$, $B_x$, $G_x$ is true.) Let $S_1(x)$ denote the induced subgraph of Graph(Form($G$)) corresponding to these five clauses associated with $x$. For any edge $[x, y]$ in $G$, there is a set of three clauses in Form($G$) that expresses that $x$ and $y$ must be assigned different colors. Let $N(x, y)$ denote the induced subgraph of Graph(Form($G$)) corresponding to these three clauses associated with the edge $[x, y]$. (The vertices in the subgraph $N(x, y)$ are the vertices in $N(x)$ and $N(y)$ as well as the triangle graph with underlying vertices $v_1$, $v_2$, and $v_3$.) Note that $N(x, y)$ is the same as $N(y, x)$ because our graphs are undirected. Figure 2 identifies $N(x, y)$ for edge $[x, y]$. Special edges that will be referred to later are labeled. For brevity, a vertex labeled by "1" indicates that this vertex is connected to vertex $v1$, and similarly for labels "2" and "3."

The graph $G = (V_G, E_G)$ will be constructed from $H$ in four steps.

(1) Starting with the graph Graph(Form($G$)), we obtain an enlarged graph, $H_1$, by adding additional vertices and edges.
(2) Using $H_1$, we construct the subgraph of $H_1$ where for each edge $[x, y]$ in $G$, the edges $e^R_{xy}$, $e^B_{xy}$, $e^G_{xy}$ are removed. Call this graph $H_2$.
(3) Using $H_2$, we construct the graph $G \cup T$, where $T$ is the triangle graph, $G$ is

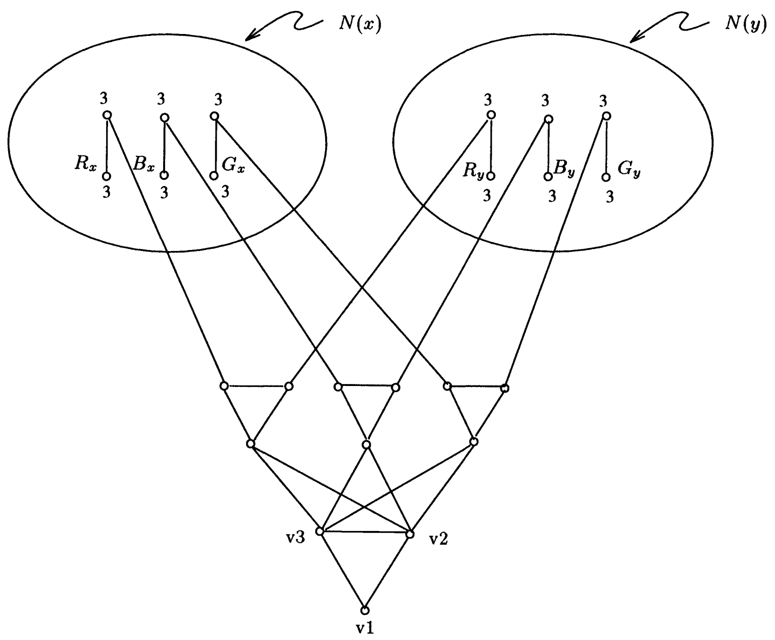


FIG. 2. *The subgraph* $N(x,y)$.

the graph that we would like to construct, and there are no edges between $G$ and $T$.

(4) Using $G \cup T$, we construct the graph $G$.

**Step (1).** We construct the graph $H_1$ by adding the following subgraphs to $H$.

(a) Add the disjoint graph $G$.

(b) For each vertex $x \in G$, we add a subgraph called $S'(x)$. $S'(x)$ connects $x$ to $N(x)$ such that if $c$ is a proper 3-coloring of $H_1$ such that $c(R_x) = c(v1)$, $c(B_x) = c(v2)$, $c(G_x) = c(v2)$, then $c(x) = c(v1)$; if $c(R_x) = c(v2)$, $c(B_x) = c(v1)$, $c(G_x) = c(v2)$, then $c(x) = c(v2)$; and if $c(R_x) = c(v2)$, $c(B_x) = c(v2)$, $c(G_x) = c(v1)$, then $c(x) = c(v3)$. For node $x$, $S'(x)$ is shown in Fig. 3. Let $S(x)$ denote the induced subgraph containing all edges and vertices of $S_1(x)$ plus all edges and vertices of $S'(x)$.

(c) For each edge $[x, y]$ in $G$, we add a few extra vertices and edges to $N(x, y)$; let $N'(x, y)$ denote the subgraph with vertex set $N(x, y)$ plus these new vertices. For edge $[x, y]$, the extra edges and vertices are shown in Fig. 4. (The new edges are indicted by dotted lines.) For each $[x, y]$, the subgraph $N'(x, y)$ has the property that if $c$ is a 3-coloring of the vertices of $N'(x, y)$ such that the only monochromatic edge is $e_{xy}^R$, then both $R_x$ and $R_y$ have the color $c(v1)$.

**Step (2).** We will show how to remove the edges $e_{xy}^R$, $e_{xy}^B$, and $e_{xy}^G$, for all $[x, y] \in E$, one at a time. Let us begin with the graph $H_1 = (V, E)$ constructed above; we will show how to construct efficiently the graph $H_1 \setminus e_{xy}^R$, where $[x, y] \in G$. Let $W(x, y)$ be the subgraph of $H_1$ consisting of $\{S(x), S(y), N'(x, y), x, y\}$. Note that the size of $W(x, y)$ is constant, independent of $|V|$.

CLAIM 27. *If $c$ is a 3-coloring to the vertices of $W(x, y)$ such that edge $e_{xy}^R$ is monochromatic, then so is some other edge in $W(x, y)$*

*Proof.* If $e_{xy}^R$ is monochromatic, then either some edge in $N'(x, y)$ is also monochro-

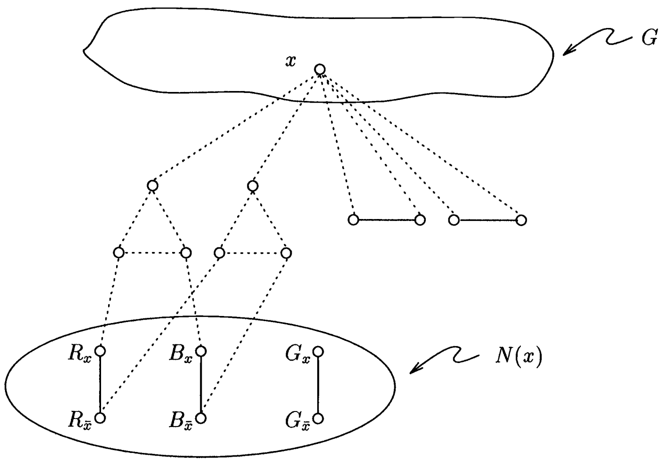


FIG. 3. *The subgraph $S'(x)$.*

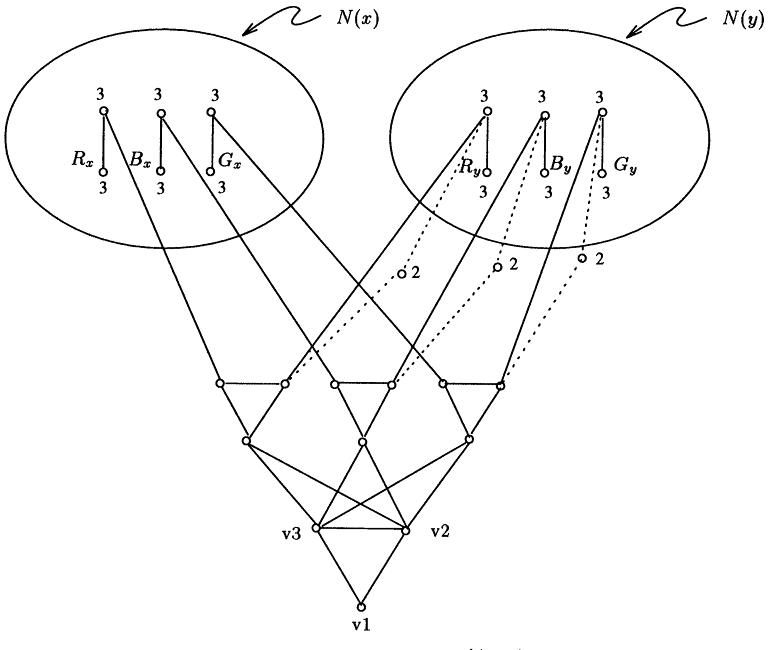


FIG. 4. *The subgraph $N'(x, y)$.*

matic, or vertices $R_x$ and $R_y$ are both colored $c(v1)$. When both vertices $R_x$ and $R_y$ are both colored $c(v2)$ and $N'(x, y)$ is properly colored, then either some edge in $S(x)$ or $S(y)$ is monochromatic or edge $[x, y] \in G$ is monochromatic. Thus, when edge $e^R_{xy}$ is monochromatic, some edge in $W(x, y)$ is also monochromatic. $\square$

The only difference between $H_1$ and the desired graph, $H_1 \setminus e^R_{xy}$, is the edge $e^R_{xy}$. Therefore, the above claim shows that $H_1 \Rightarrow H_1 \setminus e^R_{xy}$. Therefore, by Theorem 2, there exists a construction of the desired graph, $H_1 \setminus e^R_{xy}$, from $H_1$. Furthermore, because the implication is $V'$-local, where $|V'|$ is constant, by Theorem 3, the construction has polynomial size. Applying this argument repeatedly to remove all edges $e^R_{xy}$, $e^B_{xy}$, and $e^G_{xy}$, we will eventually obtain the desired graph $H_2$. After removal of $e^R_{xy}$, $e^B_{xy}$, $e^G_{xy}$ from $H_1$, $N'(x,y)$ is replaced by two subgraphs, $N'_x(x,y)$, which is the empty subgraph, and $N'_y(x,y)$, which consists of $N'(x,y)$ minus the edges $e^R_{xy}$, $e^B_{xy}$, and $e^G_{xy}$.

**Step (3).** We will now construct the graph $G \cup T$ from $H_2$. For each vertex $x \in G$, let $Q(x)$ denote the induced subgraph of $G$ containing $S(x)$ and $\{N'_x(x,y) \mid [x,y] \in G\}$. Note that for every pair $x, y \in G$, the intersection of $Q(x)$ and $Q(y)$ is the common triangle on vertices $v1$, $v2$, and $v3$. Also note that for every $x \in G$, and for every possible coloring to $x \in G$, there exists a consistent 3-coloring to the subgraph $Q(x)$.

We will show how to eliminate the edges in $Q(x)$ from $H_2$, for each $x \in G$, until we eventually obtain the desired graph $G \cup T$. Let $G' = G \cup T \cup Q(x_1) \cup Q(x_2) \cup \cdots \cup Q(x_k)$. We will show how to construct the graph $G'' = G \cup T \cup Q(x_1) \cup \cdots \cup Q(x_{k-1})$ from $G'$. First, we construct the graph $G''[x_k = 1]$. When $x_k$ is colored the same color as $v1$, there exists a proper 3-coloring of $Q(x_k)$. Therefore, there exists a way to rename each of the vertices in $Q(x_k) \setminus \{v1, v2, v3, x\}$ to either $v1$, $v2$, or $v3$. Applying this renaming to the graph $G'$, we obtain the desired graph $G''[x_k = 1]$. Similarly, we can construct the graphs $G''[x_k = 2]$ and $G''[x_k = 3]$ from $G'$. By applying the edge elimination rule, the graph $G''$ can then be constructed from the three graphs $G''[x_k = 1]$, $G''[x_k = 2]$, and $G''[x_k = 3]$.

In the same manner, we can remove all $Q(x)$ and eventually obtain the graph $G \cup T$, as desired.

**Step (4).** The graph $G$ is not 3-colorable, thus $G$ contains an odd cycle. This fact, combined with the odd-cycle lemma below prove that from $G \cup T$, we can construct $G$ in size linear in $|G|$. □

LEMMA 28 (odd-cycle lemma). *Let $G$ be a graph that contains an odd cycle, and let $T$ be a disjoint triangle graph. Then there exists a linear-sized construction of $G$ from $G \cup T$.*

*Proof.* Label the vertices of the smallest odd cycle in $G$ by $0, 1, 2, \ldots, n$, where $n$ is even. First, we embed $T$ into vertices $0, 1, 2$ of $G$ (by the contraction rule) to obtain the graph $G$ plus the edge $[0, 2]$ - call this graph $G_1$. Second, we embed $T$ into vertices $2, 3, 4$ of $G$ to obtain the graph $G$ plus the edge $[2, 4]$. We can then apply edge elimination to these two graphs to obtain a new graph with the edge $[0, 4]$—call this graph $G_2$. Third, we embed $T$ into vertices $4, 5, 6$ of $G$ to obtain the graph $G$ plus the edge $[4, 6]$. Again, we can apply edge elimination to this new graph plus graph $G_2$ to obtain $G_3$, which consists of $G$ plus the extra edge $[0, 6]$. We continue to generate $G_i$, $i \leq n/2$, until eventually we obtain the graph $G_{n/2}$, which consists of $G$ plus the edge $[0, n-2]$. We then embed $T$ in $G$ once again, this time to the vertices $0, n, n-1$ to obtain $G$ plus the extra edge $[0, n-1]$. We can then apply the edge elimination rule one last time to obtain the graph $G$. □

## 6. Graph-theoretical consequences.

In this section, we explain a connection between our work relating the complexity of the Hajós calculus to the complexity of extended Frege systems and results of Stephen Cook connecting such systems with feasibly constructive arithmetic. This connection supports the belief that proving superpolynomial lower bounds for $\mathcal{HC}$ will be very difficult. More importantly, it should guide the search for hard examples for $\mathcal{HC}$.

Intuitively, a proof of a mathematical theorem is feasibly constructive if it involves only feasible predicates and if the logical reasoning in the proof involves only feasibly computable functions. (Here we are using the common identification of "feasible" with "polynomial-time computable.") In a seminal paper [C], Cook introduced a free-variable system of arithmetic, $PV$, that contains a primitive symbol for every polynomial-time function and predicate, together with a rule of induction on notation. Boolean connectives are included, but not quantifiers. Cook proposed to make the notion of "feasibly constructive proof" precise (at least in the case of free-variable formulas) by identifying it with provability in $PV$.

Cook also demonstrated a close connection between $PV$ and $\mathcal{EF}_2$. If $F$ is a formula of $PV$, then there is a family $F_n$ of formulas in DNF, in which the $n$th formula $F_n$ expresses the assertion: "The formula $F$ is true for all numerical inputs of length $n$ or less." The formula $F_n$ is polynomial-time computable from $n$ and $F$; secondly, $F$ is valid if and only if all of the formulas $F_n$ are tautologies. If in addition $F$ is provable in $PV$, then even stronger consequences follow. The following basic simulation result was proved by Cook [C].

THEOREM 29. *If $F$ is provable in PV, then every $F_n$ has a proof in $\mathcal{EF}_2$ of polynomial size.*

In later work, Buss considered more general systems of feasibly constructive mathematics [Bu2]. In particular, he investigated a system of bounded arithmetic, $S^1_2$, based on classical predicate logic, and containing an induction scheme restricted to NP predicates. If a primitive function symbol is added to $S^1_2$ for each polynomial-time function and predicate, together with appropriate defining equations for each such function and predicate, then the resulting system is a conservative extension of Cook's $PV$ [Bu2]. This result shows that the notion of "feasibly constructive proof" is robust.

The above theorem has important consequences for proving lower bounds for $\mathcal{HC}$. Let us suppose that we have chosen a fixed method of encoding graphs as binary numbers. Then we can express the predicate "$n$ encodes a graph" as a predicate Graph$(n)$ of $PV$. Furthermore, colorings can be encoded as numbers, and the predicate "$c$ encodes a proper coloring of the graph coded by $n$" is polynomial-time computable and, hence, can be expressed as a primitive predicate Proper$(c, n)$ of $PV$. We then have the following general result relating $PV$ to $\mathcal{HC}$.

THEOREM 30. *Let $G$ be a primitive function symbol of PV for which the formulas* Graph$(G(n))$ *and* $\neg$Proper$(c, G(n))$ *are theorems of PV. Then there are polynomial-sized $\mathcal{HC}$ constructions for the family of graphs $G_n$, which are encoded by the numbers $G(n)$.*

*Proof.* By the above theorem, there is a family of tautologies $F_k$ in which the tautology $F_k$ formalizes the assertion that the above two theorems of $PV$ are true when restricted to inputs of length $k$ or less. Furthermore, this family has polynomial-sized proofs in $\mathcal{EF}_2$. By choosing $k$ large enough, we can find a tautology $F_k$ that formalizes the assertion "$G_n$ is not 3-colorable." Furthermore, this formula is polynomial-time computable from $G_n$. Using this formula we can derive the formula $\neg$Form$(G_n)$ in $\mathcal{EF}_2$ by a derivation that is polynomial in the size of $G_n$. The results of the previous section now imply that there is an $\mathcal{HC}$ construction of $G_n$ whose size is polynomial in the size of the $\mathcal{EF}_2$ derivation of $\neg$Form$(G_n)$. □

This connection gives a very easy way to check whether or not a family of graphs is a potential candidate for proving a superpolynomial lower bound for $\mathcal{HC}$. As an example, consider the family of non-3-colorable graphs constructed by Toft that are discussed in [MW] as possible candidates for graphs that show the Hajós calculus is not polynomially bounded. In fact, these graphs have short proofs by the Hajós construction, although this is not immediately obvious. However, the proof that shows all the graphs in Toft's family to be non-3-colorable is feasibly constructive, so we can infer immediately by the above result that they can be ruled out as possible hard examples. The same remark applies to other families of examples in which non-3-colorability is demonstrated by elementary constructive reasoning.

**Acknowledgments.** This work was part of the first author's Ph.D. thesis, under the supervision of Stephen Cook at the University of Toronto. The first author would like to take this opportunity to thank Steve for many enlightening conversations, as well as for his contribution to the development of this work. We would also like to thank Paul Beame for many helpful comments and suggestions. Last, we are most grateful to Vicki Shum for generating the pictures in this paper.

## REFERENCES

[Ajt1] M. AJTAI, *The complexity of the pigeonhole principle*, 29th Annual Symposium on the Foundations of Computer Science, White Plains, NY, 1988, pp. 346–355.

[Be] C. BERGE, *Graphs*, North–Holland, Amsterdam, 1985.

[BIKPPW] P. BEAME, R. IMPAGLIAZZO, J. KRAJÍČEK, T. PITASSI, P. PUDLÁK, AND W. WOODS, *Exponential lower bounds for the pigeonhole principle*, in Proc. STOC '92.

[BPU] S. BELLANTONI, T. PITASSI, AND A. URQUHART, *Approximation and small-depth Frege proofs*, SIAM J. Comput., 21 (1992), pp. 1161–1179.

[Bu2] S. BUSS, *Bounded Arithmetic*, Bibliopolis, Napoli, 1986.

[Bu4] ———, *Weak formal systems and connections to computational complexity*, Lecture notes for a topics course, University of California, Berkeley, CA, January–May 1988.

[C] S. COOK, *Feasibly constructive proofs and the propositional calculus*, in Proc. Foundations of Computer Science Conference, 1975, pp. 83–97.

[CR] S. COOK AND R. RECKHOW, *The relative efficiency of propositional proof systems*, J. Symbolic Logic, 44 (1979), pp. 36–50.

[Do] M. DOWD, *Model-theoretic aspects of $P \neq NP$*, manuscript, 1985.

[GJS] M. R. GAREY, D. S. JOHNSON, AND L. STOCKMEYER, *Some simplified $NP$-complete graph problems*, Theoret. Comput. Sci., 1 (1976), pp. 237–267.

[Haj] G. HAJÓS , *Über eine konstruktion nicht n-färbbarer graphen*, Wiss. Zeit. Martin-Luther-Univ. Halle-Wittenberg Math-Natur. Reihe, A10 (1961), pp. 116–117.

[HRT] D. HANSON, G. C. ROBINSON, AND B. TOFT, *Remarks on the graph color theorem of Hajós* , in Proc. 17th Southeastern Conference on Combinatorics, Graph Theory and Computing, Boca Raton, 1986, Congr. Numer., 55 (1986), pp. 69–76.

[KP] J. KRAJÍČEK AND P. PUDLÁK, *Propositional proof systems, the consistency of first-order theories, and the complexity of computations*, J. Symbolic Logic, 54 (1989), pp. 1063–1079.

[KPW] J. KRAJÍČEK, P. PUDLÁK, AND A. WOODS, *Exponential lower bounds to the size of bounded-depth frege proofs of the pigeonhole principle*, preprint, 1991.

[Ly] J. LYNCH, *A depth-size tradeoff for boolean circuits with unbounded fan-in*, in Structures in Complexity Theory Proc., Lecture Notes in Computer Science 223, Springer, New York, 1986, pp. 234–248.

[MW] A. J. MANSFIELD AND D. J. A. WELSH, *Some coloring problems and their complexity*, in Ann. Discrete Math. 13, North–Holland, Amsterdam, 1982, pp. 159–170.

[Ore] O. ORE, *The Four-Color Problem*, Academic Press, New York, 1967.

[PBI] T. PITASSI, P. BEAME, AND R. IMPAGLIAZZO, *Exponential lower bounds for the pigeonhole principle*, TR 257/91, University of Toronto, Toronto, Ontario, 1991.

[PU] T. PITASSI AND A. URQUHART, *The complexity of the Hajós calculus*, Proc. 33rd Symposium on the Foundations of Computing, Pittsburgh, PA, 1992.

[Tse] G. S. TSEITIN, *On the complexity of derivation in the propositional calculus*, in Studies in Constructive Mathematics and Logic, Part II, A. O. Slisenko, 1968.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 485 498, November 1995

# ON-LINE AND FIRST-FIT COLORING OF GRAPHS THAT DO NOT INDUCE $P_5$ *

HENRY A. KIERSTEAD†, STEPHEN G. PENRICE†, AND WILLIAM T. TROTTER†

**Abstract.** For a graph $H$, let Forb($H$) be the class of graphs that do not induce $H$, and let $P_5$ be the path on five vertices. In this article, we answer two questions of Gyárfás and Lehel. First, we show that there exists a function $f(\omega)$ such that for any graph $G \in$ Forb($P_5$), the on-line coloring algorithm First-Fit uses at most $f(\omega(G))$ colors on $G$, where $\omega(G)$ is the clique size of $G$. Second, we show that there exists an on-line algorithm $A$ that will color any graph $G \in$ Forb($P_5$) with a number of colors exponential in $\omega(G)$. Finally, we extend some of our results to larger classes of graphs defined in terms of a list of forbidden subgraphs.

**Key words.** on-line algorithm, graph coloring, greedy algorithm

**AMS subject classification.** 05C35

**Introduction.** An on-line graph is a structure $G^< = (V, E, <)$, where $G = (V, E)$ is a graph and $<$ is a linear ordering of $V$. We allow $V$ to be finite or countably infinite. We call $G^<$ an *on-line presentation* of the graph $G$. The on-line subgraph of $G^<$ induced by a subset $S \subset V$ is the on-line graph $G^<[S] = (S, E', <')$, where $E'$ is the set of edges in $E$ both of whose end points are in $S$, and $<'$ is $<$ restricted to $S$. We shall always assume that $V = \{x_1, x_2, \ldots\}$, where $x_i < x_j$ iff $i < j$. Let $V_i = \{x_j : j \leq i\}$ and $G_i^< = G^<[V_i]$. An algorithm for coloring the vertices of $G^<$ is said to be *on-line* if the color of a vertex $v_i$ is determined solely by $G_i^<$. Intuitively, the algorithm colors the vertices of $G^<$ one at a time in some externally determined order $x_1, \ldots, x_n$, and at the time a color is irrevocably assigned to the vertex $x_i$, the algorithm can only see $G_i^<$. A simple but important example of an on-line algorithm is the algorithm First-Fit, denoted by FF, which colors the vertices of $G$ with an initial sequence of the colors $\{1, 2, \ldots\}$ by assigning to the vertex $x_i$ the least possible color not already assigned to any vertex of $V_{i-1}$, adjacent to $x_i$.

The *clique size* and *chromatic number* of a graph $G$ are denoted by $\omega(G)$ and $\chi(G)$ respectively. Let $A$ be an on-line graph-coloring algorithm. Then $\chi_A(G^<)$ denotes the number of colors $A$ uses to color the on-line graph $G^<$ and $\chi_A(G)$ denotes the maximum of $\chi_A(G^<)$ over all on-line presentations $G^<$ of $G$. A class of graphs $\Gamma$ is said to be *$\chi$-bounded* if there exists a function $f$ such that for all $G \in \Gamma, \chi(G) \leq f(\omega(G))$. The function $f$ is called a *$\chi$-binding function* for $\Gamma$. Easy examples of $\chi$-bounded classes include the class of perfect graphs, the class of line graphs, and, more generally, the class of claw-free graphs. Similarly, for an on-line algorithm $A$, the class $\Gamma$ is $\chi_A$-bounded if there exists a function $f$ such that for all $G \in \Gamma, \chi_A(G) \leq f(\omega(G))$. In this case we say that $A$ is a *$\chi$-binding algorithm* for $\Gamma$ and $f$ is an *on-line $\chi$-binding function* for $\Gamma$. The class $\Gamma$ is *on-line $\chi$-bounded* if $\Gamma$ is $\chi_A$-bounded for some on-line algorithm $A$. In this article, we are interested in classes of graphs that are on-line $\chi$-bounded. The class of perfect graphs is not on-line $\chi$-bounded. In fact, the subclass of trees is not on-line $\chi$-bounded (see, for example, Bean [1]). However, the class of claw-free

* Received by the editors August 16, 1991; accepted for publication (in revised form) November 19, 1991. The research of the first and second authors was partially supported by Office of Naval Research grant N00014-90-J-1206. The research of the third author was partially supported by National Science Foundation grant DMS-89-02481 and by Office of Naval Research grant N00014-95-1-0701.

† Department of Mathematics, Arizona State University, Tempe, Arizona 85287.

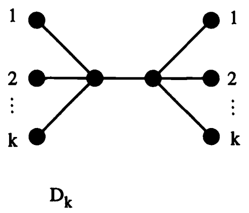


FIG. 1.

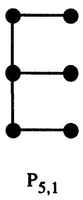


FIG. 2.

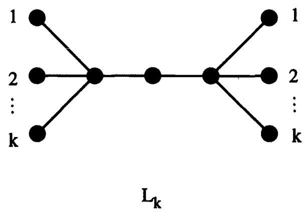


FIG. 3.

graphs is on-line $\chi$-bounded. Here we shall be interested in other classes defined by forbidding certain induced subgraphs.

For a graph $H$, let Forb$(H)$ be the class of graphs that do not induce $H$. Similarly, let Forb$(H_1, \dots, H_n)$ be the class of graphs that do not induce any of the graphs $H_1, \dots, H_n$. Gyárfás [3] and Sumner [15] independently conjectured that if $T$ is a tree, then Forb$(T)$ is $\chi$-bounded. Gyárfás [4] has shown this to be the case when $T$ is a path. Gyárfás, Szemerédi, and Tuza [8] verified the conjecture for triangle-free graphs in Forb$(T)$ when $T$ is a radius-two tree, and Kierstead and Penrice [11] extended this result by showing that Forb$(T)$ is $\chi$-bounded whenever $T$ has radius two.

Gyárfás and Lehel [7], [6] opened up an exciting and unexpected new area for study when they proved that Forb$(P_5)$ is on-line $\chi$-bounded, where $P_n$ is a path on $n$ vertices. They also showed that Forb$(P_6)$ is not on-line $\chi$-bounded. These results led to many interesting questions. The Gyárfás–Lehel algorithm was quite complicated and gave a superexponential on-line $\chi$-binding function. They asked whether Forb$(P_5)$ had an exponential on-line $\chi$-binding function and whether the simple algorithm First-Fit was an $\chi$-binding algorithm for Forb$(P_5)$. In this article we prove the following theorems.

THEOREM 1.1. *There exists an on-line algorithm $A$ and an exponential function $f(\omega) = (4^\omega - 1)/3$ such that $\chi_A(G) \leq f(\omega(G))$, for any graph $G \in$* Forb$(P_5)$.

THEOREM 2.1. Forb$(P_5)$ *is $\chi_{\mathrm{FF}}$-bounded.*

The smallest function known to be an (off-line) $\chi$-binding function for Forb$(P_5)$ is $2^n$. Our on-line $\chi$-binding function for Forb$(P_5)$ is within a power of two of this function. In light of the Gyárfás–Sumner conjecture, one is led to ask for which trees $T$, Forb$(T)$ is on-line $\chi$-bounded. Since Forb$(P_6)$ is not on-line $\chi$-bounded, neither is Forb$(T)$ if $T$ has radius greater than two. The authors [13] have recently proved that Forb$(T)$ is on-line $\chi$-bounded if $T$ has radius at most two.

Theorem 2.1, together with some observations of Gyárfás and Lehel [5], allow us to characterize those trees $T$ for which Forb$(T)$ is $\chi_{\mathrm{FF}}$-bounded.

THEOREM 2.2. *Let $T$ be a tree.* Forb$(T)$ *is $\chi_{\mathrm{FF}}$-bounded if and only if $T$ does not induce $K_2 + 2K_1$.*

For other trees $T$, we shall try to determine reasons why Forb$(T)$ is not $\chi_{\mathrm{FF}}$-bounded. In [11] Kierstead and Penrice showed that for any tree $T$ and integer $t$, Forb$(T, K_{t,t})$ is $\chi_{\mathrm{FF}}$-bounded. This fact is used in [13] to prove that Forb$(T)$ is on-line $\chi$-bounded for any radius-two tree $T$. However, the fact that a graph contains $K_{t,t}$ does not explain why it might have a large First-Fit chromatic number. Let $B_t$ be the graph obtained from $K_{t,t}$ by removing a perfect matching $M$. If $B_t^<$ is the on-line presentation of $B_t$, where adjacent vertices of $M$ appear consecutively, then it is easy to see (and is explained in more detail in the proof of Theorem 2.2) that $\chi_{\mathrm{FF}}(B_t^<)$ is $t$. Thus if a graph induces $B_t$, then we have a certificate that its First-Fit chromatic number is at least $t$. We shall prove the following two theorems which are quite satisfying from this point of view. The trees $D_k$ and $P_{5,1}$ are shown in Figs. 1

and 2 (also see the definition at the end of this section).

THEOREM 2.3. *For every positive integer $k$,* $\mathrm{Forb}(P_{5,1}, B_t)$ *is* $\chi_{\mathrm{FF}}$*-bounded.*

THEOREM 2.4. *For every positive integer $k$,* $\mathrm{Forb}(D_k, B_t)$ *is* $\chi_{\mathrm{FF}}$*-bounded.*

It should be noted that $B_3$ is just the six cycle $C_6$.

Finally, we mention some related results on First-Fit. Woodall [16] showed that the class of interval graphs is $\chi_{\mathrm{FF}}$-bounded. Gyárfás and Lehel [5] gave an improved bound for this problem and introduced the notion of a wall, which we shall also use. Recently Kierstead [10] showed that the binding function is linear. Interval graphs are cocomparability graphs of interval orders. The class of comparability graphs contains the class of trees and so is not on-line $\chi$-bounded. However, Kierstead [12] has proved that the class of comparability graphs of interval orders is $\chi_{\mathrm{FF}}$-bounded. A consequence of a theorem of Chvátal [2] is that First-Fit uses exactly $\omega(G)$ colors to color $G \in \mathrm{Forb}(P_4)$, where $T$ is the path on four vertices.

Theorem 1.1 is proved in §1. Theorems 2.1–2.4 are proved in §2. In §3, we discuss some problems for further research. In the remainder of this section we review our terminology and notation.

Let $G = (V, E)$ be a graph. Adjacency between two vertices $x$ and $y$ is denoted by $x \sim y$ and nonadjacency is denoted by $x \not\sim y$. The neighborhood of a vertex $x$ is denoted by $N(x) = \{y \in V : y \sim x\}$. The clique number, the independence number, the chromatic number, and the number of vertices of $G$ are denoted by $\omega(G), \alpha(G), \chi(G)$, and $\nu(G)$, respectively. The following special notation is used to denote certain graphs.

$P_k$ : path on $k$ vertices.

$S_k$ : star with $k$ leaves, i.e., a star on $k+1$ vertices.

$D_k$ : tree obtained by adding $k-1$ leaves to the second and third vertices of $P_4$.

$L_k$ : tree obtained by adding $k-1$ leaves to the second and fourth vertices of $P_5$ (see Fig. 3).

$P_{n,k}$ : tree obtained by adding $k$ leaves to the third vertex of $P_n$.

$P^2_{n,k}$ : broom obtained by adding $k-1$ leaves to the second vertex of $P_n$.

$LS_k$ : tree on $2k+1$ vertices consisting of $k$ independent edges and a vertex which is adjacent to exactly one vertex of each of these edges.

$K_s$ : clique on $s$ vertices.

$K_{t,t}$ : complete bipartite graph with $t$ vertices in each part.

$B_t$ : bipartite graph obtained by deleting a perfect matching from $K_{t,t}$.

For a set $S$, let $[S]^2 = \{A \subset S : |S| = 2\}$. We may write $(\alpha > \beta)$ for the two element set $\{\alpha, \beta\}$ to denote that $\alpha > \beta$. For a coloring $p\colon [S]^2 \to \{1, \dots, n\}$, we say that $H \subset S$ is *homogeneous* if $p$ restricted to $[H]^2$ is constant. Let $R_i(t)$ be the Ramsey function such that for every coloring $p\colon [S]^2 \to \{1, \dots, i\}$, with $|S| \geq R_i(t)$, there exists a homogeneous subset $H \subset S$ such that $|H| = t$. Similarly, let $R(t_1, \dots, t_i)$ be the Ramsey function such that for every coloring $p\colon [S]^2 \to \{1, \dots, i\}$, with $|S| \geq R(t_1, \dots, t_i)$, there exists a homogeneous set $H$ with $|H| = t_j$ such that $p(\alpha > \beta) = j$, for all $\alpha, \beta \in H$ with $\alpha \neq \beta$. Note that if $G$ is a graph such that $\nu(G) \geq R(\omega, \alpha)$, then either $\omega(G) \geq \omega$ or $\alpha(G) \geq \alpha$.

## 1. An exponential on-line algorithm for $P_5$-free graphs.

In this section we prove Theorem 1. Before starting the proof, we establish two useful properties of $\mathrm{Forb}(P_5)$.

LEMMA 1.2. *Suppose $G \in \mathrm{Forb}(P_5)$. If $M_1$ and $M_2$ are distinct maximal cliques in the same connected component of $G$, then there exist vertices $x \in M_1$ and $y \in M_2$ such that $x$ is adjacent to $y$.*

*Proof.* Suppose not. Then $M_1$ and $M_2$ are disjoint. Since $M_1$ and $M_2$ are in the same connected component, there exists a path from $M_1$ to $M_2$, i.e., a path $P = (x_1, \dots, x_m)$ such that $x_1 \in M_1, x_m \in M_2$. Let $P$ be a shortest such path. Then $P$ is an induced path and, by our initial assumption, $m \geq 3$. Since $M_1$ is a maximal clique, there exists $x_0 \in M_1$ not adjacent to $x_2$. Since $P$ is a shortest path, $y$ is not adjacent to $x_i$ for $i \geq 3$. Similarly, there exists $x_{m+1} \in M_2$ such that $x_{m+1}$ is not adjacent to $x_i$ for $i \leq m-1$. Thus $x_0 + P + x_{m+1}$ is an induced path of length at least five, which is a contradiction. □

COROLLARY 1.3. *If $x$ is a cut vertex of a connected graph $G$ in* Forb$(P_5)$ *and $M_1$ and $M_2$ are maximal cliques in distinct components of $G - x$, then either $M_1 \cup \{x\}$ or $M_2 \cup \{x\}$ is a clique.*

*Proof of Theorem* 1.1. We first present an on-line algorithm $A$ and then prove that it properly colors any on-line presentation $G^<$ of a graph $G \in$ Forb$(P_5)$ with at most $f(\omega(G))$ colors. An important point is that the algorithm must be independent of the clique size of $G$. For this reason, the algorithm is defined recursively. Let $A(x, G^<)$ denote the color $A$ assigns to $x$, when $x$ is considered as a vertex of the on-line graph $G^<$. When a new vertex $x_i$ is presented, the algorithm first assigns $x_i$ to one of the sets $S_{j,k}$, where $1 \leq j \leq 4$ and $k$ is the clique size of the connected component of $x_i$ in $G_i^<$. This will be done so that for $j < 4$, each $S_{j,k}$ has clique size less than $k$ and each $S_{4,k}$ is an independent set. Then $A$ assigns $x_i$ a color derived from the color $A(x_i, G^<[S_{j,k}])$ obtained by a recursive call of itself. Vertices in distinct $S_{j,k}$ receive colors from disjoint sets of colors. Note that the exponential function $f(w) = (4^w - 1)/3$ is defined recursively by $f(1) = 1$ and $f(k) = 4f(k-1) + 1$.

At each stage of the algorithm, each connected component $C$ of $G_i^<$ will have a special maximum clique $K$, called the *active clique*. Once a clique becomes active, it remains active until a larger clique is formed in its component. The choice of which $S_{j,k}$ in which to put $x_i$ is completely determined by $\omega(C)$ and the adjacencies between $x_i$ and elements of $K$, where $C$ denotes the connected component of $x_i$ in $G_i^<$ and $K$ denotes the active clique of $C$.

Suppose $A$ has colored $G_{i-1}$. We specify how $A$ assigns a color to the next vertex $x_i$.

ALGORITHM $A(x_i, G^<)$.

Find $C$, the connected component of $x_i$ in $G_i^<$, and set $k = \omega(C)$.

**Case 1:** $k > \omega(C - \{x_i\})$. Put $x_i$ in $S_{4,k}$. [Claim 1: $S_{4,k}$ is independent.] Set $A(x_i, G^<) = f(k)$. Let $K$ be a $k$-clique in $C$. Deactivate any active cliques of $C - \{x_i\}$ and designate $K$ as the active clique of $C$.

**Case 2:** $k = \omega(C - \{x_i\})$. Let $K$ be the active clique of $C$. [Claim 2: $K$ is unique.]

**Case 2a:** For some $v \in K$, both $x_i \sim v$ and $u \sim v$, for all $u \in C \cap S_{1,k}$. Put $x_i$ in $S_{1,k}$. [Claim 3: $\omega(S_{1,k}) < k$.] Set $A(x_i, G^<) = f(k-1) + A(x_i, G^<[S_{1,k}])$.

**Case 2b:** Not Case 2a and for some $v \in K$, both $x_i \sim v$ and $u \sim v$, for all $u \in C \cap S_{2,k}$. Put $x_i$ in $S_{2,k}$. [Claim 4: $\omega(S_{2,k}) < k$.] Set $A(x_i, G^<) = 2f(k-1) + A(x_i, G^<[S_{2,k}])$.

**Case 2c:** Not Case 2a or Case 2b. Put $x_i$ in $S_{3,k}$. [Claim 5: $\omega(S_{3,k}) < k$.] Set $A(x_i, G^<) = 3f(k-1) + A(x_i, G^<[S_{3,k}])$.

Next we show that the algorithm produces a proper $f(\omega(G))$-coloring of $G^<$, assuming the five claims above. Then we shall verify the claims. We argue by induction on $\omega(G)$. If $\omega(G) = 1$, then every point is isolated. Thus each $x_i$ is assigned to the independent set $S_{4,1}$ and colored $f(1) = 1$.

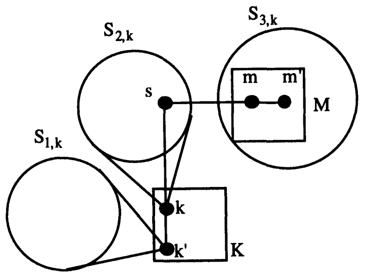


FIG. 4

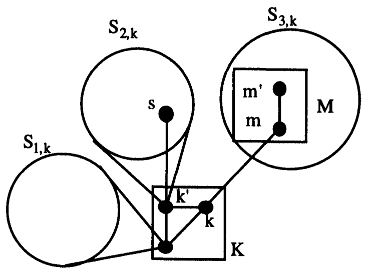


FIG. 5

Now suppose $\omega(G) = k > 1$. By Claims 3–5, for each $j \le 3$ and $m \le k, \omega(S_{j,m}) \le k - 1$. By the induction hypothesis, $S_{j,m}$ is properly colored with colors from the set $\{jf(m-1)+1, \dots, (j+1)f(m-1)\}$, since $A(x_i, G^{<}[S_{j,m}]) \le f(m-1)$ for each $x_i \in S_{j,k}$. By Claim 1, for each $m \le k$, the set $S_{4,m}$ is independent and colored with $f(m) = 4f(m-1)+1$. Since these sets of colors are easily seen to be pairwise disjoint, we are done.

Claim 3 is easy because all vertices in the same connected component of $S_{1,k}$ have a common neighbor in their active $k$-clique. Claim 4 is similar. We shall prove Claims 1 and 2 by induction on $i$. When $i = 1$, both claims are trivial. Suppose $i > 1$. If Claim 1 fails, then by induction $x_i$ is adjacent to some $x_j \in S_{4,k}$, where $j < i$. Clearly the component $C'$ of $x_j$ in $G_j^{<}$ is contained in $C$ since $x_j \in S_{4,k}, \omega(C') = k$, which contradicts $\omega(C - \{x_i\}) < k$. By the induction hypothesis, the only way that Claim 2 could fail is if $x_i$ is a cut vertex of $C$ and two distinct components of $C - \{x_i\}$ contain $k$-cliques. Then by Corollary 1.3, $x_i$ is in a $(k+1)$-clique of $C$, contradicting $\omega(C) = k$.

It remains to prove Claim 5. Suppose it is false. Let $M$ be a $k$-clique in $S_{3,k}$. Let $v_{i_1}, \dots, v_{i_k}$ be the vertices of $M$ with $i_1 < i_2 < \cdots i_k$. If $K$ is the active clique of $v_{i_1}$ in $G_{i_1}^{<}$, it is easy to show by induction that $K$ is the active clique of the component of $v_{i_j}$ in $G_{i_j}^{<}$ whenever $1 \le j \le k$, since $K$ could only be deactivated by the addition of a vertex which raised the clique size of the component. But in that case, $v_{i_k}$ would have been assigned to $S_{j,k'}$ for some $j \le 4, k' > k$. We consider two cases.

*Case* 1. There exist $s \in S_{2,k}$ and $m \in M$ such that $s \sim m$. Since $\omega(C) = k$, there exists $m' \in S_{2,k}$ such that $m' \not\sim s$. (See Fig. 4.) Since $m$ and $m'$ are not in $S_{2,k}$, there exists $k \in K$ such that $k \sim s, k \not\sim m$, and $k \not\sim m'$. Also since $m, m'$, and $s$ are not in $S_{1,k}$, there exists $k' \in K$ such that $k' \not\sim m, k' \not\sim m'$, and $k' \not\sim s$. But this is a contradiction, since $(k', k, s, m, m')$ is an induced $P_5$.

*Case* 2. For all $s \in S_{2,k}$ and $m \in M, s \not\sim m$. (See Fig. 5.) By Lemma 1.2 there exist $m \in M$ and $k \in K$ such that $m \sim k$. Also there exists $m' \in M$ such that $k \not\sim m'$.

Since $m \notin S_{2,k}$, there exists $s \in S_{2,k}$ such that $s \not\sim k$. Since neither $m$ nor $m'$ is in $S_{2,k}$, there exists $k' \in K$ such that $s \sim k', m \not\sim k'$, and $m' \not\sim k'$. By the hypothesis of this case, $s \not\sim m$ and $s \not\sim m'$. Thus $(s, k', k, m, m')$ is an induced $P_5$, which is a contradiction. □

**2. First-Fit.** We begin this section with the proof of Theorem 2.2 assuming Theorem 2.1. We then state and prove a series of lemmas which lead eventually to the proofs of Theorems 2.3 and 2.4. Along the way we pause to prove Theorem 2.1.

*Proof of Theorem* 2.2. The reader may check that if $T$ does not induce $K_2 + 2K_1$, then $T$ is either a star or a path on 5 or fewer vertices. Thus it suffices to show that $\mathrm{Forb}(K_2 + 2K_1)$ is not $\chi_{\mathrm{FF}}$-bounded, whereas $\mathrm{Forb}(S_k)$ and $\mathrm{Forb}(P_5)$ are.

Gyárfás and Lehel [5] noted that $\mathrm{Forb}(K_2 + 2K_1)$ is not $\chi_{\mathrm{FF}}$-bounded as follows. Recall that $B_t$ is the graph formed by deleting a perfect matching from a complete bipartite graph with $t$ vertices in each part. Let $\{a_1, \dots, a_t\}$ and $\{b_1, \dots, b_t\}$ be the independent sets of the bipartition of $B_t$, and assume that the pairs $\{a_i, b_i\}$ are independent. It is easy to check that, for all positive integers $t, B_t$ does not induce $K_2 + 2K_1$ and that First-Fit will use $t$ colors on $B_t$ if the vertices are presented in the order $a_1, b_1, a_2, b_2, \dots, a_t, b_t$.

Now note that $\mathrm{Forb}(S_t)$ is $\chi_{\mathrm{FF}}$-bounded. If $G \in \mathrm{Forb}(S_t)$ and $\omega(G) = k$, First-Fit will use no more than $R(k,t)$ colors on $G$. If a vertex $x$ receives color $R(k,t) + 1$, it must have $R(k,t)$ neighbors. Because $\omega(G) = k, x$ must in fact have $t$ independent neighbors, which, together with $x$, form an induced $S_t$.

$\mathrm{Forb}(P_5)$ is $\chi_{\mathrm{FF}}$-bounded by Theorem 2.1, and thus we are done. □

For the arguments to follow, it is useful to be able to analyze the performance of First-Fit in terms of static substructures rather than on-line presentations of graphs. To this end, we introduce the notion of a *wall*, which is due originally to Gyárfás and Lehel [5]. A *colored graph* is a pair $W = (G(W), f(W))$, where $G(W) = (V(W), E(W))$ is a graph and $f(W)$ is a proper coloring of $G(W)$. Let $C(W)$ be the range of $f(W)$. For $I \subset C(W)$, let $[I]_W$ denote the set of vertices in $G$ which are colored with some color in $I$. If $I = \{\alpha\}$, we may write $[\alpha]_W$ for $[\{\alpha\}]_W$. The colored graph $W$ is called a *wall* if, for all $\alpha > \beta$ in $C(W)$ and for every vertex $x \in [\alpha]_W$, there exists a vertex $y \in [\beta]_W$ such that $x$ is adjacent to $y$. The color classes of the wall, $[\alpha]_W$, are called *levels*, and we say that $[\alpha]_W$ is a *higher* level than $[\beta]_W$, if $\alpha > \beta$. The *height* $h(W)$ of a wall $W$ is $|C(W)|$, or, equivalently, the number of levels. A wall $W$ is said to *support* a vertex $x$ if $x$ is adjacent to some vertex at every level of $W$. We say that a wall $W$ is *in* a graph $G$ if $G(W)$ is an induced subgraph of $G$. We say that a colored graph $W'$ is an induced subwall of a wall $W$ if $W'$ is a wall in $G(W)$ and $f(W')$ is the restriction of $f(W)$ to $W'$. Note that if $W$ is a wall, then $[I]_W$ induces a subwall of $W$ for any $I \subset C(W)$. We call $[I]_W$ a level induced subwall. The following easy observation allows us to discuss walls rather than on-line graphs when considering First-Fit.

LEMMA 2.5. *Let $G$ be a graph. Then $\chi_{\mathrm{FF}}(G) = \max h(W)$, where the maximum is taken over all walls in $G$.*

*Proof.* Suppose $G^<$ is an on-line presentation of $G$ with $\chi_{\mathrm{FF}}(G^<) = t$. Then $(G, f)$ is a wall of height $t$, where $f$ is the coloring produced by First-Fit when applied to $G^<$. Alternatively, suppose that $W$ is a wall in $G$ with $h(W) = t$. Then $\chi_{\mathrm{FF}}(G^<) \geq t$ if $G^<$ is any on-line presentation in which the vertices of the lowest level of $W$ precede the vertices of the second-lowest level, which precede those of the third level, and so on through $W$, and all vertices of $W$ precede all vertices of $G - W$. □

Let $W$ be a wall in $G$, which supports a vertex $x$. Define a coloring $p = p_{W,x}$ on the two element subsets of $C(W)$ by $p(\alpha > \beta) = c$, where

$$\begin{aligned} c = 1 \text{ iff } &\text{(a) } \exists y \in ([\alpha]_W \cap N(x))\ \forall z \in ([\beta]_W \cap N(x))(y \not\sim z) \text{ and}\\ &\text{(b) } \exists y' \in ([\alpha]_W - N(x))\ \forall z' \in ([\beta]_W - N(x))(y' \not\sim z');\\ c = 2 \text{ iff } &\text{not (a); and}\\ c = 3 \text{ iff } &\text{both (a) and not (b).}\end{aligned}$$

If (a) holds for $y$, we call $y$ a *left witness point* for the pair $(\alpha, \beta)$. Note that since $W$ is a wall, in this case $y$ must be adjacent to some vertex $z' \in [\beta]_W - N(x)$. If (b) holds for $y'$ we call $y'$ a *right witness point* for the pair $(\alpha, \beta)$. Note that $y'$ is adjacent to some vertex $z \in [\beta]_W \cap N(x)$. $W$ is said to have the *cross property* if for some $x, p_{W,x}(\alpha > \beta) = 1$, for every pair of colors $\alpha, \beta \in C(W)$. The following observation is crucial to our arguments. If $p(\alpha > \beta) = 2$ for every pair $\alpha, \beta \in C(W)$, then $W \cap N(x)$ is a wall, and if $p(\alpha > \beta) = 3$ for every pair $\alpha, \beta \in C(W)$, then $W - N(x)$ is a wall.

The *path number* $\pi_G(W) = \pi(W)$ of a wall $W$ in $G$ is the length of the longest induced path $P = (x_1, \ldots, x_n)$ in $G$ such that $x_1 \in [\alpha]_W$, where $\alpha$ is the largest color in $C(W)$, and no vertex of $(x_2, \ldots, x_n)$ is adjacent to any vertex of $W - \{x_1\}$.

LEMMA 2.6. *There exists a function $g(h)$ such that for any graph $G = (V, E)$, if $W$ is a wall in $G$ such that $h(W) \geq g(h)$, then there exists an induced subwall $W'$ of $W$ such that $h(W') \geq h$ and*

(i) *$W'$ is a level induced subwall and has the cross property; or*

(ii) *$\omega(W') < \omega(W)$; or*

(iii) *both $\omega(W') = \omega(W)$ and $\pi(W') > \pi(W)$.*

*Proof.* First note that for any induced subwall $W'$ of $W, \omega(W') \leq \omega(W)$. Let $g = g(h) = R_3(2h) + 1$. Suppose $W$ is a wall in $G$ with $h(W) \geq g$. Let $P = (x = x_1, \ldots, x_\pi)$ be a path that witnesses the value of $\pi(W)$. Let $I$ be the set of the $g - 1$ smallest colors of $C(W)$ and $W_0 = [I]_W$. Let $p = p_{W_0,x}$ be the coloring of 2-subsets of $I$ defined above. By Ramsey's theorem there exists a homogeneous $2h$-subset $H \subset I$. Let $p(\alpha > \beta) = c$, for any $\alpha, \beta \in H$ with $\alpha \neq \beta$.

*Case* 1. $c = 1$. Then $H' = [H]_W$ is a level induced subwall of $W$ with the cross property and $h(W') \geq h$.

*Case* 2. $c = 2$. Then $W' = [H]_{W \cap N(x)}$ is a wall with $h(W') \geq h$. Since $V(W') \subset N(x)$ and $x \in V(W), \omega(W') < \omega(W)$.

*Case* 3. $c = 3$. Then $[H]_{W-N(x)}$ is a wall of height at least $2h - 1$. Let $\alpha > \beta$ be the two largest colors in $H$ and let $y$ be a left witness point for the pair $(\alpha, \beta)$. Let $J = \{\gamma \in H : y \sim z'$, for some $z' \in [\gamma]_W - N(x)\}$. If $|J| \geq h - 1$, let $W' = \{y\} \cup [J]_{W-N(x)}$. Then $y + P$ witnesses that $\pi(W') > \pi(W)$. See Fig. 6. Otherwise let $W' = [(H - J) \cup \{\beta\}]_{W \cap (V - N(x))}$. Then $z' + y + P$, where $y \sim z'$ and $z' \in [\beta] - N(x)$, witnesses that $\pi(W') > \pi(W)$. In either case, $h(W') \geq h$ and $\omega(W') \leq \omega(W)$. See Fig. 7. □

We now use Lemma 2.6 to give an inductive proof of Theorem 2.1.

*Proof of Theorem* 2.1. Consider a graph $G \in \text{Forb}(P_5)$. First note (1) if $W$ is a wall in $G$ with the cross property, then $h(W) = 1$ : Otherwise, let $y$ and $y'$ be left and right witness points for the pair $(\alpha, \beta)$, where $\{\alpha, \beta\} \subset C(W)$. Thus by our remark above, there exist $z' \in [\beta]_W - N(x)$ and $z \in [\beta]_W \cap N(x)$ such that $y \sim z'$ and $y' \sim z$. Since $y$ and $y'$ are witness points, $y \not\sim z$ and $y' \not\sim z'$. Thus $\{z', y, x, z, y'\}$ induces $P_5$, which is a contradiction. See Fig. 8.

Next note (2) if $W$ is a wall in $G$ with $\pi(W) \geq 3$, then $W$ contains an induced subwall $W_0$ such that $h(W_0) = h(W) - 1$ and $\omega(W_0) < \omega(W)$. Suppose $P = (x_1, x_2, x_3)$

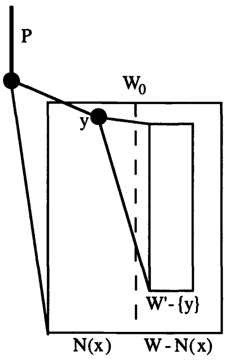


FIG. 6.

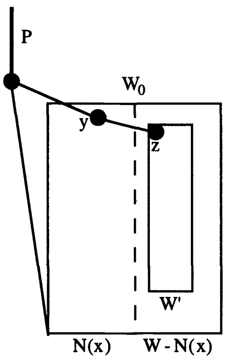


FIG. 7.

is a path that witnesses that $\pi(W) \geq 3$. If $W_0 = W \cap N(x)$ is an induced wall, we are clearly done; otherwise there exist $\alpha > \beta \in C(W_0)$ and $y_1 \in [\alpha]_{W_0}$ such that $y_1$ is not supported in $[\beta]_{W_0}$. Thus $y_1$ is supported by some $y_2 \in [\beta]_W - N(x)$. But then $x_1 \sim y_1, y_1 \sim y_2, x_1 \not\sim y_2$, and $\{x_3, x_2, x_1, y_2, y_3\}$ induces $P_5$, which is a contradiction.

Let $g$ be the function defined in Lemma 2.6. We claim that the function $f$, defined recursively by $f(1) = 1$ and $f(\omega + 1) = g \circ g(1 + f(\omega))$, is a $\chi_{\text{FF}}$-binding function for Forb$(P_5)$. We show by induction on $\omega$ that if $G \in$ Forb$(P_5)$ and $\omega(G) \leq \omega$, then $\chi_{\text{FF}}(G) \leq f(\omega)$. The base step is trivial. For the inductive step, suppose $\omega(G) \leq \omega$ and $\chi_{\text{FF}}(G) > f(\omega)$. By Lemma 2.5, $G$ contains a wall $W$ of height $\chi_{\text{FF}}(G)$. Thus by Lemma 2.6, $G$ has a wall $W_1$ of height $g(1+f(\omega))$ such that either (i) $W_1$ has the cross property, (ii) $\omega(W_1) < \omega$, or (iii) $\pi(W_1) = \omega$ and $\pi(W_1) \geq 2$. But (i) is impossible by (1) above and (ii) is impossible by the induction hypothesis and Lemma 2.5. Thus (iii) holds. Applying Lemma 2.6 to $W_1$, and using the same reasoning, we obtain a wall $W_2$ such that $h(W_2) \geq 1 + f(\omega)$ and $\pi(W_2) \geq 3$. Thus by (2) above, $W_2$ contains an induced subwall $W_3$ such that $\omega(W_3) < \omega$ and $h(W_3) \geq f(\omega) > f(\omega(W_3))$, which, using Lemma 2.5, contradicts the induction hypothesis. □

Let $W$ be a wall, which has the cross property with respect to $x$. Then for every $\alpha > \beta$ in $C(W)$, there exists a right witness point $y_\beta$ in $[\alpha]_W \cap N(x)$ for the pair $(\alpha, \beta)$. However, for different values of $\beta$, the right witness points $y_\beta$ may be distinct. We say that $y \in [\alpha]_W \cap N(x)$ is a left *-witness point for $\alpha$ if $y$ is a left witness point for every pair $(\alpha, \beta)$, with $\beta \in C(W)$ and $\alpha > \beta$. Similarly, $y' \in [\alpha]_W - N(x)$ is a right *-witness point for $\alpha$ if $y'$ is a right witness point for every pair $(\alpha, \beta)$, with $\beta \in C(W)$ and $\alpha > \beta$. We say that $W$ has *-witnesses for $\alpha$ if there exist left and right *-witnesses for $\alpha$. We say that $W$ has the *strong cross property* if for every color $\alpha \in C(W), W$ has *-witnesses for $\alpha$. In order to establish the existence of a relatively high wall with the strong cross property, we need the following lemma.

LEMMA 2.7. *There exists a function $j(h)$ such that, if $W$ is a wall in a graph $G, W$ has height $j = j(h)$, and $W$ supports a vertex $x$, then there exists an induced subwall $W'$ of $W$ such that $W'$ supports $x, h(W') \geq h$, and $(*)$ for every vertex $y$ in $W'$ and for all $\alpha > \beta$ in $C(W'), y$ is a left or right witness for $(\alpha, \beta)$ iff $y$ is a left or right *-witness for $\alpha$.*

*Proof.* Let $j(h) = 2^{2^h}$. We construct $W'$ one level at a time starting at the top. At each new level, we must add points to support all the points from higher levels already added to $W'$. In order to ensure that regardless of how we later add points at lower levels, these new points will satisfy $(*)$, we remove certain lower levels from consideration. This idea is formalized as follows.

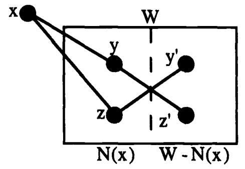


FIG. 8.

*Stage* 0. Let $I_0 = C(W), I'_0 = \varnothing$, and $v_0 = x$.
*Stage* $s+1$. Suppose we have constructed $V_s = \{v_0, \dots, v_n\}, I_n$, and $I'_s$ such that:
(1) $n < 2^s, |I_n| \geq j2^{-n}$, and $|I'_s| = s$;
(2) $\forall \alpha \in I_n \ \forall \beta \in I'_s (\alpha < \beta)$;
(3) $[I']_{W \cap V_s}$ is a wall which supports $x$ and satisfies (*);
(4) $\forall y \in (N(x) \cap V_s) \ \forall \alpha, \beta \in I'_s$,

$$[\exists z \in ([\alpha]_W \cap N(x))(y \sim z)] \Leftrightarrow [\exists z \in ([\beta]_W \cap N(x)(y \sim z)];$$

and

(5) $\forall y \in (V_s - N(x)) \ \forall \alpha, \beta \in I'_s$,

$$[\exists z \in ([\alpha]_W - N(x))(y \sim z)] \Leftrightarrow [\exists z \in ([\beta]_W - N(x))(y - z)].$$

Let $\alpha$ be the largest color in $I_n$. Set $I'_{s+1} = I'_s \cup \{\alpha\}$. For each $v_i \in V_s$, choose $v_{n+i} \in [\alpha]_W$ such that $v_i \sim v_{n+i}$. Set $V_{s+1} = \{v_0, \dots, v_{2n}\}$. Define $I_i$, for $i = n + 1, \dots, 2n$ by induction on $i$ as follows. Suppose $I_i$ has been defined. Let $J = \{\gamma \in I : \exists z \in [\gamma]_W [v_{n+i+1} \sim z \text{ and } (v_{n+i+1} \sim x \Leftrightarrow z \sim x)]\}$. If $|J| \geq |I_i|/2$, set $I_{i+1} = J$; otherwise set $I_{i+1} = I_i - (J \cup \{\alpha\})$. It is easy to check that conditions (1)–(5) are maintained. This completes the proof. □

Lemma 2.7 allows us to strengthen Lemma 2.6 as follows.

LEMMA 2.8. *There exists a function $g^*(h)$ such that for any graph $G = (V, E)$, if $W$ is a wall in $G$ and $h(W) \geq g^*(h)$, then there exists an induced subwall $W'$ of $W$ with $h(W') \geq h$ and*

(i) *$W'$ has the strong cross property; or*
(ii) *$\omega(W') < \omega(W)$; or*
(iii) *both $\omega(W') \leq \omega(W)$ and $\pi(W') > \pi(W)$.*

*Proof.* Let $g^*(h) = j \circ g(h)$. Suppose $h(W) \geq g^*(h)$. Then by Lemma 2.7 there exists an induced subwall $W_1 \subset W$ such that $h(W_1) \geq g(h)$ and $W_1$ satisfies (*). By Lemma 2.6, there exists an induced subwall $W' \subset W$ with $h(W') \geq h$, and either $W'$ is a level induced subwall of $W_1$ and has the cross property, $\omega(W') < \omega(W)$, or both $\omega(W') \leq \omega(W)$ and $\pi(W') > \pi(W)$. In the latter two cases, we are immediately done. In the first case we are also done, since $W_1$ satisfies (*) and $W'$ is a level induced subwall of $W_1$. □

LEMMA 2.9. *There exists a function $e(h, \omega)$ such that if $W$ is a wall in a graph $G$ with $G \in \text{Forb}(P_{5,1}), h(W) \geq e(h)$, and $\omega(G) \leq \omega$, then there exists an induced subwall $W' \subset W$ such that $h(W') \geq h$ and $W'$ has the strong cross property.*

*Proof.* The proof is essentially the same as the proof of Theorem 2.1, with Lemma 2.6 replaced by Lemma 2.8 and observation (2) replaced by the following remark: (2′) if $W$ is a wall in a graph $G$ with $\pi(W) \geq 3$, then $W$ contains a subwall $W_0$ such that $h(W_0) = h(W) - 1$ and $\omega(W_0) < \omega(W)$. Let $P = (x_1, x_2, x_3)$ be a path that witnesses that $\pi(W) \geq 3$. If $W_0 = W \cap N(x)$ is a wall, we are clearly done; otherwise, there exist

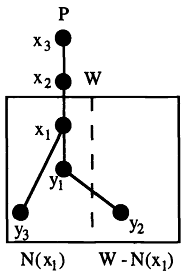


FIG. 9.

$y_1 \in [\alpha]_W$ with $x_1 \sim y_1$ and $y_2, y_3 \in [\beta]_W$, such that $y_1 \sim y_2, x_1 \not\sim y_2$, and $x_1 \sim y_3$, where $\alpha > \beta$. But then $\{x_3, x_2, x_1, y_1, y_2, y_3\}$ induces $P_{5,1}$, which is a contradiction. See Fig. 9. □

We note that Lemma 2.9 holds with $P_{5,1}$ replaced by $P_{n,1}$ or $P^2_{n,k}$. However, we as yet have no application for such results. We need one more lemma for the proof of Theorem 2.3.

LEMMA 2.10. *Let $G$ be a graph in* Forb$(P_{5,1})$ *with $\omega(G) \leq \omega$, let $x$ be a vertex of $G$, and let $W$ be a wall in $G$ such that both $h(W) \geq R(\omega+1, t)$ and $W$ has the strong cross property with respect to $x$. Then $G$ induces $B_t$.*

*Proof.* Let $r = R(\omega + 1, t)$, and for $1 \leq \alpha \leq r$, let $y_\alpha$ and $y'_\alpha$ denote the left and right *-witnesses for $\alpha$. First observe that for $1 \leq \beta < \alpha \leq r, y'_\alpha$ is adjacent to $y_\beta$. Otherwise, since $y'_\alpha$ is a right *-witness point, there exists $z \in N(x) \cap [\beta]_W$ such that $z \sim y'_\alpha$. Since $y_\alpha$ is a left *-witness point, there exists $z' \in [\beta]_W - N(x)$ such that $y_\alpha \sim z'$. But then $\{z', y_\alpha, x, z, y'_\alpha, y_\beta\}$ induces $P_{5,1}$, which is a contradiction. In particular, if $\alpha > \beta$, then the left *-witness for $\beta$ supports some vertex in $[\alpha]_W - N(x)$. See Fig. 10.

We call a vertex $z' \in [\gamma]_W - N(x)$ *special* for $\gamma$ if, for all $\alpha \neq \gamma, y_\alpha \sim z'$. We next show that for every color $\gamma \in C(W)$, there exists a vertex $z'_\gamma$ that is special for $\gamma$. For each $\alpha \neq \gamma$, let $N_\alpha = \{z' \in [\gamma]_W - N(x)\colon y_\alpha \sim z'\}$. We must show that $\cap_{\alpha \neq \gamma} N_\alpha \neq \varnothing$. Each $N_\alpha$ is nonempty. If $\alpha > \gamma$, then this follows from the definition of $y_\alpha$, and if $\alpha < \gamma$, then it follows from the observation above. Thus it suffices to show that for all $\alpha, \beta \in C(W) - \{\gamma\}, N_\alpha \subset N_\beta$ or $N_\beta \subset N_\alpha$. Suppose not. Then there exist $z', w' \in [\gamma]_W - N(x)$ such that $y_\alpha \sim z' \not\sim y_\beta$ and $y_\alpha \not\sim w' \sim y_\beta$. But then $\{z', y_\alpha, x, y_\beta, w', y_\gamma\}$ induces $P_{5,1}$, which is a contradiction. See Fig. 11.

Finally, by the choice of $r$, there exists a subset $H \subset C(W)$ such that $|I| = t$ and $\{z'_\gamma\colon \gamma \in I\}$ is independent. Then the set $\{y_\gamma\colon \gamma \in I\} \cup \{z'_\gamma\colon \gamma \in I\}$ induces $B_t$. □

Theorem 2.3 now follows easily from Lemmas 2.5, 2.9, and 2.10.

*Proof of Theorem* 2.3. Fix $t$. We claim that $f(\omega) = e(R(\omega + 1, t), \omega)$ is a $\chi$-binding function for Forb$(P_{5,1}, B_t)$. Suppose not. Then there exists a graph $G$ in Forb$(P_{5,1}, B_t)$ such that $\chi_{\mathrm{FF}}(G) \geq f(\omega(G))$. By Lemma 2.5, there is a wall $W$ in $G$ such that $h(W) \geq f(\omega(G))$. Thus by Lemma 2.9, there exists a wall $W$ in $G$ such that $h(W) \geq R(\omega + 1, t)$ and $W$ has the strong cross property. Thus by Lemma 2.10, $G$ induces $B_t$, which is a contradiction. □

*Proof of Theorem* 2.4. Since we are not concerned with finding an optimal binding function, we may assume that $k = t$. Let $f$ be defined recursively by $f(1) = 1$ and $f(\omega + 1) = j \circ R(1 + f(\omega), 1 + R_{16}(\max\{2t, \omega + 1\}))$, where $j$ is the function from Lemma 2.7. We shall show by induction on $\omega$ that, if $G \in$ Forb$(D_t, B_t)$ and $\omega(G) \leq \omega$, then $\chi_{\mathrm{FF}}(G) \leq f(\omega(G))$. The base step is trivial, so suppose the result holds for $\omega$ and suppose both $\omega(G) = \omega + 1$ and $\chi_{\mathrm{FF}}(G) > f(\omega + 1)$. Then, by Lemma 2.5, there

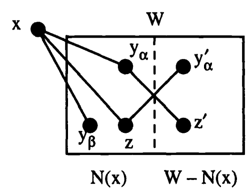


FIG. 10

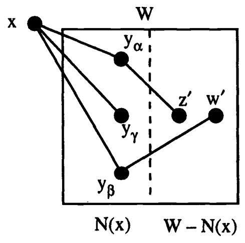


FIG. 11

exists a wall $W$ in $G$ of height $f(\omega + 1)$ that supports a vertex $x$. We shall obtain a contradiction in two steps. We first show (1) there exists a set of vertices $X = \{x, y, a_1, \ldots, a_s, b_1, \ldots, b_s\}$ such that $s = R_{16}(\max\{2t, \omega + 1\}), x \sim y, \{a_1, \ldots, a_s\} \subset N(x) - N(y), \{b_1, \ldots, b_s\} \subset N(y) - N(x)$, and $a_i \not\sim b_i$ for all $i$. We then show (2) there exists a subset of $X$ that induces either $D_t$ or $B_t$.

By Lemma 2.7 there exists an induced subwall $W_0 \subset W$ such that $W_0$ supports $x, h(W_0) \geq R(1 + f(\omega), 1 + R_{16}(\max\{2t, k + 1\})$, and (*) holds. Define a coloring $q$ on the two element subsets of $C(W_0)$ by $q(\alpha > \beta) = c$, where

$c = 2$ iff $\exists y \in ([\alpha]_{W_0} \cap N(x))\ \forall z \in ([\beta]_{W_0} \cap N(x))(y \not\sim z)$ and

$c = 1$ otherwise.

By Ramsey's theorem, there exists a homogeneous subset $H \subset C(W_0)$ such that either $q(\alpha > \beta) = 1$ for all $\alpha, \beta \in H$, and $|H| = 1 + f(\omega)$ or $q(\alpha > \beta) = 2$ for all $\alpha, \beta \in H$, and $|H| = 1 + R_{16}(\max\{2t, \omega + 1\})$. In the first case, $W_1 = [H]_{W_0}$ is a wall such that $\omega(W_1) \leq \omega$ and $h(W_1) \geq 1 + f(\omega)$, which by Lemma 2.5 contradicts the induction hypothesis. In the second case, for each $\gamma \in C(W_1)$, there exists a left *-witness $y_\gamma$ for $\gamma$. Let $y = y_\alpha$, where $\alpha$ is the largest color in $C(W_1)$, and let $a_i = y_{\gamma_i}$, where $\gamma_i$ is the $i$th smallest color of $C(W_1)$. Finally choose $b_i \in [\gamma_i]_{W_1}$, so that $y_\alpha \sim b_i$. It is now easy to check that $X = \{x, y, a_1, \ldots, a_s, b_1, \ldots, b_s\}$ has the desired properties for (1).

Define a coloring $r$ on the two element subsets of $[s]$ by $r(\beta > \gamma) = Y$, where $Y$ is the image of the graph $G_{\beta,\gamma} = G[\{a_\beta, b_\beta, a_\gamma, b_\gamma\}]$ under the graph isomorphism that maps $a_\beta, b_\beta, a_\gamma, b_\gamma$ to 1, 2, 3, 4, respectively. There are 16 possibilities for such graphs depending on which of four possible edges are present. Thus, by Ramsey's theorem, there exists a homogeneous subset $H'$ such that $|H'| \geq \max\{2t, \omega + 1\}$ and $r(\beta > \gamma) = Y$, for all $\beta, \gamma \in H'$. Let $A = \{a_i : i \in H'\}$ and $B = \{b_i : i \in H'\}$. Since $|H| \geq \omega + 1$ and $A \subset N(y), 1 \not\sim 3$ in $Y$, i.e., $A$ is an independent set. Similarly $2 \not\sim 4$ in $Y$ and $B$ is an independent set. This leaves four possibilities, which are illustrated in Figs. 12–15, for $Y$. If $Y$ has no edges, then $G[X]$ contains an induced $D_{2t}$; if $Y$ has one edge, then $G[X]$ contains an induced $D_t$; and if $Y$ has two edges, then $G[X]$ contains an induced $B_{2t}$. Each possibility is a contradiction, so we are done. □

**3. Open problems.** The problem of determining whether Forb($P_5$) has a polynomial on-line $\chi$-binding function remains open. In fact, this problem is open even in the off-line case; all that is known is that if $f$ is a $\chi$-binding function for Forb($P_5$), then $f$ satisfies $c(\omega/\log\omega)^2 \leq f(\omega) \leq 2^\omega$. The lower bound follows from an observation of Gyárfás [4]: if $\alpha(G) < 3$, then $G \in$ Forb($P_5$) and $\chi(G) \geq \nu(G)/2$, and thus $(R(\omega, 3) - 1)/2 \leq f(\omega)$. The result then follows from a well-known lower bound on $R(3, \omega)$. The upper bound is only a small improvement on the on-line $\chi$-binding function presented here.

For trees $T$ for which Forb($T$) is not $\chi_{\mathrm{FF}}$-bounded, it may be possible to determine the reason why. We have previously noted that Forb($T, K_{t,t}$) is $\chi_{\mathrm{FF}}$-bounded for any

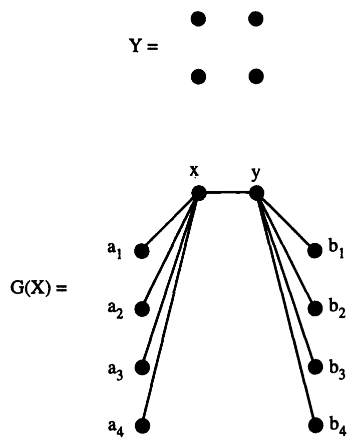


FIG. 12

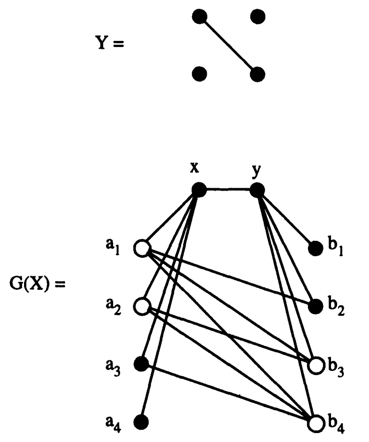


FIG. 13

tree. However, this does not tell us why Forb($T$) is not $\chi$-bounded. Our result that Forb($T, B_t$) is $\chi$-bounded for $T = D_k$ or $P_{k,1}$ is more informative, since $\chi_{\mathrm{FF}}(B_t) = t$. It would be interesting to prove similar results for other trees. However, the following two negative examples show that some caution is in order.

Gyárfás and Lehel's proof [7] that Forb($P_6$) is not on-line $\chi$-bounded actually shows more. Since the graphs they constructed do not induce $B_3$, their arguments show that Forb($P_6, B_3$) is not on-line $\chi$-bounded. Thus if $T$ is a tree with radius greater than two, then Forb($T, B_t$) is not on-line $\chi$-bounded. In particular, it is not $\chi_{\mathrm{FF}}$-bounded.

We next present an example which provides a general construction for graphs which force the First-Fit algorithm to use a large number of colors. This example includes $B_t$ as a special case.

*Example* 3.1. Let $t \geq 2$ and let $H = (V, E)$ be a graph such that

1. $V = A_1 \cup A_2 \cup \cdots \cup A_t$;
2. $A_j = \{a_{1j}, a_{2j}, \ldots, a_{jj}\}$ is a set of $j$ independent vertices for $j = 1, 2, \ldots, t$;
3. $A_j \cap A_{j+1} = \varnothing$ for $j = 1, 2, \ldots, t-1$;
4. $a_{ij} \not\sim a_{ij+1}$ whenever $1 \leq i \leq j \leq t-1$; and
5. $a_{ij} \sim a_{kj+1}$ whenever $1 \leq i < k \leq j+1 \leq t$.

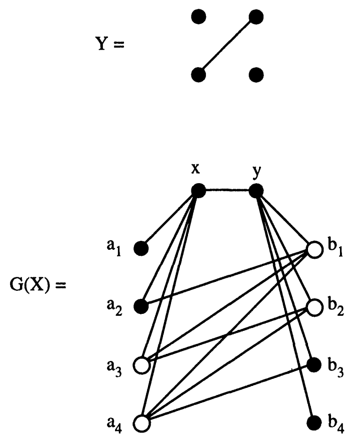


FIG. 14

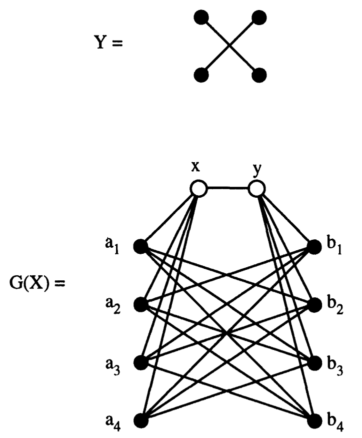


FIG. 15

Note that we do not require that $A_j \cap A_{j'} = \varnothing$ when $|j' - j| \geq 2$. Now let $|V| = n$ and let $v_1 < v_2 < \cdots < v_n$ be a linear order on $V$ so that $\alpha < \beta$ whenever $v_\alpha = a_{ij}, v_\beta = v_{kj+1}$, and $1 \leq i < k \leq j + 1 \leq t$. Then an easy inductive argument shows that the First-Fit algorithm will color $H$ with $t$ colors when the vertices are presented in this order; in fact, FF will color a vertex $v_\alpha = a_{ij}$ with color $i$.

The graph $B_t$ (actually $B_t$ with a single vertex removed) is obtained if $a_{ij} = a_{ij+2}$ whenever $1 \leq i \leq j \leq t - 2$. More generally, suppose there is some $k \geq 2$ so that $a_{ij} = a_{ij+k}$ whenever $1 \leq i \leq j \leq t - k$, and the only adjacencies in $G$ are those required by property 5 above. Then the chromatic number of the graph is three if $k$ is odd and two if $k$ is even.

On the other hand, if the sets $A_1, A_2, \ldots, A_t$ are pairwise disjoint and independent whenever $|j - i| \geq 2$, then we obtain a bipartite graph $H$ which is the complement of a comparability graph (a *cocomparability* graph). In [9], Kierstead used this example to show that First-Fit can be forced to use arbitrarily many cliques to cover a comparability graph with independence number two. Of course this implies that First-Fit can be forced to use arbitrarily many chains to cover a width-two ordered set.

Since covering a comparability graph with cliques is equivalent to coloring a cocomparability graph, and cocomparability graphs induce neither $LS_3$ nor $B_3$, it follows that $\mathrm{Forb}(LS_3, B_3)$ is not $\chi_{\mathrm{FF}}$-bounded.

Motivated by the results presented previously and the examples discussed above, we suggest the following problems.

*Problem* 1. Given a tree $T$, do there exist a function $g(\omega, \chi)$ and an integer $r$ such that if $G \in \mathrm{Forb}(T)$ and $\chi_{\mathrm{FF}}(G) > g(\omega(G), \chi)$, then there exists an induced subgraph $H$ of $G$ with $\chi(H) \leq r$ and $\chi_{\mathrm{FF}}(H) \geq \chi$?

*Problem* 2. Given a tree $T$, does there exist a function $h(\omega, \chi)$ such that if $G \in \mathrm{Forb}(T)$ and $\chi_{\mathrm{FF}}(G) > g(\omega(G), \chi)$, then $G$ contains an induced subgraph $H$ of the type constructed in Example 3.1 with $\chi_{\mathrm{FF}}(H) \geq \chi$?

*Problem* 3. Is $\mathrm{Forb}(L_k, B_t)$ $\chi_{\mathrm{FF}}$-bounded?

## REFERENCES

[1] D. Bean, *Effective coloration*, J. Symbolic Logic, 41 (1976), pp. 469–480.

[2] V. Chvátal, *Perfectly ordered graphs*, in Topics on Perfect Graphs, Ann. Discrete Math., 21 (1989), pp. 63–65.

[3] A. Gyárfás, *On Ramsey covering-numbers*, Colloq. Math. Soc. János Bolyai, 10 (1975), and in Infinite and Finite Sets, North–Holland/American Elsevier, New York, 1973, pp. 801–816.

[4] ———, *Problems from the world surrounding perfect graphs*, Zastos. Mat., XIX (1985), pp. 413–441.

[5] A. Gyárfás and J. Lehel, *On-line and first-fit coloring of graphs*, J. Graph Theory, 12 (1988), pp. 217–227.

[6] ———, *First-fit and on-line chromatic number of families of graphs*, Ars Combin., 29C (1990), pp. 160–167.

[7] ———, *Effective on-line coloring of $P_5$-free graphs*, Combinatorica, 11 (1991), pp. 181–184.

[8] A. Gyárfás, E. Szemerédi, and Z. Tuza, *Induced subtrees in graphs of large chromatic number*, Discrete Math., 30 (1980), pp. 235–244.

[9] H. A. Kierstead, *An effective version of Dilworth's theorem*, Trans. Amer. Math. Soc., 268 (1981), pp. 63–77.

[10] ———, *The linearity of First-Fit for coloring interval graphs*, SIAM J. Discrete Math., 1 (1988), pp. 526–530.

[11] H. A. Kierstead and S. G. Penrice, *Radius two trees specify $\chi$-bounded classes*, submitted.

[12] ———, *Recent results on a conjecture of Gyárfás*, Congr. Numer., to appear.

[13] H. A. Kierstead, S. G. Penrice, and W. T. Trotter, *On-line coloring and recursive graph theory*, SIAM J. Discrete Math., 7 (1994), pp. 72–89.

[14] H. A. Kierstead and W. T. Trotter, *An extremal problem in recursive combinatorics*, Congr. Numer., 33 (1981), pp. 143–153.

[15] D. P. Sumner, *Subtrees of a graph and chromatic number*, in The Theory and Applications of Graphs, Gary Chartrand, ed., John Wiley and Sons, New York (1981), pp. 557–576.

[16] D. R. Woodall, *Problem No.* 4, *Combinatorics*, Proc. British Combinatorial Conference, 1973, and London Math. Soc. Lecture Note Ser. 13, T. P. McDonough and V. C. Marvon, eds., Cambridge University Press, 1974, p. 202.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 499–506, November 1995

# ON THE $\lambda$-NUMBER OF $Q_n$ AND RELATED GRAPHS *

MARSHALL A. WHITTLESEY†, JOHN P. GEORGES† AND DAVID W. MAURO†

**Abstract.** An $L(2,1)$-labeling of graph $G$ is an integer labeling of $V(G)$ such that adjacent vertices have labels that differ by at least 2 and such that vertices distance 2 apart have labels that differ by at least 1. The $\lambda$-number of $G, \lambda(G)$, is the minimum range over all $L(2,1)$-labelings. We examine the properties of $\lambda$-labelings of the $n$-cube $Q_n$. Griggs and Yeh have determined $\lambda(Q_n)$ for $n \leq 5$ and have established $n+3 \leq \lambda(Q_n) \leq 2n+1$ for $n \geq 6$. We modify a technique used in coding theory to improve the upper bound. We also examine the $\lambda$-labelings of related graphs, such as the subdivision of the $n$-cube and the Cartesian products of paths.

**Key words.** $\lambda$-labeling, $n$-cube, vertex labeling

**AMS subject classification.** Primary, 05C

## 1. Introduction.

An $L(2,1)$-labeling of a graph $G$ is an integer assignment $f$ to the vertices of $G$ such that

(i) $|f(v) - f(w)| \geq 2$ if $v$ and $w$ are adjacent, and

(ii) $|f(v) - f(w)| \geq 1$ if the shortest path from $v$ to $w$ is of length 2.

The problem of finding an $L(2,1)$-labeling of $G$, which is a variation of the well-known channel assignment problem (see [3], [5], [6]), was first introduced in 1988 by Roberts in private communication with Griggs.

Let $\mathcal{L}(G)$ be the collection of all $L(2,1)$-labelings of $G$. For any $f \in \mathcal{L}(G)$, we define the span of $f, s(f)$ to be the absolute difference between a maximum and minimum vertex assignment of $f$. We define the $\lambda$-number of $G, \lambda(G)$ to be $\min_{f\in\mathcal{L}(G)} s(f)$. If $s(f) = \lambda(G)$, then we say that $f$ is a $\lambda$-labeling of $G$. With no loss of generality, we will assume that the smallest label of a labeling is 0.

Griggs and Yeh [2] have investigated the relationship between $\lambda(G)$ and other graph invariants, such as chromatic number $\chi(G)$ and maximum vertex degree $\Delta(G)$. Sakai [7] has obtained bounds on the $\lambda$-number of unit interval graphs and chordal graphs in general. Georges, Mauro, and Whittlesey [1] have established a relationship between $\lambda(G)$ and the path-covering number of $G^c$.

In this paper, we focus our attention on $L(2,1)$-labelings of the $n$-cube $Q_n$ and related graphs. Griggs and Yeh [2] have shown $\lambda(Q_n) \leq 2n+1$, and Jonas [2] has shown that $n+3 \leq \lambda(Q_n)$. By constructive methods, we find the $\lambda$-number of the product of sufficiently long paths and reproduce the upper bound of $2n+1$ in a more general setting. Next, we apply a coding theory approach to the algebraic structure of $Q_n$ in order to improve the upper bound of $2n+1$. This bound ranges from $\lfloor n+1+\log_2 n \rfloor$ to $2n$, depending on the value of $n$. Finally, we investigate the $\lambda$-number of the subdivision of $G$ (denoted $G^S$) and obtain $\lambda(Q_n^S)$.

## 2. The products of paths and $Q_n$.

We will begin our discussion by investigating the $\lambda$-number of the product $P = \prod_{i=1}^{n} P^i$ of nontrivial paths $P^i$, where $P^i$ is a path of $p_i$ vertices. For certain values of $p_i$, we obtain the exact value of $\lambda(P)$. Since $Q_n$ (the product of $n$ copies of $P_2$) is a subgraph of $P$, we obtain an upper bound for $\lambda(Q_n)$.

* Received by the editors September 30, 1992; accepted for publication (in revised form) July 25, 1994.

† Department of Mathematics, Trinity College, Hartford, Connecticut 06106.

It is convenient to represent a vertex $v$ of $P$ by the $n$-tuple $(v_1, v_2, v_3, \ldots, v_n)$ of integers, where $1 \le v_i \le p_i$. It follows that a shortest path between the vertices $u = (u_1, u_2, \ldots, u_n)$ and $w = (w_1, w_2, \ldots, w_n)$ has length $\sum_{i=1}^{n} |u_i - w_i|$. Vertices $u$ and $w$ are therefore adjacent if and only if, for some $j$,

$$|w_j - u_j| = 1 \quad \text{and} \quad |w_i - u_i| = 0, \qquad i \ne j.$$

Similarly, vertices $u$ and $w$ are distance two apart if and only if one of the following two cases holds:

(1) there exists $j, 1 \le j \le n$, such that

$$|w_j - u_j| = 2 \quad \text{and} \quad |w_i - u_i| = 0, \qquad i \ne j;$$

(2) there exist $j, k, 1 \le j < k \le n$, such that

$$|w_j - u_j| = |w_k - u_k| = 1 \quad \text{and} \quad |w_i - u_i| = 0, \qquad i \ne j, k.$$

LEMMA 2.1 [2]. *Let $G$ be a graph with maximum degree $\Delta \ge 2$. Then the following statements hold.*

(i) $\lambda(G) \ge \Delta + 1$.

(ii) *If $\lambda(G) = \Delta + 1$, then each vertex of degree $\Delta$ is assigned* 0 *or* $\Delta + 1$ *under all $\lambda$-labelings.*

(iii) *If $G$ has three vertices of degree $\Delta$ such that one such vertex is adjacent to the other two, then $\lambda(G) \ge \Delta + 2$.*

LEMMA 2.2. *Let $p_i \ge 3$ for $1 \le i \le n$. If there exists integer $j$ such that $p_j \ge 5$, or if there exist distinct integers $j$ and $k$ such that $p_j = p_k = 4$, then $\lambda(P) \ge 2n + 2$.*

*Proof.* Let $x$ be the vertex in $P$ each of whose coordinates is 2. If $p_j \ge 5$, then let $y$ be the vertex adjacent to $x$ whose $j$th coordinate is 3, and let $z$ be the vertex adjacent to $y$ whose $j$th coordinate is 4. Since each of these vertices has degree $\Delta = 2n$, Lemma 2.1 implies $\lambda(P) \ge 2n + 2$. If $p_j = p_k = 4$, then let $y$ and $z$ be the vertices adjacent to $x$ whose $j$th and $k$th coordinates are 3, respectively. Since each of these vertices has degree $2n$, the result again follows from Lemma 2.1. □

LEMMA 2.3. *Let $p_n = 2$ and $p_i \ge 3, 1 \le i \le n - 1$. In addition, let $p_j \ge 4$ for some $j$. Then $\lambda(P) \ge 2n + 1$.*

*Proof.* Let $x$ be the vertex each of whose coordinates is 2. Let $y$ and $z$ be the vertices adjacent to $x$ whose $n$th coordinate is 1 and whose $j$th coordinate is 3, respectively. Since each of these vertices has degree $\Delta = 2n - 1$, the result follows from Lemma 2.1. □

In the proof of the next lemma, which establishes corresponding upper bounds for the $\lambda$-number of products of paths, we make use of the following fact: If $a, b$, and $m$ are positive integers such that $|a - b| < m$, then

$$|a \bmod m - b \bmod m| = |a - b| \quad \text{or} \quad m - |a - b|.$$

Our general method of proof is similar to that used by Griggs and Yeh [2] when they obtained an upper bound of $2n + 1$ for $\lambda(Q_n)$.

LEMMA 2.4. *If $p_i \ge 2$ for all $i$, then $\lambda(P) \le 2n + 1 + r$, where $r = 0$ when $p_n = 2$ and $r = 1$ otherwise.*

*Proof.* Let vertex $(v_1, v_2, \ldots, v_n)$ be assigned the integer $[\![\sum_{i=1}^{n} (i+1)v_i]\!] \bmod (2n + 2 + r)$. Then for some $k, 1 \le k \le n$, adjacent vertices have labels whose absolute difference is $k + 1$ or $2n + 2 + r - (k + 1)$, each of which is greater than or equal to 2.

If $w$ and $u$ are at distance 2, then the $n$-tuple representations of $w$ and $u$ differ by precisely 1 in two positions or precisely 2 in one position.

If the latter condition holds, then $\sum_{i=1}^{n}(i+1)w_i$ and $\sum_{i=1}^{n}(i+1)u_i$ differ by $2(k+1)$ for some $k, 1 \leq k \leq n-1+r$. Since $2(k+1)$ is less than the modulus, we have that

$$\left| \left[\!\!\left[ \sum_{i=1}^{n}(i+1)w_i \right]\!\!\right] \bmod(2n+2+r) - \left[\!\!\left[ \sum_{i=1}^{n}(i+1)u_i \right]\!\!\right] \bmod(2n+2+r) \right|$$

equals either $2k+2$ or $2n+2+r-(2k+2)$. Since neither of these terms is 0, the distance-2 condition is satisfied in this case.

If the former condition holds, then $\sum_{i=1}^{n}(i+1)w_i$ and $\sum_{i=1}^{n}(i+1)u_i$ differ by $|(k_1+1)-(k_2+1)|$ or $|(k_1+1)+(k_2+1)|$ for some distinct $k_1, k_2$. Since each of these is less than the modulus, we have that

$$\left| \left[\!\!\left[ \sum_{i=1}^{n}(i+1)w_i \right]\!\!\right] \bmod(2n+2+r) - \left[\!\!\left[ \sum_{i=1}^{n}(i+1)u_i \right]\!\!\right] \bmod(2n+2+r) \right|$$

equals $|(k_1+1)-(k_2+1)|$ or $2n+2+r-|(k_1+1)-(k_2+1)|$ or $(k_1+1)+(k_2+1)$ or $2n+2+r-((k_1+1)+(k_2+1))$. Since none of these four terms is 0, the distance-2 condition is again satisfied. Hence, vertices at distance 2 have distinct labels. □

Combining Lemmas 2.2, 2.3, and 2.4, we obtain the following theorem.

THEOREM 2.5.

(i) *Suppose* $p_i \geq 3$ *for* $1 \leq i \leq n$. *If there exists an integer* $j$ *such that* $p_j \geq 5$, *or if there exist distinct integers* $j$ *and* $k$ *such that* $p_j = p_k = 4$, *then* $\lambda(P) = 2n+2$.

(ii) *Suppose* $p_n = 2$ *and* $p_i \geq 3, 1 \leq i \leq n-1$. *If there exists an integer* $j$ *such that* $p_j \geq 4$, *then* $\lambda(P) = 2n+1$.

## 3. A coding theory approach to $\lambda(Q_n)$.

In this section, we use coding theory methods (see [4]) to improve the bounds on $\lambda(Q_n)$. With vertices of $Q_n$ represented as binary $n$-tuples, we find a linear mapping $M : V(Q_n) \to V(Q_k)$ and an injection $f : V(Q_k) \to N$ such that $M \circ f$ is an $L(2,1)$-labeling. The mapping $M$ shall be represented by an $n \times k$ binary matrix (also denoted $M$), and the $k$-tuple $(v)M$ shall be the matrix product $v * M$ whose calculation is in binary arithmetic. Similarly, the injection $f$ shall be represented by a $k \times 1$ matrix (also denoted $f$) such that $(w)f = wf$. Hence, $(v)(M \circ f) = (v * M)f \equiv v * Mf$.

In the following discussion, $e_i$ shall denote the $i$th row of the $n \times n$ identity matrix. Thus $\varepsilon_i * M$ is the $i$th row of $M$. Our arithmetic for the most part will be binary, and the operation of binary subtraction (addition) will be denoted by $\ominus$.

We now derive some conditions on $M$ necessary for $M \circ f$ to be an $L(2,1)$-labeling.

LEMMA 3.1. *Adjacent vertices of* $Q_n$ *have different labels under* $M \circ f$ *if and only if no row of* $M$ *is the zero vector.*

*Proof.* ($\Leftarrow$) Suppose adjacent vertices $v$ and $w$ have the same labels under $M \circ f$, and suppose that $v \ominus w = e_i$. Then $v*Mf = w*Mf$, which implies that $v*M = w*M$, by the injectivity of $f$. This in turn implies that $(v \ominus w) * M = e_i * M = \vec{0}$.

($\Rightarrow$) Suppose that the $i$th row of $M$ is $\vec{0}$. Then $e_i * M = \vec{0} * M = \vec{0}$, which implies that $e_i * Mf = \vec{0} * Mf$. Adjacent vertices $\vec{0}$ and $e_i$ thus have identical labels. □

LEMMA 3.2. *Vertices of* $Q_n$ *that are at distance* 2 *have distinct labels if and only if the rows of* $M$ *are distinct.*

*Proof.* ($\Rightarrow$) Suppose that rows $i$ and $j$ in $M$ are identical. Then $e_i * M = e_j * M$, which implies that $e_i * Mf = e_j * Mf$. Hence, $e_i$ and $e_j$ have the same label. However, $e_i$ and $e_j$ are at distance 2.

($\Leftarrow$) Suppose that vertices $v$ and $w$ are at distance two and have identical labels. Then $v * Mf = w * Mf$, which implies that $v * M = w * M$, or $(v \ominus w) * M = \vec{0}$. However, $v \ominus w$ can be expressed as $e_i \ominus e_j$ for some $i, j$ due to the distance constraint. Hence, $e_i * M \ominus e_j * M = \vec{0}$, which implies $e_i * M = e_j * M$. Consequently, rows $i$ and $j$ of $M$ are not distinct. □

Next, suppose that $f$ is the column vector $(a_1, a_2, \dots, a_k)^T$ of positive integers. We define

$$B(f) = \left\{ (b_1, b_2, \dots, b_k) | b_i = 0, 1, \text{ or } -1 \text{ and } \left| \sum_{i=1}^{k} a_i b_i \right| = 1 \right\}$$

and

$$B'(f) = \{ (b'_1, b'_2, \dots, b'_k) | b'_i = b_i \bmod 2 \text{ and } (b_1, b_2, \dots, b_k) \in B(f) \}.$$

We note that $B'(f)$ is a set of binary $k$-tuples and hence contains elements of $V(Q_k)$.

LEMMA 3.3. *Let $M : Q_n \to Q_k$. If the rows of matrix $M$ are nonzero and distinct, and if $n \geq 2^{k-1}$, then the mapping $M$ is onto.*

*Proof.* If the rank of $M$ is less than or equal to $k - 1$, then there are at most $2^{k-1} - 1$ nonzero, distinct rows in $M$. However, $n \geq 2^{k-1}$; hence the rank of $M$ is $k$. Since the dimension of the image of $M$ is the rank of $M$, then $M$ maps $Q_n$ onto $Q_k$. □

LEMMA 3.4. *Let $M : Q_n \to Q_k$. If the rows of matrix $M$ are nonzero and distinct, and if $n \geq 2^{k-1}$, then adjacent vertices have labels that differ by at least 2 under $M \circ f$ if and only if no row of $M$ is an element of $B'(f)$.*

*Proof.* We prove that at least one row of $M$ is in $B'(f)$ if and only if there exist adjacent vertices whose labels differ by 1 or 0 under $M \circ f$.

($\Leftarrow$) Let $v$ and $w$ be adjacent vertices of $Q_n$ which differ in the $r$th coordinate. Denote $v * M$ and $w * M$ by $(v_1^m, v_2^m, v_3^m, \dots, v_k^m)$ and $(w_1^m, w_2^m, w_3^m, \dots, w_k^m)$, respectively. We have $|v * Mf - w * Mf| < 2$,
$\Rightarrow |v * Mf - w * Mf| = 1$ (by Lemma 3.1),
$\Rightarrow |(v_1^m, v_2^m, v_3^m, \dots, v_k^m)f - (w_1^m, w_2^m, w_3^m, \dots, w_k^m)f| = 1,$
$\Rightarrow |\sum a_i v_i^m - \sum a_i w_i^m| = 1,$
$\Rightarrow |\sum a_i (v_i^m - w_i^m)| = 1,$
$\Rightarrow (v_1^m - w_1^m, v_2^m - w_2^m, \dots, v_k^m - w_k^m) \in B(f),$
$\Rightarrow (v_1^m \ominus w_1^m, v_2^m \ominus w_2^m, \dots, v_k^m \ominus w_k^m) \in B'(f),$
$\Rightarrow (v * M \ominus w * M) \in B'(f),$
$\Rightarrow (v \ominus w) * M \in B'(f),$
$\Rightarrow e_r * M \in B'(f).$

($\Rightarrow$) Suppose there exists a row $b' = e_r * M$ in $B'(f)$. Let $b = (b_1, b_2, \dots, b_k)$ be an element of $B(f)$ such that $b'_i = b_i \bmod 2$ and $|bf| = |\sum_{i=1}^{k} a_i b_i| = 1$. For $1 \leq i \leq k$, let $x_i = 0$ if $b_i = 0$ or $-1$; 1 otherwise.

Since $M$ is onto, there is a vertex $v$ of $Q_n$ such that $v * M = (x_1, x_2, \dots, x_k)$. The adjacent vertices $v$ and $v \ominus e_r$ then have labels that differ by

$$|v * Mf - (v \ominus e_r) * Mf| = |xf - (x \ominus b')f| = |(x - (x \ominus b'))f|.$$

However, $x_i - (x_i \ominus b'_i) = b_i$. Thus $|(x - (x \ominus b'))f| = |bf| = 1$, which implies that adjacent vertices $v$ and $v \ominus e_r$ have labels that differ by 1. □

We combine Lemmas 3.1, 3.2, and 3.4 to obtain the following theorem.

THEOREM 3.5. *Suppose $f$ is an injective linear mapping from $V(Q_k)$ into the nonnegative integers such that $(x_1, x_2, \dots, x_k)f = (x_1, x_2, \dots, x_k)(a_1, a_2, \dots, a_k)^T =$*

*$\sum_{i=1}^{k} a_i x_i$, where $a_i$ is a positive integer. Suppose also that $M$ is a linear mapping (represented by an $n \times k$ matrix) from $V(Q_n)$ to the binary $k$-tuples such that $n \geq 2^{k-1}$ and $(v_1, v_2, \ldots, v_n)M = (v_1, v_2, \ldots, v_n) * M$. Then $M \circ f : V(Q_n) \rightarrow N$ is an $L(2,1)$-labeling of $Q_n$ if and only if the rows of $M$ are nonzero, distinct, and not in $B'(f)$.*

We next find an injection $f = (a_1, a_2, \ldots, a_k)^T$, from which $B'(f)$ and, hence, a suitable $M$ follow. An upper bound on $\lambda(Q_n)$ is then $\sum_{i=1}^{k} a_i$. Since the rows of $M$ must be selected from the complement of $B'(f)$, it is important to note that the complement must have at least $n$ elements in order for $M$ to be comformable with $(v_1, v_2, \ldots, v_n)$. This, however, can be guaranteed by characterizing the complement and choosing $k$ sufficiently large.

LEMMA 3.6. *Let $r_i = 2^{i-1}, i = 1, 2, 3, \ldots, k$, and let $\varepsilon_j$ denote the $j$th row of the $k \times k$ identity matrix. For fixed $q, 1 \leq q \leq k+1$, let*

$$s_i = 0 \quad \text{if } i = 1, 2, 3, \ldots, q-1; \qquad s_i = 2^{i-q} \quad \text{if } i = q, q+1, q+2, \ldots, k.$$

*Set $f = (a_1, a_2, \ldots, a_k)$, where $a_i = r_i + s_i$. Then $f$ is injective, and $B'(f) = \{\sum_{j=1}^{m} \varepsilon_j | m = 1, 2, 3, \ldots, q-1\}$, where this set is understood to be empty if $q = 1$.*

*Proof.* To establish the injectivity of $f$, we need only note that $\sum_{i=1}^{k} r_i x_i$ and $\sum_{i=1}^{k} s_i x_i$ are, respectively. strictly increasing and increasing functions of the binary integer $(x_k, x_{k-1}, \ldots, x_1)$.

To prove that $B'(f) = \{\sum_{j=1}^{m} \varepsilon_j | m = 1, 2, \ldots, q-1\}$, let $b' = (b'_1, b'_2, \ldots, b'_k) \in B'(f)$ and $b = (b_1, b_2, \ldots, b_k) \in B(f)$, where $b'_i = b_i \bmod 2$. Let $c$ denote $\max\{i | 1 \leq i \leq k \text{ and } b_i \neq 0\}$. Without loss of generality, we may assume that $b_c = 1$ since $|\sum_{i=1}^{k} 2^{i-1} b_i| = 1$ if and only if $|\sum_{i=1}^{k} 2^{i-1}(-b_i)| = 1$. By repeated application of the triangle inequality, we have

$$\begin{aligned}
2^{c-1} + s_c &= r_c + s_c \\
&= |(r_c + s_c) b_c| \\
&= \left| \sum_{i=1}^{c} (r_i + s_i) b_i + \sum_{i=1}^{c-1} (r_i + s_i)(-b_i) \right| \\
&\leq \left| \sum_{i=1}^{c} (r_i + s_i) b_i \right| + \left| \sum_{i=1}^{c-1} (r_i + s_i)(-b_i) \right| \\
&= 1 + \left| \sum_{i=1}^{c-1} (r_i + s_i)(-b_i) \right| \qquad (3.1) \\
&\leq 1 + \sum_{i=1}^{c-1} (r_i + s_i) |(-b_i)| \\
&\leq 1 + \sum_{i=1}^{c-1} (r_i + s_i) \\
&= 1 + \sum_{i=1}^{c-1} r_i + \sum_{i=1}^{c-1} s_i \\
&= 1 + \sum_{i=1}^{c-1} 2^{i-1} + \sum_{i=1}^{c-1} s_i \\
&= 2^{c-1} + \sum_{i=1}^{c-1} s_i.
\end{aligned}$$

If $c \geq q$, then $\sum_{i=1}^{c-1} s_i = \sum_{i=q}^{c-1} 2^{i-q} = 2^{c-q} - 1 < 2^{c-q} = s_c$. By the first and last lines of (3.1), we have the contradiction that $2^{c-1} + s_c \leq 2^{c-1} + \sum_{i=1}^{c-1} s_i < 2^{c-1} + s_c$. Hence $c \leq q - 1$. However, if $c \leq q - 1$, then $\sum_{i=1}^{c-1} s_i = s_c = 0$. Thus all of the inequalities in (3.1) are, in fact, equalities, implying

$$2^{c-1} = 1 + \sum_{i=1}^{c-1} 2^{i-1}|(-b_i)|.$$

This equality, constrained by the conditions $b_c = 1$ and $b_i = 0$ for $i > c$, is solved only by $b_i = -1$ for $1 \leq i \leq c-1$. Thus $b = (-1, -1, \ldots, -1, 1, 0, \ldots, 0)$. This implies that $(1, 1, \ldots, 1, 1, 0, \ldots, 0) = \sum_{j=1}^{c} \varepsilon_j \in B'(f)$. Since $c$ ranges between 1 and $q-1$, the theorem is proved. $\square$

We are now able to improve the bound on the $\lambda$-number of $Q_n$. Let $n \geq 2^{k-1}, 1 \leq q \leq k+1$, and $f = (r_1 + s_1, r_2 + s_2, \ldots, r_k + s_k)$ as above. Then by Lemma 3.6, $f$ is injective and $B'(f)$ contains $q - 1$ $k$-tuples. Thus there are $2^k - q$ nonzero $k$-tuples in the complement of $B'(f)$ from which we may form rows of $M$ so that, by Theorem 3.5, $M \circ f$ is an $L(2,1)$-labeling. Since $M$ is onto, there exists a vertex $v$ in $Q_n$ such that $v * M = (1, 1, \ldots, 1)$. Consequently, the span of $M \circ f$ is $\sum_{i=1}^{k}(r_i + s_i) = 2^k + 2^{k-q+1} - 2$. Furthermore, $2^k - q$ must be greater than or equal to $n$ in order that the product $v * M$ be conformable. Thus we have the following theorem.

THEOREM 3.7. *Let $2^{k-1} \leq n \leq 2^k - q$, where $1 \leq q \leq k+1$. Then $\lambda(Q_n) \leq 2^k + 2^{k-q+1} - 2$. In particular, $\lambda(Q_{2^k - q}) \leq 2^k + 2^{k-q+1} - 2$.*

*Examples.* Setting $k = 4$ and $q = 5, 4, 3, 2, 1$, we have $\lambda(Q_{11}) \leq 15, \lambda(Q_{12}) \leq 16, \lambda(Q_{13}) \leq 18, \lambda(Q_{14}) \leq 22$, and $\lambda(Q_{15}) \leq 30$, respectively. Suppose $n = 80$. Then, subject to the constraints $2^{k-1} \leq n \leq 2^k - q$ and $1 \leq q \leq k+1$, the upper bound on $\lambda(Q_{80})$ is minimized at $k = 7$ and $q = 8$. Hence, $\lambda(Q_{80}) \leq \lambda(Q_{120}) \leq 127$.

COROLLARY 3.7.1. $\liminf \lambda(Q_n)/n = 1$.

*Proof.* If $q = k+1$, it follows immediately from Theorem 3.7 that $\liminf \lambda(Q_n)/n \leq 1$. Because $\lambda(Q_n) \geq n + 3$ for all $n \geq 3$, we conclude that $\liminf \lambda(Q_n)/n = 1$. $\square$

THEOREM 3.8. *For all $n, \lambda(Q_n) \leq 2n$.*

*Proof.* Suppose that $n$ is of the form $2^k - q, 1 \leq q \leq k+1$. If $q = k+1$, then, by Theorem 3.7, $\lambda(Q_n) = \lambda(Q_{2^k-k-1}) \leq 2^k - 1$, which is less than $2n$ if $k \geq 3$ $(n \geq 4)$. If $q < k+1$, then, by Theorem 3.7, $\lambda(Q_n) = \lambda(Q_{2^k - q}) \leq 2^k + 2^{k-q+1} - 2$, which can be shown to be less than or equal to $2n$.

If $n$ is not of the form $2^k - q, 1 \leq q \leq k+1$, let $r$ be the smallest integer such that $2^{r-1} \leq n \leq 2^r - (r+1)$. Then $\lambda(Q_n) \leq \lambda(Q_{2^r-(r+1)}) \leq 2^r - 1 < 2^r \leq 2n$. $\square$

We note that $\lambda(Q_n)$ is strictly less than $2n$ for all $n > 2$ that are not of the form $2^k - 1$.

These methods can be extended to $L(a,b)$ labelings.

**4. The subdivision of a graph and $Q_n^S$.** In this section, we obtain bounds on the $\lambda$-number of the subdivision of a graph $G$. These bounds are expressed in terms of the maximum vertex degree $\Delta(G)$, the vertex chromatic number $\chi(G)$, and the edge chromatic number $\chi'(G)$.

The subdivision of $G$, denoted by $G^S$, is the graph obtained by inserting one vertex along each edge of $G$. A vertex of $G^S$ that is also a vertex of $G$ is called an old vertex; otherwise, it is called a new vertex. We observe that old vertices are adjacent to new vertices in $G^S$ only, and vice versa. If $G$ has $n$ vertices and $m$ edges, then $G^S$ has $m + n$ vertices and $2m$ edges.

THEOREM 4.1. *For $k \geq 1$, if $G$ is $k$-regular, then $\lambda(G^S) \geq k+2$.*

*Proof.* If $k = 1$, then $G^s$ is the sum of paths of length 3, and hence $\lambda(G^s) = 3$.

If $k \geq 2$, then $\Delta(G^s) = k$ and, by Lemma 2.1, $\lambda(G^s) \geq k+1$. Suppose $\lambda(G^s) = k+1$. Then each new vertex is adjacent to an old vertex labeled 0 and an old vertex labeled $k+1$, and so each new vertex must have a label in $\{2, 3, 4, \ldots, k-1\}$. However, for any old vertex $v$, the $k$ new vertices adjacent to $v$ must receive distinct labels, an impossibility. Therefore, $\lambda(G^s) \geq k+2$. □

THEOREM 4.2. $\lambda(G^S) \leq \chi(G) + \chi'(G)$.

*Proof.* This is a proof by construction. Let $C : V(G) \to \{0, 1, 2, \ldots, \chi(G)-1\}$ be a vertex coloring of $V(G)$, and let $C' : E(G) \to \{\chi(G)+1, \chi(G)+2, \ldots, \chi(G)+\chi'(G)\}$ be an edge coloring of $E(G)$. These colorings induce an $L(2,1)$-labeling of $V(G^S)$ as follows: $L(v) = C(v)$ or $C'(e)$ if, respectively, $v$ is an old vertex or $v$ is a new vertex inserted along edge $e$ in $G$. □

THEOREM 4.3. *If $G$ is a $k$-regular, bipartite graph, $k \geq 1$, then $\lambda(G^S) = k+2$.*

*Proof.* If $G$ is bipartite, then $\chi(G) = 2$ and $\chi'(G) = \Delta(G)$. Thus $k+2 \leq \lambda(G^S) \leq 2+k$. □

COROLLARY 4.3.1. $\lambda(Q_n^S) = n+2$.

We next obtain an upper bound for $\lambda(G^S)$ in terms of $\Delta(G)$. To this end, we recall that Vizing's theorem establishes the edge-chromatic number of a simple graph to be $\Delta$ or $\Delta+1$ and that Brooks's theorem places an upper bound of $\Delta$ on the vertex-chromatic number of a connected graph that is neither an odd cycle nor a complete graph.

THEOREM 4.4. *For any graph $G, \lambda(G^S) \leq 2\Delta+1$.*

*Proof.* It suffices to show that if $G$ is connected, then $\lambda(G^S) \leq 2\Delta+1$.

Let $G$ be a connected graph. We consider separately the cases where $G$ is an odd cycle, $G$ is a complete graph, and $G$ is neither an odd cycle nor a complete graph.

*Case* 1. *$G$ is an odd cycle.* Griggs and Yeh [2] have shown that the $\lambda$-number of any cycle is 4. Since $G^S$ is an even cycle, $\lambda(G^S) = 4 < 2\Delta+1$.

*Case* 2. *$G$ is a complete graph.* We observe that $\chi(G) = \Delta(G)+1$. If $G$ has an even number of vertices, then $\chi'(G) = \Delta(G)$, and, by Theorem 4.2, it follows that $\lambda(G) \leq 2\Delta(G)+1$. If $G$ has an odd number of vertices, let $G^S$ have vertices $v_{i,j}(1 \leq i, j \leq n)$, where $v_{i,i}$ corresponds to the $i$th vertex in $G$ and $v_{i,j}$ to the vertex for the edge joining the $i$th and $j$th vertices in $G, i \neq j$. Define $f(v_{i,j}) = 2(i+j \bmod n)$ for all vertices $v_{i,j} \in V(G^S)$. Note that two vertices $v_{i,i'}$ and $v_{j,j'}$ are of distance at most 2 in $G^S$ if and only if $i = i' \neq j = j'$ or $|\{i, i'\} \cap \{j, j'\}| = 1$. In any case, $|f(v_{i,i'}) - f(v_{j,j'})| = 2|(i+i') \bmod n - (j+j') \bmod n| \geq 2$, since $i+i' \not\equiv j+j' (\bmod n)$.

*Case* 3. *$G$ is neither an odd cycle nor a complete graph.* By Brooks's theorem and Vizing's theorem, respectively, $\chi(G) \leq \Delta(G)$ and $\chi'(G) \leq \Delta(G)+1$. Consequently, by Theorem 4.2, it follows that $\lambda(G) \leq 2\Delta(G)+1$. □

**Acknowledgments.** The authors would like to thank the referees for their comments, which resulted in a much improved paper.

## REFERENCES

[1] J. P. GEORGES, D. W. MAURO, AND M. A. WHITTLESEY, *On $\lambda$-numbers and path coverings*, Discrete Math., 135 (1994), pp. 103–111.

[2] J. R. GRIGGS AND R. K. YEH, *Labelling graphs with a condition at distance* 2, SIAM J. Discrete Math., 5 (1992), pp. 586–595.

[3] W. K. HALE, *Frequency assignment: Theory and application*, in Proc. Institute for Electrical and Electronics Engineers, 68 (1980), pp. 1497–1514.

[4] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, North–Holland, New York, 1977.

[5] F. S. Roberts, *From garbage to rainbows: Generalizations of graph coloring and their applications*, in Proceedings of the 6th International Conference on Theory and Applications of Graphs, Wiley, New York, 1990.

[6] F. S. Roberts, *$T$-colorings of graphs: Recent results and open problems*, Discrete Math., 93 (1991), pp. 229–245.

[7] D. Sakai, *Labeling chordal graphs: Distance-two condition*, SIAM J. Discrete Math., 7 (1994), pp. 133–140.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 507–516, November 1995

# THE MEDIAN PROCEDURE IN A FORMAL THEORY OF CONSENSUS *

F. R. MCMORRIS† AND R. C. POWERS†

**Abstract.** A consensus rule on a finite set $X$ is a function $c$ from the set of $k$-tuples for all $k > 0$ into the set of nonempty subsets of $X$. Elements in the image of $c$ represent a consensus, or agreement, of the input. Axioms for consensus rules are presented, and when $X$ is partially ordered, some consequences of these axioms are determined. A generalization of the median consensus rule is given when $X$ is a distributive semilattice and is based on a weighting of the least move metric on the covering graph of $X$. It is characterized under the assumption that every join irreducible of $X$ is an atom.

**Key words.** consensus, median, semilattice

**AMS subject classifications.** 06A12, 90A08

## 1. Introduction.

The notion of consensus appears in many different fields such as biology, economics, and sociology when the aggregation of processed data is desired. To encompass such a wide range of applications, Barthélemy and Janowitz [6] propose a broad formal model for consensus based in the theory of ordered sets. Within their model they achieve axiomatic characterizations of several types of consensus rules. We refer the reader to [6] and [7] for extensive examples and motivation for this approach.

One natural way to produce a consensus of a $k$-tuple $(x_1, \ldots, x_k)$ in a finite metric space $(X, d)$ is to find all those elements $x$ in $X$ that are "closest" to $(x_1, \ldots, x_k)$. That is, find all $x$ such that $\sum_{1 \le i \le k} d(x, x_i)$ is minimized. This is referred to as the *median procedure* and the resulting function is called the *median consensus rule.* This rule has been well studied (cf., Bandelt and Barthélemy [1], Barthélemy and McMorris [8], Leclerc [10]–[12], Monjardet [14]). The present paper improves two of the main results in [6] dealing with the median procedure, namely Theorems 19 and 28, by removing two out of the five axioms and, at the same time, weakening a third axiom. In doing this, we will give a generalization of the median procedure for nonsymmetric distance functions. An example of this situation is provided by weighting the standard minimum-moves metric on the covering graph of a distributive semilattice $X$. We are able to characterize this generalized median procedure under the assumption that every join irreducible of $X$ is an atom.

Our paper closely follows the terminology and notation of Barthélemy and Janowitz [6] and is organized in five sections, with the first section being this introduction. In order to keep this exposition relatively self-contained, we include some preliminaries in §2. In §3, we consider consensus rules on finite partially ordered sets as well as a generalization of the $t$-condorcet axiom from [6]. In §4, we consider consensus rules on finite semilattices, with §5 covering finite distributive semilattices. The characterization theorems mentioned above are contained in this last section.

## 2. Some preliminaries and the axioms.

We start with some definitions in the most general setting and assume here that $X$ is just a finite set. A *consensus*

* Received by the editors October 13, 1993; accepted for publication (in revised form) August 10, 1994.

† Department of Mathematics, University of Louisville, Louisville, Kentucky 40292. The research of the first author was supported by Office of Naval Research grant N00014-89-J-1643.

*rule* on $X$ is a map $c: X^* \to \mathbf{P}(X) \setminus \{\varnothing\}$, where $\mathbf{P}(X)$ is the power set of $X$, and $X^* = \bigcup_{k>0} X^k$ with $X^k$ as the $k$-fold product of $X$. An element $x^* = (x_1, \ldots, x_k)$ of $X^*$ is called a *profile,* and we let $(x)_k$ denote the *constant profile* $(x)_k = (x_1, \ldots, x_k)$, with $x_1 = \ldots = x_k = x$. Set $V_k = \{1, \ldots, k\}$.

Now consider an arbitrary function $d : X \times X \to [0, \infty)$. For $x^* \in X^*$, set $D(x^*, x) = \sum_{1 \le i \le k} d(x_i, x)$ and $D(x, x^*) = \sum_{1 \le i \le k} d(x, x_i)$. Define the consensus rules $m_L$ and $m_R$ on $X$ as follows:

$$m_R(x^*) = \{x \in X : D(x^*, x) \text{ is minimum}\}$$

and

$$m_L(x^*) = \{x \in X : D(x, x^*) \text{ is minimum}\}.$$

We call $m_L$ the *left median consensus rule* and $m_R$ the *right median consensus rule.* In the case where $d$ is a metric, we have $m_L = m_R$, and we get the ordinary *median consensus rule* [6], [8]. We can think of $m_L(x^*)$ as those $x$ whose "distance" *to* the profile is as small as possible, whereas $m_R(x^*)$ contains those $x$ whose "distance" *from* the profile is as small as possible. Clearly $m_L(x^*)$ need not equal $m_R(x^*)$, but there is the obvious left–right duality, so that general results for $m_L$ will also hold for $m_R$. For this reason, we will focus on the left median rule in §5.

A consensus rule $c$ on $X$ is *consistent* [15] if an only if for all profiles $x^*$ and $y^*$, the condition $c(x^*) \cap c(y^*) \neq \varnothing$ implies that $c(x^*y^*) = c(x^*) \cap c(y^*)$, where $x^*y^*$ denotes the concatenation of the two profiles $x^*$ and $y^*$. $c$ satisfies *unanimity* if $c((x)_k) = \{x\}$ for all constant profiles $(x)_k$ in $X^*$. Clearly, if $d$ satisfies $d(x, y) = 0$ if and only if $x = y$ for all $x, y \in X$, then $m_L$ and $m_R$ satisfy unanimity. The proof of the next result is the same as that given for Lemma 7 in [6].

LEMMA 1. *The left and right median consensus rules are consistent.*

## 3. Posets.

We now start to add some structure to the set $X$ by letting $X$ be a finite partially ordered set (poset), i.e., $X$ is equipped with a reflexive, antisymmetric, transitive relation $\le$ . Recall that the *supremum,* or *join,* of a subset $A$ of $X$ is denoted by $\sup(A)$ when it exists. Dually, the *infimum,* or *meet,* of $A$ is denoted by $\inf(A)$. The poset $X$ is a *lattice* if and only if $\sup\{x, y\}$ and $\inf\{x, y\}$ exist for all $x, y$ in $X$. As is standard practice, set $x \wedge y = \inf\{x, y\}$ and $x \vee y = \sup\{x, y\}$ [9], [15].

An element $s$ in $X$ is *join irreducible* if $s = x \vee y$ implies that either $s = x$ or $s = y$. Dually, an element $m$ is *meet irreducible* if $m = x \wedge y$ implies that either $m = x$ or $m = y$. We say that two elements $x$ and $y$ in $X$ are *join compatible* if $x \vee y$ exists and *meet compatible* if $x \wedge y$ exists.

### 3.1. Condorcet axioms for posets.

A condorcet axiom (named after the Marquis de Condorcet, an 18th century Frenchman who is considered one of the founding fathers of modern voting theory) for hierarchical classifications was first introduced by Barthélemy and McMorris in [8] and then generalized to semilattices by Barthélemy and Janowitz in [6]. We now further extend this axiom to arbitrary posets. To accomplish this, we need the notions of index and dual index.

The *index* of the element $s \in X$ in the profile $x^* \in X^k$ is

$$\gamma(s, x^*) = |\{i \in V_k : s \le x_i\}|/k,$$

whereas the *dual index* of $s$ is

$$\gamma'(s, x^*) = |\{i \in V_k : s \ge x_i\}|/k.$$

Let $S$ be a nonempty subset of $X, t \in [0,1]$ be rational, and $c$ be a consensus rule on $X$.

(1) $c$ is *upward $t$-condorcet with respect to* $(S, \sup)$ if and only if, for each $s \in S$ and $x^* \in X^*$ such that $\gamma(s, x^*) = t$, $s$ join compatible with $x \in c(x^*)$ implies that $x \vee s \in c(x^*)$.

(2) $c$ is *upward $t$-condorcet with respect to* $(S, \inf)$ if and only if, for each $s \in S$ and $x^* \in X^*$ such that $\gamma'(s, x^*) = t, x \wedge s \in c(x^*)$ implies that $x \in c(x^*)$.

(3) $c$ is *downward $t$-condorcet with respect to* $(S, \sup)$ if and only if, for each $s \in S$ and $x^* \in X^*$ such that $\gamma(s, x^*) = t, x \vee s \in c(x^*)$ implies that $x \in c(x^*)$.

(4) $c$ is *downward $t$-condorcet with respect to* $(S, \inf)$ if and only if, for each $s \in S$ and $x^* \in X^*$ such that $\gamma'(s, x^*) = t$, $s$ meet compatible with $x \in c(x^*)$ implies that $x \wedge s \in c(x^*)$.

(5) To agree with the definition in [6], we say that $c$ is *$t$-condorcet* if $c$ is both upward and downward $t$-condorcet with respect to $(S, \sup)$, where $S$ is the set of all join irreducibles of $X$.

**3.2. Consequences of the axioms.** Since consistency, unanimity, and condorcet are reasonable conditions on a consensus rule, we now see how these affect the output in the most general poset case.

THEOREM 1. *Let $t \in (0,1)$ be rational and let $c$ be a consensus rule on $X$ that satisfies unanimity and consistency and is upward $t$-condorcet with respect to $(S, \sup)$. For any profile $x^* \in X^*$ and $s \in S$, if $\gamma(s, x^*) > t$ and $x \in c(x^*)$ is join compatible with $s$, then $s \leq x$.*

*Proof.* Let $x^* \in X^k$. If $k = 1$, then the result follows from the unanimity condition, so assume that $k \geq 2$ and let $t = m/n$. Suppose that there exist $x \in c(x^*)$ and $s \in S$ such that $\gamma(s, x^*) > t, x \vee s$ exists, and $s$ is not less than or equal to $x$.

First assume $\gamma(s, x^*) = 1$, so that $s \leq x_i$ for all $i = 1, \ldots, k$. Let $y^* = (x^*)^m (x)^{k(n-m)} \in X^{kn}$. It follows from consistency that $x \in c((x^*)^m)$ and from consistency and unanimity that $c((x)^{k(n-m)}) = \{x\}$. Hence by consistency $c(y^*) = \{x\}$. Note that $\gamma(s, y^*) = km/kn = m/n = t$, and so by upward $t$-condorcet $x \vee s \in c(y^*)$, contrary to $c(y^*) = \{x\}$.

Now assume $\gamma(s, x^*) = u/k > m/n$ where $0 < u < k$. Let $y^* = (x^*)^m (x)^{nu-mk} \in X^{nu}$. As before, we can show that $c(y^*) = \{x\}$. Note that $\gamma(s, y^*) = um/nu = m/n = t$, and so by upward $t$-condorcet, $x \vee s \in c(y^*)$, contrary to $c(y^*) = \{x\}$. $\square$

THEOREM 2. *Let $t \in (0,1)$ be rational and $c$ be a consensus rule on $X$ that satisfies unanimity and consistency and is downward $t$-condorcet with respect to $(S, \sup)$. For any profile $x^* \in X^*$ and $s \in S$, if $x = y \vee s \in c(x^*)$ for some $y \neq x$, then $\gamma(s, x^*) \geq t$.*

*Proof.* Let $x^* \in X^k$. If $k = 1$, then the result follows from the unanimity condition, so assume that $k \geq 2$ and let $t = m/n$. Assume that there exists $x \in c(x^*)$ and $s \in S$ such that $x = y \vee s$, with $y \neq x$ and $\gamma(s, x^*) < t$.

First, suppose $\gamma(s, x^*) = 0$ and let $y^* = (x^*)^{(n-m)} (x)^{km} \in X^{kn}$. Then, as in the proof of Theorem 1, $c(y^*) = \{x\}$. Note that $\gamma(s, y^*) = km/kn = m/n = t$. Since $x = y \vee s$, with $y \neq x$, it follows from downward $t$-condorcet that $y \in c(y^*)$, which contradicts $c(y^*) = \{x\}$.

Now assume $\gamma(s, x^*) = u/k < m/n$, where $0 < u < k$. Letting $y^* = (x^*)^{(n-m)} (x)^{(km-nu)} \in X^{(kn-nu)}$, we get, as above, $c(y^*) = \{x\}$. Note that $\gamma(s, y^*) = [(n-m)u + mk - nu]/(nk - nu) = [m(k-u)]/n(k-u) = m/n = t$. Since $x \notin S$, and $x = y \vee s$, with $y \neq x$, it follows from downward $t$-condorcet that $y \in c(y^*)$ contrary to $c(y^*) = \{x\}$. $\square$

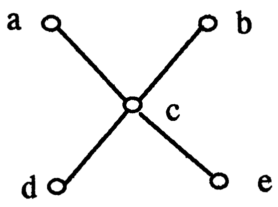


FIG. 1.

We note that the preceding two theorems, as well as Theorem 3 in §5, can be proved using weaker hypotheses. For example, the unanimity axiom can be replaced by an axiom that requires a consensus rule $c$ on $X$ to satisfy $c((x)) = \{x\}$ for all profiles $(x) \in X^1$, while the consistency axiom can be replaced by the following two axioms:

(i) $c((x^*)^r) = c(x^*)$ for all profiles $x^*$ in $X^*$ and positive integers $r$. (This is called *weak consistency* in Barthélemy and Janowitz [6].)

(ii) For all profiles $x^*, y^*$ where either $x^*$ or $y^*$ is a constant profile, if $c(x^*) \cap c(y^*) \neq \varnothing$, then $c(x^*y^*) = c(x^*) \cap c(y^*)$.

We finish this section by giving an example illustrating some of the above concepts. First, let $X$ be a poset and let $x, y \in X$. Then $y$ *covers* $x$ if $x \leq y$ and $x \leq z < y$ implies that $x = z$. The *covering graph* of $X$ is a graph $G$ with vertex set $X$ such that $xy$ is an edge if and only if either $y$ covers $x$ or $x$ covers $y$. The *lattice metric* $\partial$ on $X$ is the minimum-moves metric on $G$ (see Barthélemy, Leclerc and Monjardet [7]).

Now let $X$ be the poset depicted in Fig. 1. The set of join irreducibles of $X$ is $S = \{a, b, d, e\}$. It is straightforward to verify that the median consensus rule on $X$ (with respect to the lattice metric) satisfies unanimity and consistency and is upward 1/2-condorcet with respect to $(S, \sup)$.

In the next section, however, we will show that there are posets (namely, finite join semilattices that are not upper semimodular) where the median consensus rule is not upward 1/2-condorcet. In fact, with enough structure on the poset $X$, we will be able to characterize when the median consensus rule is upward 1/2-condorcet.

## 4. Semilattices.

We now impose additional structure on the poset $X$. Recall that $X$ is a *join* (*meet*) *semilattice* if $x \vee y$ ($x \wedge y$) exists for all $x, y \in X$. Let $\sup(\varnothing) = \inf(X)$ and $\inf(\varnothing) = \sup(X)$. Note that if $X$ is a join semilattice but not a lattice, then $\inf(\varnothing)$ exists but $\sup(\varnothing)$ does not exist. The reverse situation occurs when $X$ is a meet semilattice but not a lattice. The semilattice $X$ is *graded* if there exists an integer-valued function $g$ on $X$ such that $y$ covers $x$ implies that $g(y) = g(x) + 1$. If $X$ has a least element 0 and $g(0) = 0$, then $g$ is called the *height function* of $X$.

### 4.1. Join semilattices.

Let $X$ be a finite join semilattice and $S$ the set of all join irreducibles of $X$. For any real number $a$, let $[a]^*$ denote the least integer strictly greater than $a$. Define $m^*$ by $m^*(x^*) = \sup\{s \in S : k\gamma(s, x^*) \geq [k/2]^*\} = \sup\{s \in S : \gamma(s, x^*) > 1/2\}$.

If $X$ is not a lattice, then $\sup(\varnothing)$ does not exist, and thus $m^*(x^*)$ does not exist for all those profiles $x^*$ such that $\{s \in S : k\gamma(s, x^*) \geq [k/2]^*\} = \varnothing$. However, if $X$ is a lattice, then $m^*$ is a function from $X^*$ to $X$. In this case, there is an equivalent expression for $m^*$ from Proposition 4.1 of [10]:

$$m^*(x^*) = \sup\{\inf_{i \in W} x_i : W \text{ is a subset of } \{1, \ldots, k\} \text{ such that } |W| \geq [k/2]^*\}.$$

This consensus rule on $X$ is called the *majority consensus rule*.

COROLLARY 1. *Let $c$ be a consensus rule on a finite join semilattice $X$ that satisfies unanimity and consistency and is upward $1/2$-condorcet with respect to $(S, \sup)$. Then, for any profile $x^* \in X^*$ such that $m^*(x^*)$ exists, we have $m^*(x^*) \leq y$ for all $y \in c(x^*)$.*

*Proof.* This result follows immediately from Theorem 1. □

Recall that $X$ is *upper semimodular* if, for every $x, y \in X$ such that $x \wedge y$ exists, if $x$ covers $x \wedge y$ and $y$ covers $x \wedge y$, then $x \vee y$ covers $x$ and $x \vee y$ covers $y$. For the next result, we assume that $X$ is a graded join semilattice and that $m$ is the median consensus rule on $X$ with respect to the lattice metric $\partial$.

COROLLARY 2. *The median consensus rule $m$ on a finite join semilattice $X$ is upward $1/2$-condorcet with respect to $(S, \sup)$ if and only if $X$ is upper semimodular.*

*Proof.* Suppose $m$ is upward $1/2$-condorcet with respect to $(S, \sup)$. Then, since $m$ satisfies both consistency and unanimity, it follows from Corollary 1 that, for any profile $x^* \in X^*$ such that $m^*(x^*)$ exists, we have $m^*(x^*) \leq y$ for all $y \in m(x^*)$. It follows from Theorem 4.4 in Leclerc [11] that the height function $g$ is an upper valuation, which in turn is equivalent to $X$ being upper semimodular (see statements (6) and (7) in Leclerc [11]).

Now suppose $X$ is upper semimodular. Then, as mentioned above, the height function $g$ is an upper valuation. To show that $m$ is upward $1/2$-condorcet with respect to $(S, \sup)$, let $x^* \in X^*, x \in m(x^*)$, and $s \in S$ be such that $\gamma(s, x^*) = 1/2$ and $x \vee s > x$. Now apply Proposition 4.1 in [11] to observe that $x \vee s \in m(x^*)$. Hence, $m$ is upward $1/2$-condorcet with respect to $(S, \sup)$. □

We note that in [11], the domain of a consensus rule is $X^k$ and not $X^*$. Since these two notions agree when a consensus rule is restricted to profiles of fixed length, there is no problem in applying the results found in [11] to prove Corollary 2.

The problem of characterizing the median consensus rule for finite semimodular lattices was also posed in Leclerc [10]. It follows from Corollary 2 that the median consensus rule on a finite (upper) semimodular lattice satisfies unanimity and consistency and is upward $1/2$-condorcet. However, the converse is not true, as can be seen by letting $X$ be a finite lattice and $c : X^* \to \mathbf{P}(X) \setminus \{\varnothing\}$ be the *dual unanimity function* defined by $c(x^*) = c(x_1, \ldots, x_k) = \{\sup\{x_1, \ldots, x_k\}\}$. The complete solution to this problem is still an interesting open problem.

**4.2. Meet semilattices.** The previous section can be easily dualized. For example, the *dual majority consensus rule* $(m^*)'$ on a finite lattice $X$ is given by

$$(m^*)'(x^*) = \inf\{\sup_{i \in W} x_i : W \text{ is a subset of } \{1, \ldots, k\} \text{ such that } |W| \geq [k/2]^*\}.$$

If $S'$ is the set of all meet-irreducible elements of $X$, then

$$(m^*)'(x^*) = \inf\{m \in S' \colon k\gamma'(m, x^*) \geq [k/2]^*\}.$$

For the case when $X$ is a meet semilattice but not a lattice, so that $\inf(\varnothing)$ does not exist, $(m^*)'(x^*)$ need not always exist.

For Corollary 3, we assume that $X$ is a finite meet semilattice and $S'$ is the set of all meet irreducible elements of $X$. For Corollary 4, we also assume that $X$ is graded and that $m$ is the median consensus rule on $X$ with respect to the lattice metric.

COROLLARY 3. *Let $c$ be a consensus rule on $X$ that satisfies unanimity and consistency and is downward $1/2$-condorcet with respect to $(S', \inf)$. Then, for any profile $x^* \in X^*$ such that $(m^*)'(x^*)$ exists, we have $(m^*)'(x^*) \geq y$ for all $y \in c(x^*)$.*

COROLLARY 4. *The median consensus rule $m$ is downward $1/2$-condorcet with respect to $(S', \inf)$ if and only if $X$ is lower semimodular.*

A finite lattice $X$ is *modular* if it is both upper and lower semimodular. For the next result, we assume that $X$ is a finite lattice.

COROLLARY 5. *The median consensus rule $m$ is upward $1/2$-condorcet with respect to $(S, \sup)$ and is downward $1/2$-condorcet with respect to $(S', \inf)$ if and only if $X$ is modular.*

## 4.3. Quota numbers and the $m_t$ consensus rule.

Let $t \in [0, 1]$ and suppose that $X$ is a meet semilattice and $S$ is the set of all join irreducibles having the property that for each $x^* \in X^*, \alpha_t(x^*) = \sup\{s \in S : \gamma(s, x^*) > t\}$ exists. Define the consensus rule $m_t$ on $X$ by $m_t(x^*) = \{\alpha_t(x^*)\} \cup \{\alpha_t(x^*) \vee y : y = s_1 \vee \cdots \vee s_m$, with $\gamma(s_i, x^*) = t$ and $y$ join compatible with $\alpha_t(x^*)\}$. We can now contrast Corollary 5 with Proposition 20 in [6], which states that for a lattice $X, m_{1/2}$ is consistent if and only if $X$ is distributive.

The *quota number* $q(X)$ of the semilattice $X$ is the infimum of the real numbers $t \in [0, 1]$ such that $\alpha_t(x^*)$ exists in $X$ for each $x^* \in X^*$. It is clear that the quota number always exists and that $\alpha_t(x^*)$ exists in $X$ for each $x^* \in X^*$ if and only if $q(X) \leq t \leq 1$.

In [6], it is shown that for a semilattice $X$, either $q(X) = 0$ and $X$ is a lattice or else $q(X) \geq 1/2$. For further information on quota numbers, we refer the reader to Bandelt and Meletiou [5] and Bandelt, Janowitz, and Meletiou [4]. The notion of the quota number is crucial to the characterization of $m_t$, but first we must add still more structure on the poset $X$.

## 5. Distributive meet semilattices.

A *distributive meet semilattice* $X$ is a meet semilattice such that every principal ideal is a distributive lattice. Bandelt, Janowitz, and Meletiou [4] show that if $X$ is a distributive meet semilattice, then $q(X)$ is rational. In the following, $S$ is, as usual, the set of all join irreducibles of the finite distributive meet semilattice $X$. Thus, since $X$ is finite, we also have that $q(X) < 1$.

### 5.1. A characterization of the $m_t$ consensus rule.

We now give an example to show that a characterization of the $m_t$ rule in terms of the axioms of unanimity and $t$-condorcet must be restricted to values of $t$ in (0, 1).

*Example* 1. Let $X = \{0, a, b, 1\}$ be the four-element Boolean algebra: $a$ and $b$ are the noncomparable join irreducibles, 0 is the least element, and 1 is the greatest element. Let $c$ be a consensus rule on $X$. We assert that if $c$ satisfies the unanimity condition, then $c$ is neither 0-condorcet nor 1-condorcet.

*Proof.* Assume that $c$ is 0-condorcet. Since $\gamma(b, (a)) = 0, a \in c((a))$ if and only if $a \vee b = 1 \in c((a))$, so that $c$ does not satisfy unanimity. Now assume $c$ is 1-condorcet. Since $\gamma(a, (a)) = 1, 0 \in c((a))$ if and only if $0 \vee a = a \in c((a))$, and thus $c$ does not satisfy unanimity. $\square$

The next result is an improvement of Theorem 26 in Barthélemy and Janowitz [6]. We note that in their theorem, $t$ is implicitly assumed to be rational and, given Example 1 above, $t$ needs to be in the open interval (0, 1).

THEOREM 3. *Let $X$ be a distributive meet semilattice, $t \in [q(X), 1)$ be rational, with $t \neq 0$, and $c$ be a consensus rule on $X$. Then $c = m_t$ if and only if $c$ satisfies unanimity and consistency and is $t$-condorcet.*

*Proof.* If $c = m_t$, then $c$ satisfies the three conditions from Lemma 25 and Theorem 26 in Barthélemy and Janowitz [6].

Because of $t$-condorcet and the fact that every element is the join of join irreducibles, we need only to show that for each $s \in S, x^* \in X^*$, and $x \in c(x^*)$:

(1) If $\gamma(s, x^*) > t$, then $s \leq x$.

(2) If $\gamma(s, x^*) < t$, then $s \leq x$ is not true.

Condition (1) follows directly from Theorem 1. For condition 2, assume $s \leq x$ and let $x = \sup\{t_i : t_i$ is a join irreducible and $t_i \leq t_j$ if and only if $i = j\}$. If $x = t_j = 0 \vee t_j$, then, since $t_j \neq 0$, it follows from Theorem 2 that $\gamma(t_j, x^*) \geq t$. Since $s \leq t_j$, we have that $\gamma(s, x^*) \geq t$. If $x$ is not join irreducible, then by distributivity $s \leq t_i$ for some $i$. Now note that $x = y \vee t_i$ for some $y \neq x$, so by Theorem 2 it follows that $\gamma(t_i, x^*) \geq t$, and again $\gamma(s, x^*) \geq t$. □

A distributive meet semilattice $X$ whose quota number is either 0 or 1/2 is a *median semilattice.* If $X$ is a median semilattice, then the median consensus rule $m$ on $X$ is equal to $m_{1/2}$. We will now apply Theorem 3 to this special case to obtain one of our main results.

THEOREM 4. *Let $c$ be a consensus rule on the median semilattice $X$. Then $c$ is the median consensus rule with respect to the lattice metric if and only if $c$ is consistent and* 1/2*-condorcet and satisfies unanimity.*

Theorem 4 improves Theorem 19 in [6]. Indeed, we have shown that the axioms of symmetry and stability are not necessary and that axioms of unanimity and consistency can also be weakened (see the end of §3).

One of the interests in median semilattices lies in the fact that many posets arising in applications in classification theory are so structured. For example, the set of all $n$-trees of a finite set (a type of hierarchical classification) forms a median semilattice under set inclusion. (An *$n$-tree* on $S$ is a subset $T$ of $\mathbf{P}(S)$ such that $\{x\} \in T$ for all $x \in S, \varnothing \notin T, S \in T$, and $A \cap B \in \{\varnothing, A, B\}$ for all $A, B \in T$.) We state the next result as an improvement to the main theorem in Barthélemy and McMorris [8].

COROLLARY 6. *The median procedure for $n$-trees with respect to the symmetric difference metric is the unique consensus rule that satisfies unanimity and consistency and is* 1/2*-condorcet.*

The axioms of unanimity, consistency, and 1/2-condorcet do not characterize the median procedure with respect to the lattice metric for a nonmedian distributive meet semilattice, as Example 2 in the following section will show. The problem in this general setting is that $m$ and $m_{1/2}$ are not equal. In fact, $m \neq m_t$ for any $t \in [0, 1]$. It is possible, however, to view the $m_t$ consensus rule as a median procedure in its own right with respect to a weighted lattice metric.

**5.2. The $m_t$ consensus rule as a left median.** Let $X$ be a distributive meet semilattice, so that the lattice metric on the covering graph of $X$ is given by

$$\partial(x, y) = h(x) + h(y) - 2h(x \wedge y),$$

where $h$ is the height function. Now suppose $t = m/n \in (0, 1)$ and weight the above metric as follows:

$$\partial_t(x, y) = mh(x) + (n - m)h(y) - nh(x \wedge y).$$

If $m = 1$ and $n = 2$, so that $t = 1/2$, we see that $\partial_t = \partial$. If $t \neq 1/2$, then $\partial_t$ is not symmetric and is thus not a metric.

Recall that an element of $X$ is an *atom* if it covers the least element 0. Let $S$ be the set of all join irreducibles.

LEMMA 2. *Let $X$ be a distributive meet semilattice such that every join irreducible of $X$ is an atom. Let $x^* = (x_1, \dots, x_k) \in X^*, x \in X$, and $s \in S$ be such that $s$ is not less than or equal to $x$. If $x$ is join compatible with $s$, then $D(x \vee s, x^*) = D(x, x^*) + k(m - n\gamma(s, x^*))$, where $D(x, x^*) = \sum_{i=1}^{k} \partial_t(x, x_i)$.*

*Proof.* Since $s$ is a join irreducible and is also an atom, it follows from distributivity that $x \vee s$ covers $x$. Thus $h(x \vee s) = h(x) + 1$. From distributivity we get that $(x \vee s) \wedge x_i = (x \wedge x_i) \vee (s \wedge x_i)$ for $i = 1, \dots, k$. Since $s$ is an atom, either $s \wedge x_i = s$ or $s \wedge x_i = 0$. The first case occurs if and only if $s \leq x_i$. If $s \leq x_i$, then $h((x \vee s) \wedge x_i) = h(x \wedge x_i) + 1$. Otherwise $h((x \vee s) \wedge x_i) = h(x \wedge x_i)$. Therefore,

$$\begin{aligned}\partial_t(x \vee s, x_i) &= mh(x \vee s) + (n - m)h(x_i) - nh((x \vee s) \wedge x_i)\\ &= \partial_t(x, x_i) + (m - n) \quad \text{if } s \leq x_i, \text{ or}\\ &= \partial_t(x, x_i) + m \quad \text{if } s \leq x_i \text{ fails.}\end{aligned}$$

Thus we have $D(x \vee s, x^*) = \sum_{i=1}^{k} \partial_t(x \vee s, x_i) = \sum_{i=1}^{k} \partial_t(x, x_i) + k(m - n\gamma(s, x^*)) = D(x, x^*) + k(m - n\gamma(s, x^*))$. □

THEOREM 5. *Let $X$ be a distributive meet semilattice such that every join irreducible of $X$ is an atom. If $t \in [q(X), 1)$ is rational and $t \neq 0$, then the consensus rule $m_t$ on $X$ is the left median consensus rule with respect to the function $\partial_t$.*

*Proof.* Let $m_L$ denote the left median consensus rule with respect to $\partial_t$. By Lemma 1, $m_L$ is consistent. Since $\partial_t(x, y) = 0$ if and only if $x = y$, it follows that $m_L$ satisfies unanimity. Finally, using Lemma 2, it is easy to verify that $m_L$ is $t$-condorcet. Hence, by Theorem 3, we get $m_L = m_t$, and we are done. □

We now give an important example to show that the restriction $t \in [q(X), 1)$ is crucial in Theorem 5.

*Example* 2. Let $Y$ be a finite set with $n$ elements. A *weak hierarchy* (Bandelt and Dress [2]) on $Y$ is a subset $H$ of $\mathbf{P}(Y)$ such that $A_1 \cap A_2 \cap A_3 \in \{A_1 \cap A_2, A_1 \cap A_3, A_2 \cap A_3\}$ for all $A_1, A_2, A_3 \in H$. In addition, we require $\{y\} \in H$ for all $y \in Y, \varnothing \notin H$, and $Y \in H$, so that weak hierarchies are generalizations of $n$-trees. If $H_1$ and $H_2$ are two weak hierarchies on $Y$, then their intersection is also a weak hierarchy on $Y$. Clearly, if $W$ denotes the set of all weak hierarchies on $Y$, then $W$ is a meet semilattice that is partially ordered under set inclusion. Notice that the join irreducibles of $W$ are those weak hierarchies $H$ that contain one and only one *nontrivial cluster* $A$, i.e., $1 < |A| < |Y|$. Moreover, $W$ is a distributive meet semilattice with $q(W) = 2/3$ (Bandelt, Janowitz, and Meletiou [4]).

Define a consensus rule $c$ on the distributive meet semilattice $W$ as follows:

$$c(x^*) = \{H \in W : D'(H, x^*) \text{ is maximum}\}$$

where $D' : W \times W^* \to R$ is given by

$$D'(H, x^*) = \sum_{A \in H, 1 < |A| < n} |A|(2k\gamma(A, x^*) - k).$$

Here, $k$ is the length of the profile $x^*$ and the nontrivial cluster $A$ is identified with the join irreducible in $W$ that contains $A$.

For this example we let $Y = \{y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8\}$ and $t = 1/2$. We will show that $c$ satisfies unanimity and consistency and is 1/2-condorcet but is not the left median consensus rule.

CLAIM 1. *$c$ satisfies unanimity.*

*Proof.* Let $x^* = (H, \ldots, H) \in W^k$ and note that for each $A \in H$ we have $\gamma(A, x^*) = 1$, so that $D'(H, x^*) = \sum_{A \in H, 1<|A|<n} k|A|$. Observe that for any $A \in H, D'(H, x^*) > D'(H - \{A\}, x^*)$. Moreover, if $B \notin H$, then $\gamma(B, x^*) = 0$. If $H \cup \{B\} \in W$, then $D'(H, x^*) > D'(H \cup \{B\}, x^*)$. Therefore, $c(x^*) = \{H\}$. □

CLAIM 2. *c is* 1/2-*condorcet.*

*Proof.* Let $x^* = (H_1, \ldots, H_k) \in W^k$ and suppose $H \cup \{A\} \in W$, where $H \in W$ and $\gamma(A, x^*) = 1/2$. Now $D'(H, x^*) = D'(H \cup \{A\}, x^*)$, so that $H \in c(x^*)$ if and only if $H \cup \{A\} \in c(x^*)$. □

CLAIM 3. *c is consistent.*

*Proof.* Let $x^* \in W^{k_1}, y^* \in W^{k_2}$, and $A$ be a subset of $Y$. Suppose $\gamma(A, x^*) = u_1/k_1$ and $\gamma(A, y^*) = u_2/k_2$. Then $\gamma(A, x^*y^*) = (u_1+u_2)/(k_1+k_2)$, and so $|A|(2k_1\gamma(A, x^*) - k_1) + |A|(2k_2\gamma(A, y^*) - k_2) = |A|[2(k_1 + k_2)\gamma(A, x^*y^*) - (k_1 + k_2)]$. It follows that $D'(H, x^*) + D'(H, y^*) = D'(H, x^*y^*)$ for all $H \in W$.

Now suppose $c(x^*) \cap c(y^*) \neq \varnothing$. If $H \in c(x^*) \cap c(y^*)$, then $D'(H, x^*)$ and $D'(H, y^*)$ are both maximal. Therefore, $D'(H, x^*) + D'(H, y^*)$ is maximal and hence $H \in c(x^*y^*)$. Conversely, if $H' \notin c(x^*) \cap c(y^*)$, then $D'(H', x^*) + D'(H', y^*) < D'(H, x^*) + D'(H, y^*)$. Therefore, $H' \notin c(x^*y^*)$. □

CLAIM 4. *c is not the left median consensus rule.*

*Proof.* Since $t = 1/2$, it follows that $m_L = m_R = m$, so we need only show that $c \neq m$. Let $x^* = (H_1, H_2, H_3)$, where $H_1$ has $\{y_1, y_2, y_3, y_4, y_5, y_6\}, \{y_1, y_7\}$, and $\{y_1, y_7, y_8\}$ as its nontrivial clusters, where $H_2$ has $\{y_1, y_2, y_3, y_4, y_5, y_6\}, \{y_6, y_7\}$, and $\{y_6, y_7, y_8\}$ as its nontrivial clusters, and where $H_3$ has $\{y_1, y_7\}, \{y_1, y_7, y_8\}$, $\{y_6, y_7\}$, and $\{y_6, y_7, y_8\}$ as its nontrivial clusters. Then $m(x^*) = \{H_3\}$, while $c(x^*) = \{H_1, H_2\}$. □

**5.3. Conclusions.** In this paper, we looked at the axioms of unanimity, consistency, and $t$-condorcet for consensus rules defined on finite partially ordered sets. If the poset has the structure of a median semilattice and $t = 1/2$, then these axioms characterize the median consensus rule. On a nonmedian, distributive meet semilattice whose quota number is less than or equal to $t$ (so $t > 1/2$), these axioms characterize the $m_t$ consensus rule. If, in addition, each join irreducible is an atom, then $m_t$ is the left median consensus rule with respect to the function $\partial_t$. A natural example of this situation is the rule $m_{2/3}$ on the semilattice $W$ [13]. If the quota number is greater than $t$, then these axioms, in general, do not determine a unique consensus rule.

**Note added in proof.** Henry Martyn Mulder has found an example showing that the $t$-condorcet condition needs to be modified for Theorems 19 and 26 of Barthélemy and Janowitz [6] to be true. However, we can show that everything is valid under the assumption that all join irreducibles are atoms. Thus this condition should be imposed on the hypotheses of our Theorems 3 and 4. All other results stand as stated.

## REFERENCES

[1] H.-J. BANDELT AND J. P. BARTHÉLEMY, *Medians in median graphs*, Discrete Appl. Math., 8 (1984), pp. 131–142.

[2] H.-J. BANDELT AND A. DRESS, *Weak hierarchies associated with similarity measures: An additive clustering technique*, Bull. Math. Biol., 51 (1989), pp. 133–166.

[3] H.-J. BANDELT AND J. HEDLIKOVA, *Median algebras*, Discrete Math., 45 (1983), pp. 1–30.

[4] H.-J. BANDELT, M. F. JANOWITZ, AND G. C. MELETIOU, *n-median semilattices*, Order, 8 (1991), pp. 185–195.

[5] H.-J. BANDELT AND G. C. MELETIOU, *An algebraic setting for near-unanimity consensus*, Order, 7 (1990), pp. 169–178.

[6] J. P. BARTHÉLEMY AND M. F. JANOWITZ, *A formal theory of consensus*, SIAM J. Discrete Math., 4 (1991), pp. 305–322.

[7] J. P. BARTHÉLEMY, B. LECLERC, AND B. MONJARDET, *On the use of ordered sets in problems of comparison and consensus of classifications*, J. Classification, 3 (1986), pp. 187–224.

[8] J. P. BARTHÉLEMY AND F. R. MCMORRIS, *The median procedure for n-trees*, J. Classification, 3 (1986), pp. 329–334.

[9] G. BIRKHOFF, *Lattice Theory*, 3rd ed., American Mathematical Society, Providence, RI, 1967.

[10] B. LECLERC, *Medians and majorities in semimodular lattices*, SIAM J. Discrete Math., 3 (1990), pp. 266–276.

[11] ———, *Lattice valuations, medians, and majorities*, Discrete Math., 111 (1993), pp. 345–356.

[12] ———, *Medians for weight metrics in the covering graphs of semilattices*, Discrete Appl. Math., 49 (1994), pp. 281–297.

[13] F. R. MCMORRIS AND R. C. POWERS, *Consensus weak hierarchies*, Bull. Math. Biol., 53 (1991), pp. 679–684.

[14] B. MONJARDET, *Theorie et application de la mediane dans les treilles distributifs finis*, Ann. Discrete Math., 9 (1980), pp. 87–91.

[15] H. P. YOUNG AND A. LEVENGLICK, *A consistent extension of Condorcet's election principle*, SIAM J. Appl. Math., 35 (1978), pp. 285–300.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 517–525, November 1995

# SYMMETRIC MATRICES REPRESENTABLE BY WEIGHTED TREES OVER A CANCELLATIVE ABELIAN MONOID *

HANS-JÜRGEN BANDELT† AND MICHAEL ANTHONY STEEL‡

**Abstract.** The classical result that characterizes metrics induced by paths in a labeled tree having positive real edge weights is generalized to allow the edge weights to take values in any cancellative abelian monoid satisfying the additional requirement that $x + x = y + y$ implies $x = y$. This includes the case of arbitrary real-valued edge weights, which applies to distance-hereditary graphs, thus yielding (unique) weighted tree representations for the latter.

**Key words.** trees, 4-point condition, abelian monoid, distance-hereditary graph

**AMS subject classification.** 05C05

**Introduction.** Given a tree, suppose that some of its vertices are labeled by sets which form a partition of $\{1, \ldots, n\}$, while its edges are weighted by some positive real numbers. Then let $d_{ij}$ denote the sum of the weights of all the edges on the path connecting the vertices with $i$ and $j$ in their label sets. This results in a symmetric, $n \times n$ matrix $d = [d_{ij}]$, with zero diagonal, satisfying the *4-point condition,*

$$d_{ij} + d_{kl} \leq \max\{d_{ik} + d_{jl}, d_{il} + d_{jk}\}$$

for all $i, j, k, l$ from $\{1, \ldots, n\}$. The converse, that this condition guarantees tree realizability, is also true (see, for example, Buneman [4]) and constitutes a well-known result used widely in taxonomy; cf. [1], [3] for pertinent references.

Furthermore, the weighted-tree representation for $d$ is necessarily unique—provided that no redundant vertices are used, i.e., all vertices of degree less than 3 must be labeled. This mere uniqueness result of the representation also holds when arbitrary nonzero real weights are attached to the edges. This has recently been established by Hendy [6] using a novel technique based on Hadamard matrix transformations that also allows the recovery of the weighted tree, though in exponential time. Observe that a tree weighted with possibly negative reals still satisfies a relaxed 4-point condition, viz., at least two of the three distance sums are equal (and not necessarily the two larger ones). Our main result below will show that this condition characterizes tree realizability over $\mathbf{R}$. Interestingly, the distance matrix $d$ of a graph $G$ (unweighted, undirected) satisfies this relaxed 4-point condition exactly when $G$ is a *distance-hereditary* graph, see [2]. Thus, we get a canonical tree representation for such graphs without extra effort.

The proof of the main theorem is inductive and can easily be adapted to construct the unique tree realization for $d$ in polynomial time. Furthermore, our proof clarifies the respective roles played by the inequality and equality aspects of the classical 4-point condition (described above) in generating a tree representation—namely,

* Received by the editors July 8, 1991; accepted for publication (in revised form) August 30, 1994.

† Mathematisches Seminar, Universität Hamburg, Bundestraße 55, D-20146 Hamburg, Germany.

‡ Zentrum für Interdisziplinäre Forschung, Universität Bielefeld, Wellenberg 1, D-33615 Bielefeld, Germany. Current address, Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand (mas@math.canterbury.ac.nz). The research of this author was supported by the Alexander von Humboldt–Stiftung.

previous proofs have exploited the inequality part in the construction of the tree; in fact, the equality part alone guarantees the existence and uniqueness of a tree representation with real edge weights, whereas the inequality part merely constrains the resulting edge weights to be positive. Also, in the classical situation, our proof can be further simplified, because in that case a certain complication cannot arise.

In order to cover both the classical case of tree metrics as well as the preceding one, we let the edge weights come from any submonoid of an abelian group without elements of order 2. Specifically, let $\Lambda$ be an abelian monoid satisfying the following two conditions: for $\alpha, \beta, \xi, \zeta \in \Lambda$,

$$\alpha + \xi = \alpha + \zeta \text{ implies } \xi = \zeta \quad \text{(cancellation)},$$
$$\alpha + \alpha = \beta + \beta \text{ implies } \alpha = \beta \quad \text{(uniqueness of halves)}.$$

For $\alpha + \alpha$ we also write $2\alpha$, and in case $\alpha + \beta = 0$ is solvable, we write $\beta = -\alpha$. Canonical choices for $\Lambda$ are $\Lambda = \mathbf{R}, \Lambda = \mathbf{R}^+, \Lambda = \frac{1}{2}\mathbf{Z}$, or $\Lambda = \frac{1}{2}\mathbf{N}$, under addition. The latter two choices are relevant when graphs are studied. One could, of course, also let $\Lambda = \mathbf{Z}_m$ (integers modulo $m$), where $m \geq 3$ is odd.

For a tree whose vertices are labeled as before and whose edges are weighted from $\Lambda$, the induced matrix $d$ has the property that any four numbers, not necessarily distinct, chosen from $1, \ldots, n$ can be ordered as, say, $i, j, k, l$, so that

$$d_{ij} + d_{kl} + 2\xi = d_{ik} + d_{jl} = d_{il} + d_{kj} \quad \text{for some } \xi \in \Lambda.$$

We call this property the *4-point condition with respect to the monoid* $\Lambda$. In case $\Lambda = \mathbf{R}^+ \cup \{0\}$ (the nonnegative reals under addition), the 4-point condition with respect to $\Lambda$ is nothing but the classical 4-point condition, which says that two of the three distance sums are equal and at least as large as the third. In case we choose $\Lambda = \mathbf{R}$, the condition simply requires that two of the distance sums are equal.

As in the classical case, we wish to realize a matrix over $\Lambda$ satisfying the 4-point condition by a unique edge-weighted, labeled tree. We then require that $\Lambda$ is cancellative and that half-elements are unique, and, in order to guarantee uniqueness, we must also insist that the tree has no unlabeled vertices of degree less than 3 and no zero edge weights.

THEOREM 1. *Let $d$ be a symmetric, zero-diagonal, $n \times n$ matrix with entries in a cancellative abelian monoid $\Lambda$ that has uniqueness of halves. Then $d$ satisfies the 4-point condition with respect to $\Lambda$ if and only if there exists a tree $T$ that has no unlabeled vertices of degree less than 3 and that possesses a unique weighting of its edges by nonzero elements of $\Lambda$ that realizes $d$. In this case, such a tree $T$ is necessarily unique.*

Notice that both the cancellation and uniqueness of halves properties are essential hypotheses in Theorem 1. For instance, suppose that $\alpha + \xi = \alpha + \zeta$ holds in $\Lambda$ for some $\xi \neq \zeta$. Then the two trees with three edges weighted $\alpha, \alpha, \xi$ and $\alpha, \alpha, \zeta$, respectively, yield the same matrix (of size 3). Similarly if $2\alpha = 2\beta$ but $\alpha \neq \beta$, then those two trees, now having edge weights $\alpha, \alpha, \beta$ and $\beta, \beta, \alpha$, respectively, give a common matrix. Even if we are willing to allow nonuniqueness of tree representations, the cancellation property alone is not sufficient to guarantee the existence of a tree representation for a matrix $d$ satisfying the 4-point condition. The following proposition characterizes the extra condition required to guarantee the existence of tree realizations. For the sake of simplicity, we confine ourselves to abelian groups.

PROPOSITION 1. *Let $\Lambda$ be an abelian group.*

(1) *$\Lambda$ is Boolean (i.e., every nonzero element has order 2) if and only if every symmetric, zero-diagonal matrix $d$ over $\Lambda$ that satisfies the 4-point condition with respect to $\Lambda$ can be realized on every tree for which $d_{ij} = 0$ whenever $i, j$ are in the same label set.*

(2) *$\Lambda$ has no elements of order 4 if and only if every such matrix $d$ has a realization on at least one tree, with its edges weighted by elements of $\Lambda$.*

Note that in any tree realizations we consider (such as in the proofs), there is no loss of generality in assuming that all label sets are singletons, since we can restrict the domain of $d$ to ensure this, and extend the resulting tree realization to one for $d$. Furthermore, with $\Lambda$ as in Theorem 1, the extension is unique by the 4-point condition with respect to $\Lambda$.

*Proof of Theorem* 1. The "if" direction is clear. For the converse direction, we proceed by induction on the size of the matrix $d$, that is, the number of labels $n$. In the tree representations that follow, it is implicit that if an edge weight (described by some condition) is zero, then one collapses this edge, and the (possibly empty) sets of labels on the two ends of the edge are combined. For brevity, we sometimes speak of "vertex $i$" whenever that vertex is labeled by $i$.

For $n = 2$, there is nothing to show; if $d_{12} \neq 0$, one must simply assign the weight $d_{12}$ to an edge whose ends are labeled 1 and 2.

For $n = 3$, consider the bush (i.e., a tree with no interior edges) having three leaves, labeled 1, 2, 3, each adjacent to a fourth, central vertex. Assign weights $\alpha, \beta, \gamma \in \Lambda$ to the edges of this tree incident with 1, 2, 3, respectively. Then

$$d_{23} + 2\alpha = d_{11} + d_{23} + 2\alpha = d_{12} + d_{13}$$

by virtue of the 4-point condition. Since $\Lambda$ is cancellative and half-elements are unique (whenever they exist), this equation has a unique solution for $\alpha$. Similarly, we describe $\beta$ and $\gamma$ uniquely. Then, for instance,

$$d_{23} + 2\alpha + 2\beta = d_{12} + d_{13} + 2\beta = 2d_{12} + d_{23},$$

from which we get $2(\alpha+\beta) = 2\alpha+2\beta = 2d_{12}$ (by cancellation) and hence $\alpha+\beta = d_{12}$, as desired. This settles the case $n = 3$.

As for $n = 4$, consider the generic binary tree consisting of four leaves, labeled $1, \dots, 4$, with their incident edges weighted $\alpha_1, \dots, \alpha_4$, respectively, together with a fifth edge, weighted $\xi$, which connects the path between 1 and 2 with that between 3 and 4. Thus for this tree, $d_{13} + d_{24} = d_{14} + d_{23}$. We claim that the $\alpha_i$ and $\xi$ are uniquely defined. Indeed, applying the 4-point condition to 3-subsets, we obtain (cf., the case $n = 3$)

$$d_{i-1,i+1} + 2\alpha_i = d_{i-1,i} + d_{i,i+1},$$

where indices are taken modulo 4. This defines $\alpha_i$ uniquely. Moreover, the equation

$$d_{12} + d_{34} + 2\xi = d_{13} + d_{24}$$

yields a unique $\xi$ for this labeled tree. Now,

$$d_{13} + d_{24} + 2\alpha_1 + 2\alpha_2 = d_{13} + d_{12} + d_{14} + 2\alpha_2 = 2d_{12} + d_{14} + d_{23},$$

from which we obtain $\alpha_1 + \alpha_2 = d_{12}$. Similarly, we obtain $\alpha_3 + \alpha_4 = d_{34}$. In order to recover the distance $d_{14}$, we compute

$$\begin{aligned} d_{13} + d_{24} + 2\alpha_1 + 2\xi + 2\alpha_4 &= d_{12} + d_{14} + d_{13} + 2\alpha_4 + 2\xi \\ &= d_{12} + d_{34} + 2\xi + 2d_{14} \\ &= d_{13} + d_{24} + 2d_{14}, \end{aligned}$$

yielding $\alpha_1 + \xi + \alpha_4 = d_{14}$. A similar result holds for $d_{13}, d_{23}, d_{24}$. In case $\xi = 0$, we obtain a bush. Notice that, by the cancellation and uniqueness of halves properties, as long as $\xi \neq 0$, no other labeling of the tree is consistent with the given matrix $d$. This proves the case $n = 4$.

Henceforth, let $n \geq 5$. Assume that every submatrix of $d$ of size $n-1$ has a unique tree realization of the type claimed. Suppose first that for all distinct $i, j, k, l$, all three distance sums are equal. Then the lengths $\alpha_i$ and $\beta_i$ of the edges incident with leaf $i$ in the bushes connecting the triples $i, j, k$ and $i, j, l$, respectively, satisfy

$$d_{jk} + 2\alpha_i = d_{ij} + d_{ik} \quad \text{and} \quad d_{ij} + d_{il} = d_{jl} + 2\beta_i.$$

Applying the hypothesized equality of distance sums and the properties of $\Lambda$ to the sum of the preceding equations, we infer $\alpha_i = \beta_i$. This argument shows that the subbushes for all triples fit together consistently into a bush, and uniqueness of the weighted-tree representation follows as well.

So, assume that there exist two distinct sums; without loss of generality,

$$d_{14} + d_{23} = d_{12} + d_{34} \quad \text{but} \quad d_{13} + d_{24} \neq d_{12} + d_{34}.$$

Now consider the unique tree representations $T_1, T_2$, and $T_{1,2}$ of $d$ restricted to $\{2, 3, \ldots, n\}, \{1, 3, 4, \ldots, n\}$, and $\{3, 4, \ldots, n\}$, respectively. Then $T_{1,2}$ is obtained from either tree $T_1$ or $T_2$ by deleting the labels 2 and 1, respectively, and "cleaning up" the resulting label-deleted trees. For example if $i \in \{1, 2\}$ labels a vertex $v$ of degree at least three, we simply unlabel $v$. Otherwise, we delete $v$, and if $v$ had degree one, we delete its incident edge, and if the other end of this edge $v'$ has degree 2, we delete $v'$. In this last case or if $v$ had degree 2, we then identify the two edges $e_1$ and $e_2$ incident with $v'$, respectively, $v$, to give a new edge $e$. This edge is weighted by the sum of the weights of $e_1$ and $e_2$, unless this sum is zero, in which case $e$ is contracted. Note that this contraction cannot occur in the classical situation. In order to recover $T_1$ or $T_2$ from $T_{1,2}$ we mark the edge or vertex of $T_{1,2}$ where the vertex labeled 1 or 2, respectively, is attached by a branch. A parent tree $T$ for $T_1$ and $T_2$ is then obtained from $T_{12}$ by reversing both of the processes that transformed $T_1$ and $T_2$ into $T_{12}$. However, we must show that this process and the corresponding edge weighting are well defined and unique when both marks (points of attachment of the 1-branch and the 2-branch) are either

(i) distinct but located in the interior of one and the same edge of $T_{1,2}$, or

(ii) coincident and located on a vertex.

In case (i), denote the end vertices of this edge by $a$ and $b$. The vertices 3 and 4 belong to different components of $T_{1,2}$ minus the edge connecting $a$ and $b$. Otherwise, say, if $a$ is on paths from 3 and 4 to $i$ in $T_i$ for $i = 1, 2$, then either distance sum $d_{13} + d_{24}$ or $d_{14} + d_{23}$ would equal the sum of the edge weights along the paths from $a$ to 3 and $a$ to 4 in $T_{1,2}$, $a$ to 2 in $T_1$, and $a$ to 1 in $T_2$. This, however, conflicts with the hypothesis on the quartet 1, 2, 3, 4. Thus we may, without loss of generality, assume that either $a$ is 3 or $a$ lies between 3 and 1. In either case, subdivide the edge between $a$ and $b$ by two vertices $c_1$ and $c_2$, with $c_1$ being between $a$ and $c_2$, so that $c_i$ becomes the point of attachment of the $i$-branch ($i = 1, 2$). We must distinguish the four possible subcases:

$$(a \neq 3, b \neq 4), \quad (a = 3, b \neq 4), \quad (a \neq 3, b = 4), \quad \text{and} \quad (a = 3, b = 4).$$

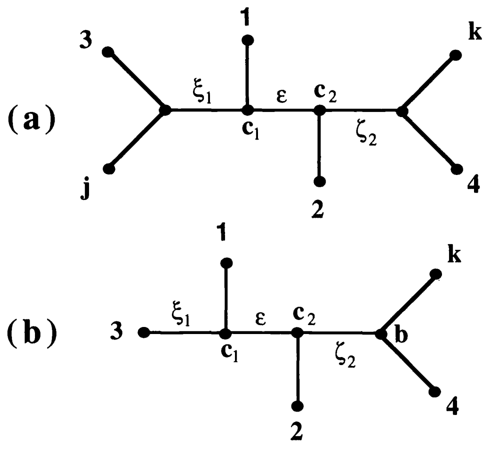


FIG. 1. *The first two subcases of case* (i) *in the proof of Theorem* 1.

For instance, the first two subcases are depicted in Fig. 1 (a) and (b), respectively. We will consider in detail only these two subcases—the treatment of the other two is similar.

Let $\varepsilon$ be the unique element of $\Lambda$ with

$$(*) \qquad d_{13} + d_{24} + 2\varepsilon = d_{14} + d_{23}.$$

Furthermore, there exist vertices labeled by $j$ ($=3$ in subcase (b), and different from 3 in subcase (a)) and $k \neq 4$ such that the paths from $j$ to 3 and $k$ to 4 hit the edge between $a$ and $b$ only in $a$ and $b$, respectively, and such that the following equalities hold:

$$(+) \qquad d_{i4} + d_{3j} + 2\xi_i = d_{i3} + d_{4j} \quad (i = 1, 2)$$

and

$$(++) \qquad d_{i3} + d_{4k} + 2\zeta_i = d_{i4} + d_{3k} \quad (i = 1, 2).$$

Now, adding up (*) and (+) for $i = 1$ yields

$$d_{24} + d_{3j} + 2(\xi_1 + \varepsilon) = d_{23} + d_{4j}.$$

Compared to equality (+) for $i = 2$, this implies

$$\xi_1 + \varepsilon = \xi_2$$

by the properties of the monoid $\Lambda$. Similarly, (++) for $i = 1$ compares to the sum of (*) and (++) for $i = 2$, thus yielding

$$\zeta_2 + \varepsilon = \zeta_1.$$

Therefore,

$$\xi_1 + \varepsilon + \zeta_2 = \xi_1 + \zeta_1 = \xi_2 + \zeta_2$$

is the weight of the edge between $a$ and $b$ (in $T_{1,2}$).

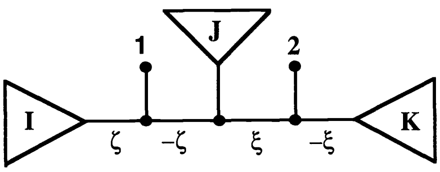


FIG. 2. *Case* (ii) *giving rise to compatible splits.*

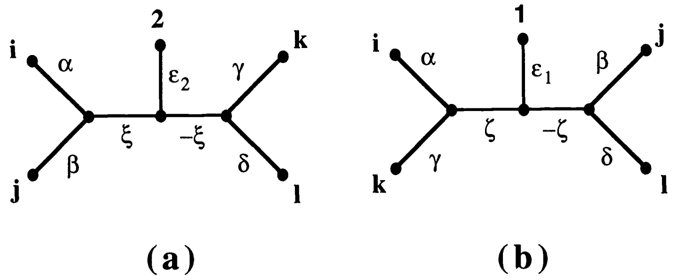


FIG. 3. *Case* (ii) *giving rise to incompatible splits.*

In case (ii) a problem may arise (though only in the nonclassical situation). Either expanded edge defines a split (i.e., a partition with two blocks) of $\{1,\ldots,n\}$. If the splits are compatible—that is, one is $\{I, J\cup K\}$ and the other is $\{I\cup J, K\}$—then we can still merge $T_1$ and $T_2$ uniquely into a single tree by the tree shown in Fig. 2, with the necessary $\pm\xi, \pm\zeta \in \Lambda$. However, the question remains of how to proceed if the splits were not compatible, that is, if for some $i,j,k,l$ in $\{3,4,\ldots,n\}$, $T_1$ and $T_2$ contained respectively the subtrees in Fig. 3 (a) and (b).

We claim that this situation cannot arise because it is in conflict with the relation between the distance sums for the quartets $\{1,2,i,j\}, \{1,2,k,l\}, \{1,2,i,l\}$, and $\{1,2,j,k\}$. The three distance sums for each of these quartets are, respectively,

$$\begin{array}{lll}
d_{12}+\alpha+\beta, & \alpha+\zeta+\beta+\xi+\varepsilon_1+\varepsilon_2, & \beta-\zeta+\alpha+\xi+\varepsilon_1+\varepsilon_2,\\
d_{12}+\gamma+\delta, & \gamma+\zeta+\delta-\xi+\varepsilon_1+\varepsilon_2, & \delta-\zeta+\gamma-\xi+\varepsilon_1+\varepsilon_2,\\
d_{12}+\alpha+\delta, & \alpha+\zeta+\delta-\xi+\varepsilon_1+\varepsilon_2, & \delta-\zeta+\alpha+\xi+\varepsilon_1+\varepsilon_2,\\
d_{12}+\beta+\gamma, & \beta-\zeta+\gamma-\xi+\varepsilon_1+\varepsilon_2, & \gamma+\zeta+\beta+\xi+\varepsilon_1+\varepsilon_2.
\end{array}$$

Equality of the last two sums in row 1 or row 2 would give $2\zeta=0$—that is, $\zeta=0$—contrary to the hypothesis. Therefore, using cancellation, we infer that $d_{12}=\eta+\varepsilon_1+\varepsilon_2$, with

$$\eta\in\{\xi+\zeta,\xi-\zeta\}\cap\{-\xi+\zeta,-\xi-\zeta\}.$$

Since $\xi=0$ or $\zeta=0$ is impossible, we obtain either $\xi=\zeta$ or $\xi=-\zeta$, whence $d_{12}=\varepsilon_1+\varepsilon_2$ in either case, and thus any equality in row 3 gives $\xi=\zeta$. From the fourth row, we deduce, however, that $\zeta=-\xi$. This final contradiction completes the argument.

We conclude that $T_1$ and $T_2$ can indeed be combined to a unique tree $T$, in which all distances except possibly $d_{12}$ are correctly represented. As for $d_{12}$, recall that $d_{14}+d_{23}=d_{12}+d_{34}$. Since $d_{14}, d_{23}$, and $d_{34}$ have the correct values on $T$, so does $d_{12}$ (by applying cancellation).

This completes the induction step and thus the proof.

*Proof of Proposition* 1. We may assume, without loss of generality, that the trees in question have all their leaves labeled. First we verify assertion (1). Suppose $\varepsilon$ is an element of $\Lambda$ such that $2\varepsilon \neq 0$. Then let $n = 4$ and consider the matrix $d$ for which $d_{ij} = 0$ if $i + j$ is even and $d_{ij} = \varepsilon$ otherwise. This matrix has no realization on the tree with four leaves, for which the path joining leaves 1 and 2 is disjoint from the path joining leaves 3 and 4; cf. the case $n = 4$ in the proof of Theorem 1.

Conversely, assume that $\Lambda$ is Boolean. Consider the bush $S$ with leaves labeled $1, \ldots, n$. Given any fixed $k$, assign weight $d_{ik}$ to the edge incident with leaf $i$ ($i = 1, \ldots, n$). Since $2\varepsilon = 0$ for all $\varepsilon \in \Lambda$, we infer the equality $d_{ij} = d_{ik} + d_{jk}$ from the 4-point condition, so $d$ is realized by $S$ with this weighting. Now every tree $T$ that contains vertices labeled $1, \ldots, n$ can be obtained from a subdivision $T_0$ of $S$ by successively applying the following "edge swap" operation: given the tree $T_k$, so far constructed, realizing $d$, assume that some vertex $x$ of $T_k$ is connected to two vertices $y$ and $z$ by edges weighted $\alpha$ and $\beta$, respectively. Then remove the edge between $x$ and $z$ and create a new edge of weight $\alpha + \beta$ connecting $y$ and $z$ instead. The resulting tree $T_{k+1}$ induces $d$ as well. Eventually, we arrive at a subdivision $T_m$ of $T$. Finally, contract all edges incident with unlabeled vertices of degree 2, which are not in $T$, and thereby add up the weights; this yields $T$.

As to (2), suppose $\Lambda$ is an abelian group that contains an element $\varepsilon$ of order 4. Define a $5 \times 5$ matrix $d$ with entries in $\Lambda$ by setting $d_{ij} = 2\varepsilon$ precisely if $\{i, j\} = \{2, 4\}, \{2, 3\}$, or $\{1, 4\}$ and setting $d_{ij} = 0$ otherwise. Then $d$ satisfies the 4-point condition with respect to $\Lambda$. However, $d$ has no tree representation with an edge weighting from $\Lambda$. This can be seen by restricting $d$ to the sets $\{1, 2, 3, 4\}, \{1, 3, 4, 5\}$, and $\{1, 2, 3, 5\}$. Specifically, for $\{1, 2, 3, 4\}$,

$$d_{13} + d_{24} \neq d_{14} + d_{23} = d_{12} + d_{34},$$

and so, for any tree representation of $d$ restricted to $\{1, 2, 3, 4\}$, the path joining the vertices labeled 1 and 3 must be disjoint from the path joining the vertices labeled 2 and 4. Similarly, by considering $\{1, 3, 4, 5\}$ and $\{1, 2, 3, 5\}$, we require that the path joining 3 and 5 is disjoint from the one joining 1 and 4 and that the path joining 1 and 5 is disjoint from the one joining 2 and 3, respectively. Clearly, however, these three constraints cannot be realized on a single tree, as claimed.

Conversely, suppose $\Lambda$ has no element of order 4. Let $\Lambda_0$ denote the subgroup of $\Lambda$ generated by the entries in $d$, together with one solution for the subsets $\{i, j, k, l\}$ of size at least 3 of the equation required of $d$ by the 4-point condition for $i, j, k, l$. Since $\Lambda_0$ is finitely generated and has no element of order 4, the structure theorem for finitely generated abelian groups implies that there is an isomorphism $\phi\colon \Lambda_0 \to \Gamma \times \Delta$, where $\Gamma$ has no elements of order 2 and $\Delta$ is a Boolean group. Let $d^\Gamma$ and $d^\Delta$ denote the projections of $\phi(d)$ onto $\Gamma$ and $\Delta$, respectively. Then $d^\Gamma$ satisfies the 4-point condition according to $\Gamma$, and so, applying the previous theorem, there is a tree $T$ and a weighting of its edges by nonzero elements of $\Gamma$ that realizes $d^\Gamma$. We now "expand" $T$ to allow for a tree representation for $d^\Delta$. Specifically, for each vertex $v$ of $T$ that is assigned a set $S$ of $s > 1$ labels, make $v$ adjacent to $s$ new leaves and assign each such leaf a unique label from $S$, thereby obtaining a tree $T'$ having only singleton labels. Extend the previous edge weighting by $\Gamma$ of $T$ to $T'$ by assigning weight $0 \in \Gamma$ to the new edges. Since $d^\Delta$ satisfies the 4-point condition with respect to $\Delta$, part (1) shows that there is a weighting of the edges of $T'$ by elements of $\Delta$ that realizes $d^\Delta$. Now, for each edge $e$ of $T'$ let

$$\lambda(e) = \phi^{-1}(\gamma(e), \delta(e)),$$

where $\gamma(e) \in \Gamma$ and $\delta(e) \in \Delta$ are the weights that were previously assigned to $e$ by considering $d^\Gamma$ and $d^\Delta$, respectively. Then $\lambda(e) \in \Lambda_0$, and the weighting of the edges of $T'$ described by $\lambda$ realizes $d$. This completes the proof.

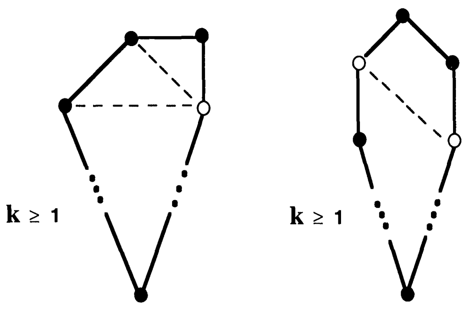


FIG. 4. *Configurations excluded by the 4-point condition with respect to* $\mathbf{R}$.

**Distance-hereditary graphs.** Let $G$ be an unweighted, undirected, connected graph with vertices numbered 1 through $n$. The shortest-path metric $d$ of $G$ (which counts the edges in the shortest path connecting pairs of vertices in $G$) then takes values among $0, 1, \ldots, n-1$. There are several options for the abelian monoid $\Lambda$ with respect to which the 4-point condition can be considered. If we take $\Lambda = \mathbf{N}$, then $d$ satisfies this condition if and only if $G$ is an unweighted tree. For $\Lambda = \frac{1}{2}\mathbf{N}$ the corresponding 4-point condition characterizes block graphs (graphs in which every maximal 2-connected subgraph (block) is complete); see Howorka [7]. The case $\Lambda = \frac{1}{2}\mathbf{Z}$ is more interesting, since it leads to a metric description of distance-hereditary graphs; see Bandelt and Mulder [2]. $G$ is said to be *distance-hereditary* if every induced path (or subgraph) is isometric, that is, constitutes a subspace with respect to the metric $d$. Actually, we may compute distance modulo $2k+1$ for any $k \geq 1$ and arrive at the same class of graphs.

PROPOSITION 2. *A graph is distance-hereditary if and only if its shortest-path metric $d$ satisfies the* 4*-point condition with respect to an abelian monoid* $\Lambda$, *where* $\Lambda$ *may be chosen as* $\Lambda = \frac{1}{2}\mathbf{Z}$ (*or* $\mathbf{R}$) *or* $\mathbf{Z}_m$ *for* $m \geq 3$ *odd. Furthermore, $G$ is bipartite and distance-hereditary if and only if its metric $d$ satisfies the* 4*-point condition with respect to* $\Lambda = \mathbf{Z}$.

In view of Theorem 1, we can thus uniquely code a distance-hereditary graph $G$ by a weighted tree over $\frac{1}{2}\mathbf{Z}$, where the vertices of the tree with degree smaller than three are labeled by the vertices of $G$. An immediate consequence of this is the following result (known to several people by now): the isomorphism problem for distance-hereditary graphs is easy. Indeed, two distance-hereditary graphs are isomorphic if and only if their associated weighted trees are isomorphic. For general graphs, by contrast, determining the complexity of the isomorphism question is a difficult and still unsolved problem [5]. Furthermore, the automorphism group of a distance-hereditary graph is isomorphic to that of a tree.

*Proof of Proposition* 2. According to [2, Thm. 2], $G$ is distance-hereditary if and only if $d$ satisfies the 4-point condition with respect to $\mathbf{R}$, so we only have to adjust for the expression of the distances modulo $m$. Assume that $G$ is not distance-hereditary. Then there exists an isometric subgraph of the form shown in Fig. 4 possessing a cycle of length $2k+3$ or $2k+4$ ($k \geq 1$) with two (or one) possible chords as indicated by the dotted lines. The four shaded vertices in either cycle yield distance sums $k+1, k+2, k+3$ and $k+1, k+3, k+5$, respectively. In either case these are all different provided $m$ is not 2 or 4.

Evidently, $G$ is bipartite if and only if each distance sum

$$d_{ij} + d_{jk} + d_{ki}$$

is even. This is precisely the case when $d$ satisfies the instances, for which $\#\{i, j, k, l\} = 3$, of the 4-point condition with respect to $\Lambda = \mathbf{Z}$. Thence the result.

## REFERENCES

[1] H.-J. BANDELT *Recognition of tree metrics*, SIAM J. Discrete Math., 3 (1990), pp. 1–6.

[2] H.-J. BANDELT AND H. M. MULDER, *Distance-hereditary graphs*, J. Combin. Theory Ser. B, 41 (1986), pp. 182–207.

[3] J.-P. BARTHÉLEMY AND A. GUÉNOCHE, *Trees and proximity representations*, Wiley, New York, 1991.

[4] P. BUNEMAN, *A note on the metric property of trees*, J. Combin. Theory Ser. B, 17 (1974), pp. 48–50.

[5] M. R. GAREY AND D. S. JOHNSON, *Computers and intractibility*, Bell Telephone Laboratories Ltd., New Jersey, 1979.

[6] M. D. HENDY, *The path sets of weighted partially labelled trees*, Austral. J. Combin., 6 (1992), pp. 277-284.

[7] E. HOWORKA, *On metric properties of certain clique graphs*, J. Combin. Theory Ser. B, 27 (1979), pp. 67–74.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 526–535, November 1995

# CORRELATION OF BOOLEAN FUNCTIONS AND PATHOLOGY IN RECURSION TREES *

INGO ALTHÖFER† AND IMRE LEADER‡

**Abstract.** A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is called trivial if it depends on only one coordinate. We show that nontrivial Boolean functions of positively correlated random variables are strictly *less correlated* than the variables themselves. This improves on a correlation inequality of Witsenhausen.

Over the last decade, several people in computer science and computer chess have investigated the problem of quality and reliability in game-tree searching, where the heuristic evaluation function is not free of errors. In random models with independent leaf values and independently occuring errors, a phenomenon of pathology was observed: the deeper the search in the tree, the worse the final estimate of the root value. The main result of this note implies that pathology is not only a feature of game trees, but appears in any sequence of increasing bivalued recursion trees with independent leaf values, independently occuring errors, and nontrivial recursion rules.

**Key words.** tree search, reliable computing, correlation inequalities, game trees

**AMS subject classifications.** 68M15, 68T20, 60E15, 90D35

## 1. Introduction.

Let $X = (X_1, \dots, X_n)$ and $Y = (Y_1, \dots, Y_n)$ be two sequences of random variables, where each pair $(X_i, Y_i)$ is independent of the others. Witsenhausen [Wit] derived a lower bound for the probability of disagreement among any pair of two-valued functions $f(X)$ and $g(Y)$, depending on the maximum correlation $\max_{1 \le i \le n} \operatorname{cor}(X_i, Y_i)$ but not on the value of $n$ itself.

Our aim in this note is to prove some strict correlation inequalities for such Boolean functions of random variables. We start with some notation. Let $X_1, \dots, X_n$ and $Y_1, \dots, Y_n$ be random variables with values in $\{0,1\}$, and assume that for all $i \in \{1, \dots, n\}$ the following statements hold:

(i) for each $i$, the dependent pair $(X_i, Y_i)$ is independent of $\{(X_j, Y_j) | j \neq i\}$;

(ii) $\mathrm{Prob}\{X_i = 1\} = \mathrm{Prob}\{Y_i = 1\} = p_i$;

(iii) $X_i$ and $Y_i$ are nonnegatively correlated, i.e., there is some $\varepsilon_i, 0 \le \varepsilon_i \le 1$, such that

$$\mathrm{Prob}\{X_i = 0, Y_i = 1\} = \mathrm{Prob}\{X_i = 1, Y_i = 0\} = \varepsilon_i p_i (1 - p_i).$$

In information theory such a sequence of pairs of dependent random variables is called a correlated source. See, for instance, [CK] as a reference.

Let $p = \mathrm{Prob}\{f(X_1, \dots, X_n) = 1\} = \mathrm{Prob}\{f(Y_1, \dots, Y_n) = 1\}$ and put $\varepsilon = \min_{1 \le i \le n} \varepsilon_i$. Call a function $f : \{0,1\}^n \to \{0,1\}$ *trivial* if it depends on only one coordinate. Our main result is as follows.

THEOREM 1. *Let $X_1, \dots, X_n, Y_1, \dots, Y_n$ be as above and $f : \{0,1\}^n \to \{0,1\}$. Then*

(i) $\mathrm{Prob}\{f(X_1, \dots, X_n) = 0, f(Y_1, \dots, Y_n) = 1\} \ge \varepsilon p (1 - p)$,

*and*

---

* Received by the editors November 24, 1992; accepted for publication (in revised form) September 8, 1994.

†Fakultät für Mathematik, Universität Bielefeld, Postfach 100 131, 33501 Bielefeld, Germany. Current address: Fakultät für Mathematik und Informatik, Universität Jena, D-07740 Jena, Germany.

‡University of Cambridge, Department of Mathematics, 16 Mill Lane, Cambridge CB2 1SB, United Kingdom.

(ii) *if $f$ is nontrivial,* $0 < \varepsilon < 1$, *and* $0 < p_i < 1$ *for all* $i \in \{1, \dots, n\}$, *then strict inequality holds above.*
In particular, if all the $\varepsilon_i$ are equal then part (ii) of Theorem 1 states that $f(X)$ and $f(Y)$ are *strictly less* correlated than each pair $(X_i, Y_i)$.

We remark that Theorem 1(i) follows from a result in [Wit]. Indeed, Theorems 1(i) and 2 in §2 and Theorem 7(i) in §4 follow from Theorems 1 and 2 and Corollary (29) of [Wit]. However, for the application to recursion trees, part (ii) of our Theorem 1 is the crucial point, and this cannot be proved by the geometric technique of [Wit].

Section 2 contains the proof of Theorem 1 and is rather technical. In §3, we apply Theorem 1 to random game trees and other recursion trees. A probabilistic generalization of Theorem 1 is given in §4. Section 5 consists of concluding remarks.

**2. The proof of Theorem 1.** We start by reformulating the problem in terms of set systems in $\{0,1\}^n$. Let $A = f^{-1}(0) = \{a \in \{0,1\}^n | f(a) = 0\}$ and $A^c = \{0,1\}^n - A = f^{-1}(1)$. We have

$$\begin{aligned}
&\text{Prob}\{f(X_1, \dots, X_n) = 0, f(Y_1, \dots, Y_n) = 1\} \\
&= \sum_{a \in A} \sum_{b \in A^c} \text{Prob}\{(X_1, \dots, X_n) = a, (Y_1, \dots, Y_n) = b\} \\
&= \sum_{a \in A} \sum_{b \in A^c} \prod_{i=1}^{n} \text{Prob}\{X_i = a_i, Y_i = b_i\},
\end{aligned}$$

where $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$.

Thus, letting

$$g_A = \sum_{a \in A} \sum_{b \in A^c} \prod_{i=1}^{n} e_i(a, b),$$

where

$$e_i(a, b) = \text{Prob}\{X_i = a_i, Y_i = b_i\} = \begin{cases} \varepsilon_i p_i(1 - p_i) & \text{if } a_i \neq b_i, \\ p_i - \varepsilon_i p_i(1 - p_i) & \text{if } a_i = b_i = 1, \\ 1 - p_i - \varepsilon_i p_i(1 - p_i) & \text{if } a_i = b_i = 0, \end{cases}$$

part (i) of Theorem 1 claims precisely that

$$g_A \geq \varepsilon p(1 - p).$$

It seems hard to prove this result *directly*. Instead, we shall generalize this statement to a stronger statement in such a way that this stronger statement can be proved by induction on $n$. We shall consider a function of a pair $A, B$ of set systems, rather than just one set system $A$.

For $A, B \subset \{0,1\}^n$, define

$$g_{A,B}(\varepsilon_1, \dots, \varepsilon_n) = \sum_{a \in A} \sum_{b \in B^c} \prod_{i=1}^{n} e_i(a, b).$$

Note that, for any set $A \subset \{0,1\}^n$, we have

$$g_A = g_{A,A}(\varepsilon_1, \dots, \varepsilon_n). \tag{2.1}$$

For convenience, we write $g_{A,B}$ for $g_{A,B}(\varepsilon_1,\dots,\varepsilon_n)$ and $x_i$ for $\varepsilon_i p_i(1-p_i)$. For $a = (a_1,\dots,a_n) \in \{0,1\}^n$, define the *weight* of $a$ to be

$$w(a) = \prod_{i=1}^{n} q_i(a), \quad \text{where } q_i(a) = \begin{cases} p_i & \text{if } a_i = 1, \\ 1-p_i & \text{if } a_i = 0, \end{cases}$$

and for $A \subset \{0,1\}^n$, set $w_A = \sum_{a\in A} w(a)$.

Our more general result is the following theorem.

THEOREM 2. *Let $A, B \subset \{0,1\}^n$. Then*

$$g_{A,B} + g_{B,A} \geq \varepsilon[w_A(1-w_A) + w_B(1-w_B)] + (w_A - w_B)^2, \quad \textit{where } \varepsilon = \min_{1\leq i\leq n} \varepsilon_i.$$

The quadratic term $(w_A - w_B)^2$ will be necessary to control some of the terms arising in the induction step. We remark that there seems to be no simple lower bound on the function $g_{A,B}$ alone, rather than $g_{A,B} + g_{B,A}$.

*Proof of Theorem* 1(i) *from Theorem* 2. Use Theorem 2 with $A = B = f^{-1}(0)$. Then $w_A = w_B = 1-p$, $(w_A - w_B)^2 = 0$, and by (2.1) we have $\text{Prob}\{f(X_1,\dots,X_n) = 0, f(Y_1,\dots,Y_n) = 1\} \geq \varepsilon w_A(1-w_A)$. □

*Proof of Theorem* 2. We proceed by induction on $n$. We start with the case $n = 0$.

*Case* $n = 0$. The assertion of Theorem 2 is symmetric in $A$ and $B$, and $|\{0,1\}^0| = 1$. Thus, we have to check three different cases, namely $|A| = |B| = 0$, $|A| = |B| = 1$, and $|A| \neq |B|$. In all these cases, the assertion of Theorem 2 holds with equality for all $\varepsilon \in \mathbb{R}$. ($\prod_{i=1}^{0} e_i(a,b) = 1$ for all $a \in A, b \in B^c$.)

*Induction step.* Let

$$\begin{aligned} A_0 &= \{a = (a_1,\dots,a_{n-1}) \in \{0,1\}^{n-1} | (a_1,\dots,a_{n-1},0) \in A\}, \\ A_1 &= \{a = (a_1,\dots,a_{n-1}) \in \{0,1\}^{n-1} | (a_1,\dots,a_{n-1},1) \in A\}, \\ B_0 &= \{b = (b_1,\dots,b_{n-1}) \in \{0,1\}^{n-1} | (b_1,\dots,b_{n-1},0) \in B\}, \\ B_1 &= \{b = (b_1,\dots,b_{n-1}) \in \{0,1\}^{n-1} | (b_1,\dots,b_{n-1},1) \in B\}. \end{aligned}$$

Then $A_0, A_1, B_0, B_1 \subset \{0,1\}^{n-1}$. Note that

$$(2.2) \qquad w_A = (1-p_n)w_{A_0} + p_n w_{A_1}, \qquad w_B = (1-p_n)w_{B_0} + p_n w_{B_1}.$$

Moreover,

$$g_{A,B} = (1-p_n-x_n)g_{A_0,B_0} + (p_n - x_n)g_{A_1,B_1} + x_n[g_{A_0,B_1} + g_{A_1,B_0}].$$

Thus

(2.3)
$$\begin{aligned} g_{A,B} + g_{B,A} &= (1-p_n-x_n)[g_{A_0,B_0} + g_{B_0,A_0}] \\ &\quad + (p_n - x_n)[g_{A_1,B_1} + g_{B_1,A_1}] + x_n[g_{A_0,B_1} + g_{A_1,B_0} + g_{B_0,A_1} + g_{B_1,A_0}]. \end{aligned}$$

The following transformations are lengthy, but not difficult.

By the induction hypothesis

$$\begin{aligned}(2.3) \geq{}& (1-p_n-x_n)[\varepsilon[w_{A_0}(1-w_{A_0})+w_{B_0}(1-w_{B_0})]+(w_{A_0}-w_{B_0})^2] \\ &+(p_n-x_n)[\varepsilon[w_{A_1}(1-w_{A_1})+w_{B_1}(1-w_{B_1})]+(w_{A_1}-w_{B_1})^2] \\ &+x_n\begin{bmatrix}\varepsilon[w_{A_0}(1-w_{A_0})+w_{B_1}(1-w_{B_1})]+(w_{A_0}-w_{B_1})^2 \\ +\varepsilon[w_{A_1}(1-w_{A_1})+w_{B_0}(1-w_{B_0})]+(w_{A_1}-w_{B_0})^2\end{bmatrix}\end{aligned} \tag{2.4}$$

$$\begin{aligned}={}& \varepsilon(1-p_n)[w_{A_0}(1-w_{A_0})+w_{B_0}(1-w_{B_0})] \\ &+\varepsilon p_n[w_{A_1}(1-w_{A_1})+w_{B_1}(1-w_{B_1})] \\ &+(1-p_n-x_n)(w_{A_0}-w_{B_0})^2 \\ &+(p_n-x_n)(w_{A_1}-w_{B_1})^2 \\ &+x_n(w_{A_0}-w_{B_1})^2+x_n(w_{A_1}-w_{B_0})^2.\end{aligned} \tag{2.5}$$

On the other hand, by (2.2) we have

$$\begin{aligned}&\varepsilon[w_A(1-w_A)+w_B(1-w_B)]+(w_A-w_B)^2 \\ &\quad=\varepsilon\begin{bmatrix}[(1-p_n)w_{A_0}+p_nw_{A_1}]\cdot[1-(1-p_n)w_{A_0}-p_nw_{A_1}] \\ +[(1-p_n)w_{B_0}+p_nw_{B_1}]\cdot[1-(1-p_n)w_{B_0}-p_nw_{B_1}]\end{bmatrix} \\ &\quad+[(1-p_n)w_{A_0}+p_nw_{A_1}-(1-p_n)w_{B_0}-p_nw_{B_1}]^2.\end{aligned} \tag{2.6}$$

It is sufficient to prove (2.5) $\geq$ (2.6). Now, (2.5) is linear in $x_n$, and so it is sufficient to prove this in the boundary cases $\varepsilon_n=\varepsilon$ and $\varepsilon_n=1$. For clarity, we split the argument up into a few cases.

*Case* $\varepsilon_n=\varepsilon$. Here we have

$$\begin{aligned}(2.5)={}&\varepsilon(1-p_n)[\cdots]+\varepsilon p_n[\cdots] \\ &+(1-p_n-\varepsilon p_n(1-p_n))(w_{A_0}-w_{B_0})^2 \\ &+(p_n-\varepsilon p_n(1-p_n))(w_{A_1}-w_{B_1})^2 \\ &+\varepsilon p_n(1-p_n)[(w_{A_0}-w_{B_1})^2+(w_{A_1}-w_{B_0})^2].\end{aligned} \tag{2.7}$$

Again, (2.7) and (2.6) are linear in $\varepsilon$, so it is sufficient to prove (2.7) $\geq$ (2.6) in the boundary cases $\varepsilon=0$ and $\varepsilon=1$.

*Subcase* $\varepsilon=0$.

$$\begin{aligned}(2.7)&=(1-p_n)(w_{A_0}-w_{B_0})^2+p_n(w_{A_1}-w_{B_1})^2, \\ (2.6)&=[(1-p_n)(w_{A_0}-w_{B_0})+p_n(w_{A_1}-w_{B_1})]^2.\end{aligned}$$

Hence (2.7) $\geq$ (2.6) by the convexity of the function $x^2$.

*Subcase* $\varepsilon=1$. It is easy to check that in this case we have (2.7) = (2.6).

*Case* $\varepsilon_n=1$. Here we have

$$\begin{aligned}(2.5)={}&\varepsilon(1-p_n)[\cdots]+\varepsilon p_n[\cdots] \\ &+(1-p_n)^2(w_{A_0}-w_{B_0})^2 \\ &+p_n^2(w_{A_1}-w_{B_1})^2 \\ &+p_n(1-p_n)[(w_{A_0}-w_{B_1})^2+(w_{A_1}-w_{B_0})^2].\end{aligned} \tag{2.8}$$

Since (2.8) and (2.6) are both linear in $\varepsilon$, it is sufficient to prove (2.8) $\geq$ (2.6) in the boundary cases $\varepsilon=0$ and $\varepsilon=1$.

*Subcase* $\varepsilon = 0$.

$$\begin{aligned}(2.8) &= (1-p_n)^2(w_{A_0}-w_{B_0})^2 + p_n^2(w_{A_1}-w_{B_1})^2 \\ &\quad + p_n(1-p_n)[(w_{A_0}-w_{B_1})^2 + (w_{A_1}-w_{B_0})^2], \\ (2.6) &= (1-p_n)^2(w_{A_0}-w_{B_0})^2 + p_n^2(w_{A_1}-w_{B_1})^2 \\ &\quad + 2p_n(1-p_n)(w_{A_0}-w_{B_0})(w_{A_1}-w_{B_1}).\end{aligned}$$

Thus it remains to show that

$$(w_{A_0}-w_{B_1})^2 + (w_{A_1}-w_{B_0})^2 \geq 2(w_{A_0}-w_{B_0})(w_{A_1}-w_{B_1}).$$

However, this is equivalent to

$$(w_{A_0}-w_{A_1})^2 + (w_{B_0}-w_{B_1})^2 \geq 0.$$

*Subcase* $\varepsilon = 1$. This subcase is identical to the earlier case when $\varepsilon_n = \varepsilon$ and $\varepsilon = 1$. This completes the proof of Theorem 2. □

We now turn to a proof of the second part of Theorem 1. This is also proved by induction on $n$.

For $1 \leq i \leq n$, let us write

$$A_0(i) \quad \text{for}$$
$$\{(a_1,\dots,a_{i-1},a_{i+1},\dots,a_n) \in \{0,1\}^{n-1} | (a_1,\dots,a_{i-1},0,a_{i+1},\dots,a_n) \in A\}$$

and analogously

$$A_1(i) \quad \text{for}$$
$$\{(a_1,\dots,a_{i-1},a_{i+1},\dots,a_n) \in \{0,1\}^{n-1} | (a_1,\dots,a_{i-1},1,a_{i+1},\dots,a_n) \in A\}.$$

Thus in the proof of Theorem 2 our $A_0$ and $A_1$ were just $A_0(n)$ and $A_1(n)$.

We start with a simple fact about nontrivial sets in $\{0,1\}^n$, for $n \geq 3$.

LEMMA 3. *Let $n \geq 3$, and let $A \subset \{0,1\}^n$ be nontrivial. Then there is some $i, 1 \leq i \leq n$ such that $A_0(i)$ or $A_1(i)$ is nontrivial.*

*Proof.* The proof is straightforward. □

The next lemma proves Theorem 1(ii) for the case $n = 2$.

LEMMA 4. *Let $A \subset \{0,1\}^2$, with $A$ nontrivial, and suppose that $0 < p_1 < 1, 0 < p_2 < 1, 0 < \varepsilon < 1$, and $\varepsilon \leq \varepsilon_1, \varepsilon_2$. Then*

$$g_{A,A} > \varepsilon w_A(1-w_A).$$

*Proof.* There are only two different types of nontrivial sets in $\{0,1\}^2$. For each, the result is easy to check. □

*Proof of Theorem* 1(ii). We proceed by induction on $n$. For $n \leq 1$, all subsets of $\{0,1\}^n$ are trivial. By Lemma 4, the result holds for $n = 2$.

*Induction step.* In the proof of Theorem 2, we divided the sets $A$ and $B$ into $A_0, A_1$ and $B_0, B_1$ according to coordinate $n$. However, we could equally well have chosen to split $A$ and $B$ according to another coordinate, say $i$. By Lemma 3, we can choose $i$ such that at least one of the sets $A_0(i)$ and $A_1(i)$ is nontrivial. Since both $1 - p_i - x_i$ and $p_i - x_i$ are strictly positive, we have by the induction hypothesis

$$\begin{aligned}&(1-p_i-x_i)(g_{A_0(i),A_0(i)} + g_{A_0(i),A_0(i)}) + (p_i - x_i)(g_{A_1(i),A_1(i)} + g_{A_1(i),A_1(i)}) \\ &\quad > 2(1-p_i-x_i)\varepsilon w_{A_0(i)}(1-w_{A_0(i)}) + 2(p_i-x_i)\varepsilon w_{A_1(i)}(1-w_{A_1(i)}).\end{aligned}$$

Hence the inequality between (2.3) and (2.4) in the proof of Theorem 2 (replacing coordinate $n$ with coordinate $i$) is strict if $A = B$ is nontrivial. This completes the proof of Theorem 1. $\square$

**3. Applications to recursion trees.** Theorem 1(ii) may be applied to prove that pathology occurs in game-tree models with independent leaf values and independently occuring errors in the heuristic evaluation function. For a general background on the phenomenon of pathology in game trees, see [Bea], [CN], [Nau], [Pea], and [Tze].

We interpret $n > 1$ as the number of leaves in the finite game tree, the $X_i$ as the true leaf values, and the $Y_i$ as the heuristic leaf values, and we let $f : \{0,1\}^n \to \{0,1\}$ be a concatenation of "negamax" functions $f_y : \{0,1\}^{\deg(y)} \to \{0,1\}$ for every inner node $y$ of the game tree: $f_y(a_1, \dots, a_{\deg(y)}) = \max\{(1-a_1), \dots, (1-a_{\deg(y)})\}$. Thus $f(x_1, \dots, x_n)$ represents the backed-up root value if the leaves have values $X_1, \dots, X_n$.

First of all, let us assume that all $\varepsilon_i$ are equal and that $0 < p_i < 1$ for all $i$. Then, according to Theorem 1(ii), the true root value $v_{\text{root}}$ and the backed-up heuristic estimate $h_{\text{root}}$ satisfy the following corollary.

COROLLARY 5. $\text{Prob}\{v_{\text{root}} = 0, h_{\text{root}} = 1\} > \varepsilon \,\text{Prob}\{v_{\text{root}} = 0\}\text{Prob}\{v_{\text{root}} = 1\}$.
Hence true and backed-up heuristic root value are *strictly less* correlated than the leaf values. This phenomenon of pathology, which contradicts observations from computer game-playing practice, occurs independently of the size and the shape of the game trees (in the model). In particular, searching deeper does *not* yield better estimates of the true root value. We remark that, in real life games like chess or checkers, the leaf values are not independent.

For the special case of a depth-regular and $m$-uniform game tree with an invariant probability $p(m) = \text{Prob}\{\text{a node has true value } 1\}$ for all nodes of the tree, Pearl [Pea, pp. 332–346] investigated the error probabilities

$$e_t = \text{Prob}\left\{\begin{array}{l}\text{a node at distance } t \text{ from the leaves} \\ \text{has true value 1 and backed-up heuristic} \\ \text{value 0}\end{array}\right\}.$$

In the interesting case $0 < e_0 < p(m)[1 - p(m)]$ he proved

$$\lim_{t \to \infty} e_t = p(m)[1 - p(m)].$$

This also follows from an iterated application of Theorem 1(ii). Hence in very high game trees, true and heuristic root values are nearly independent of each other. For this "tendency to independency" we use the term "strong pathology."

Whereas Pearl's analytical approach is restricted to depth-regular and uniform game trees with identically distributed leaf values, Theorem 1(ii) allows us to prove strong pathology in nonregular trees as well. For this, we first give a quantitative formulation of Theorem 1(ii).

THEOREM 1′. *For every triple* $(n, p^*, \varepsilon^*)$ *with* $n \in \mathbb{N}, 0 < p^* < 1$, *and* $0 < \varepsilon^* < 1$, *there exists a constant* $\delta = \delta(n, p^*, \varepsilon^*) > 0$ *such that statement* (i) *of Theorem* 1 *can be substituted by the stronger statement*

$$\text{(i}'\text{)} \quad \text{Prob}\{f(X_1, \dots, X_n) = 0, f(Y_1, \dots, Y_n) = 1\} \geq (1 + \delta)\varepsilon p(1-p),$$

*whenever* $f : \{0,1\}^n \to \{0,1\}$ *is nontrivial,* $p^* \leq p_i \leq 1 - p^*$, *and* $\varepsilon^* \leq \varepsilon_i \leq 1 - \varepsilon^*$ *for* $i = 1, \dots, n$.

*Proof.* As a first step, we prove Theorem 1′ for a fixed nontrivial function $f$ (with a constant $\delta_f = \delta(f, p^*, \varepsilon^*)$ instead of $\delta(n, p^*, \varepsilon^*)$). Indeed, we observe that the three quantities $1 - p = \text{Prob}\{f(X_1, \dots, X_n) = 0\}, p = \text{Prob}\{f(Y_1, \dots, Y_n) = 1\}$, and $\text{Prob}\{f(X_1, \dots, X_n) = 0, f(Y_1, \dots, Y_n) = 1\}$ are polynomial expressions in the $p_i$ and the $\varepsilon_i$, and hence are continuous functions of the $p_i$ and the $\varepsilon_i$. $\varepsilon = \min \varepsilon_i$ is also continuous.

Because

$$S(p^*, \varepsilon^*) = \left\{ (p_1, \dots, p_n, \varepsilon_1, \dots, \varepsilon_n) \,\middle|\, \begin{array}{l} p^* \le p_i \le 1 - p^* \text{ and } \varepsilon^* \le \varepsilon_i \le 1 - \varepsilon^* \\ \text{for all } i \end{array} \right\}$$

is a compact set, and $0 < p < 1$ by the nontriviality of $f$, the continuous function

$$h(p_1, \dots, p_n, \varepsilon_1, \dots, \varepsilon_n) = \frac{\text{Prob}\{f(X_1, \dots, X_n) = 0, f(Y_1, \dots, Y_n) = 1\}}{\varepsilon p(1 - p)}$$

assumes its minimum (say $1 + \delta_f$) on $S(p^*, \varepsilon^*)$, and this minimum must be $> 1$ by part (ii) of Theorem 1.

Since there are only finitely many functions $f : \{0,1\}^n \to \{0,1\}$ for every fixed $n$, we get $\delta = \min \delta_f > 0$. □

We now describe an interesting class of recursion trees which we can prove are strongly pathological.

A *bivalued recursion tree* consists of a finite rooted tree $T$ and recursion functions $f_x : \{0,1\}^{\deg(x)} \to \{0,1\}$ for every inner node $x$ in $T$. We write $L$ for the set of leaves of $T$. If values $v(z) \in \{0,1\}$ for all $z \in L$ are given, the inner nodes of $T$ get their $v$-values recursively by $v(x) = f_x(v(y_1), \dots, v(y_n))$, where $y_1, \dots, y_n$ are the direct successors of $x$. If probabilities $p_z = \text{Prob}\{v(z) = 1\}$ for all $z \in L$ are given, and if different leaves assume their values independently of each other, we can compute $p_x = \text{Prob}\{v(x) = 1\}$ for all nodes $x$ of $T$. Given $\{p_z | z \in L\}$, an inner node $x$ with successors $y_1, \dots, y_n$ is called *$p^*$-central,* for a $p^* \ge 0$, if $f_x$ is nontrivial (which implies $n \ge 2$) and $p^* \le p_{y_i} \le 1 - p^*$ for $i = 1, \dots, n$. The bivalued recursion tree $(T, \{f_x | x \in T\}, \{p_z | z \in L\})$ is called *$k$-rich* with respect to $p^*$ if every root–leaf path in $T$ contains at least $k$ $p^*$-central nodes. For instance, $m$-uniform game trees of regular depth $t$ with invariant probability $p(m)$ are $t$-rich with respect to $\min\{p(m), 1 - p(m)\}$.

If we denote true and heuristic leaf values by $X_z$ and $Y_z$, respectively, and the backed-up values for an inner node $x$ by $X_x$ and $Y_x$ (so that $X_x = f_x(X_{y_1}, \dots, X_{y_n})$ and $Y_x = f_x(Y_{y_1}, \dots, Y_{y_n})$), then our task is to estimate $\text{Prob}\{X_{\text{root}} = 0, Y_{\text{root}} = 1\}$. We say the recursion tree has *leaf errors* $\ge \varepsilon^*$ if

$$\varepsilon^* p_z(1 - p_z) \le \text{Prob}\{X_z = 0, Y_z = 1\} \le p_z(1 - p_z)$$

for all $z \in L$.

Formally, the pathology in $k$-rich recursion trees may be stated as follows.

THEOREM 6. *For every triple $(n, p^*, \varepsilon^*)$ with $n \ge 2, 0 < p^* < 1$, and $0 < \varepsilon^* < 1$, there is a monotone increasing sequence $(c_k)_{k=0}^{\infty}$ of positive real numbers, with $c_0 = \varepsilon^*$ and $\lim_{k \to \infty} c_k = 1$, such that*

$$\text{Prob}\{X_{\text{root}} = 0, Y_{\text{root}} = 1\} \ge c_k p_{\text{root}}(1 - p_{\text{root}})$$

*in every bivalued recursion tree which has maximum degree $\le n$, leaf errors $\ge \varepsilon^*$, and is $k$-rich with respect to $p^*$.*

*Proof.* This follows from the $k$-rich property by an iterated application of Theorem 1(i) (for the nodes that are not $p^*$-central) and Theorem 1′ (for the nodes that are $p^*$-central). □

A more general discussion of pathology in game trees and other recursion trees appears in [Alt].

**4. A probabilistic generalization of Theorem 1.** Interpreting the $Y_i$ as heuristic estimates of the true data $X_i$, it is quite natural to ask for other (deterministic or probabilistic) estimation functions $\hat{f} : \{0,1\}^n \to \{0,1\}$ instead of $f$ itself. $\hat{f}$ is called *unbiased* with respect to $f$ if $\text{Prob}\{\hat{f}(Y_1,\dots,Y_n) = 1\} = \text{Prob}\{f(X_1,\dots,X_n) = 1\}$. Theorem 1 may be generalized to this situation as follows.

THEOREM 7. *Let* $X_1,\dots,X_n,Y_1,\dots,Y_n$ *be as in* §1, $f : \{0,1\}^n \to \{0,1\}$ *a deterministic function, and* $\hat{f} : \{0,1\}^n \to \{0,1\}$ *a probabilistic or deterministic function that is unbiased with respect to* $f$. *Put* $\varepsilon = \min_{1\le i\le n} \varepsilon_i$. *Then*

(i) $\text{Prob}\{f(X_1,\dots,X_n) = 0, \hat{f}(Y_1,\dots,Y_n) = 1\} \ge \varepsilon p(1-p)$
*and*

(ii) *if* $f$ *is nontrivial,* $0 < \varepsilon < 1$, *and* $0 < p_i < 1$ *for all* $i$, *then strict inequality holds above.*

*Sketch of proof.* First, we generalize the notation $g_{A,B}$ for sets $A$ and $B$ to a function $g_{R,S}$ for fractional sets $R, S$; in other words, functions $R, S : \{0,1\}^n \to [0,1] \subset \mathbb{R}$.

$$g_{R,S} = \sum_{a\in\{0,1\}^n} \sum_{b\in\{0,1\}^n} R(a)[1 - S(b)] \prod_{i=1}^{n} e_i(a,b).$$

For $A \subset \{0,1\}^n$, set

$$R(a) = \begin{cases} 1 & \text{if } a \in A, \\ 0 & \text{if } a \notin A. \end{cases}$$

We say that the fractional set $R$ *belongs* to $A$. Thus, if $R$ belongs to $A$ and $S$ belongs to $B$ then $g_{A,B} = g_{R,S}$.

We define the weight of a fractional set $R$ to be

$$w_R = \sum_{a\in\{0,1\}^n} R(a)w(a).$$

With this notation we can formulate a probabilistic generalization of Theorem 2.

THEOREM 8. *Let* $R$ *and* $S$ *be fractional sets on* $\{0,1\}^n$. *Then*

$$g_{R,S} + g_{S,R} \ge \varepsilon[w_R(1-w_R) + w_S(1-w_S)] + (w_R - w_S)^2,$$

*where* $\varepsilon = \min_{1\le i\le n} \varepsilon_i$.

The proof of Theorem 8 is analogous to that of Theorem 2, and is therefore omitted. □

Setting

$$R(a) = \begin{cases} 1 & \text{if } f(a) = 1, \\ 0 & \text{if } f(b) = 0, \end{cases}$$

and $S(b) = \text{Prob}\{\hat{f}(b) = 1\}$, we obtain part (i) of Theorem 7.

For the proof of part (ii), we start by considering the case $n = 2$.

LEMMA 9. *If $n = 2, 0 < p_1 < 1, 0 < p_2 < 1, 0 < \varepsilon < 1, \varepsilon \leq \varepsilon_1, \varepsilon_2$, the fractional set $R$ belongs to one of the two sets $\{00, 11\}$ or $\{00\}$, and $S$ is an arbitrary fractional set on $\{0,1\}^n$, then*

$$g_{R,S} + g_{S,R} > \varepsilon[w_R(1 - w_R) + w_S(1 - w_S)] + (w_R - w_S)^2.$$

*Sketch of proof.* The difference between Lemmas 9 and 4 is that we now have more choices for the fractional set $S = (s_{00}, s_{01}, s_{10}, s_{11})$. Because the functions $g_{R,S}$ and $g_{S,R}$ are linear in each of the $s_{ij}$, we have only to check the boundary cases with $s_{ij} \in \{0,1\}$ for all $i, j$. This can be done by a case by case analysis (altogether $2 \cdot 8 = 16$ cases). The details are omitted. □

It is now easy to complete the proof of Theorem 7(ii), just as in the proof of Theorem 1(ii). □

**5. Concluding remarks.** (a) Unfortunately Theorem 7(ii) cannot be used to prove an analogue to the *strong* pathology theorem (Theorem 6) for arbitrary tree evaluation functions. Nevertheless, such an analogue does exist. It is proved in [Alt] by other methods.

(b) For our proofs it is important that $X_i$ and $Y_i$ have the same marginal distributions. The result of Witsenhausen [Wit, p. 107] also provides lower error bounds in the general case with arbitrary marginals. However, these bounds are often not tight—for wide ranges of probability distributions they are surpassed by bounds given in a paper of Ahlswede and Gács [AG], who use other measures of correlation.

(c) It would be desirable to have generalizations of our results for the case with more than only two values. For general finite sets we make the following conjecture, based on analogy with Theorem 1.

Let $X_1, \dots, X_n, Y_1, \dots, Y_n$ be random variables with values in the finite sets $M_1, \dots, M_n (|M_i| \geq 2)$, and assume that for all $i \in \{1, \dots, n\}$ the following statements hold:

(i) for each $i$, the dependent pair $(X_i, Y_i)$ is independent of $\{(X_j, Y_j) | j \neq i\}$;

(ii) $\text{Prob}\{X_i = j\} = \text{Prob}\{Y_i = j\} = p_{ij}$ for all $j \in M_i$;

(iii) there exists a constant $\varepsilon, 0 \leq \varepsilon \leq 1$, such that for every $A \subset M_i$

$$\begin{aligned} &2\varepsilon\,\text{Prob}\{X_i \in A\}\text{Prob}\{Y_i \in A^c\} \\ &\leq \text{Prob}\{X_i \in A, Y_i \in A^c\} + \text{Prob}\{X_i \in A^c, Y_i \in A\} \\ &\leq 2\,\text{Prob}\{X_i \in A\}\text{Prob}\{Y_i \in A^c\}. \end{aligned}$$

This condition (iii) is more complicated than the corresponding condition in §1 because the error probabilities $\text{Prob}\{X_i = j, Y_i = k\}$ for $j \neq k$ are not assumed to be symmetric in $j$ and $k$. In the symmetric case the inequalities in (iii) would reduce to

$$\begin{aligned} &\varepsilon\,\text{Prob}\{X_i \in A\}\text{Prob}\{Y_i \in A^c\} \\ &\qquad \leq \text{Prob}\{X_i \in A, Y_i \in A^c\} \leq \text{Prob}\{X_i \in A\}\text{Prob}\{Y_i \in A^c\}. \end{aligned}$$

We call a function $f : M_1 \times \cdots \times M_n \to \{0,1\}$ *trivial*, if there is some $i \in \{1, \dots, n\}$ and some $A_i \subset M_i$, such that

$$f(x_1, \dots, x_n) = 0 \quad \Leftrightarrow x_i \in A_i.$$

Thus $f$ is trivial if it depends only on one coordinate.

We use the abbreviations $X = (X_1, \dots, X_n)$ and $Y = (Y_1, \dots, Y_n)$.

CONJECTURE 10. *Let the $X_i, Y_i$ be as above, and let $f : M_1 \times \cdots \times M_n \to \{0, 1\}$. Then*

(i) $2\varepsilon \operatorname{Prob}\{f(X) = 0\}\operatorname{Prob}\{f(Y) = 1\} \le \operatorname{Prob}\{f(X) = 0,\ f(Y) = 1\} + \operatorname{Prob}\{f(X) = 1, f(Y) = 0\} \le 2\operatorname{Prob}\{f(X) = 0\}\operatorname{Prob}\{f(Y) = 1\}$
*and*

(ii) *if $f$ is not trivial, $0 < \varepsilon < 1$, and $p_{ij} > 0$ for all $i \in \{1, \dots, n\}, j \in \{0, \dots, m_i - 1\}$, then the first inequality in* (i) *holds strictly.*

We believe that part (ii) of this conjecture might be genuinely more difficult to prove than part (i)—in contrast to the situation with Theorem 1.

It might be possible to deduce part (i) of Conjecture 10 from the results of Witsenhausen [Wit, p. 107] or Ahlswede and Gács [AG, p. 926]. The problem is that their parameters of correlation ($S = \cos\theta$ in [Wit], $s_p(W, P)$ in [AG]) cannot be easily computed in the nonbinary case.

## REFERENCES

[AG] R. AHLSWEDE AND P. GÁCS, *Spreading of sets in product spaces and hypercontraction of the Markov operator,* Ann. Prob., 4 (1976), pp. 925–939.

[Alt] I. ALTHÖFER, *On pathology in game tree and other recursion tree models,* Habilitationsschrift, Fakultät für Mathematik, Universität Bielefeld, Germany, 1991.

[Bea] D. F. BEAL, *An analysis of minimax,* in Proc. Advances in Computer Chess 2, M. R. B. Clarke, ed., Edinburgh University Press, Scotland, 1982, pp. 103–109.

[CK] I. CSISZÁR AND J. KÖRNER, *Information Theory: Coding Theorems For Discrete Memoryless Systems,* Academic Press, New York, 1981.

[CN] P.-C. CHI AND D. S. NAU, *Comparison of the minimax and product back-up rules in a variety of games,* in Search in Artificial Intelligence, L. Kanal and V. Kumar, eds., Springer, New York, 1988, pp. 450–471.

[Nau] D. S. NAU, *Pathology on game trees: A summary of results,* in Proc. 1st National Conference on Artificial Intelligence, 1980, pp. 102–104.

[Pea] J. PEARL, *Heuristics: Intelligent Search Strategies for Computer Problem Solving,* Addison-Wesley, Reading, MA, 1984.

[Tze] C. H. TZENG, *A Theory of Heuristic Information in Game Tree Search,* Symbolic Computation—Artificial Intelligence, Springer, New York, 1988.

[Wit] H. S. WITSENHAUSEN, *On sequences of pairs of dependent random variables,* SIAM J. Appl. Math., 28 (1975), pp. 100–113.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 536–542, November 1995

# LOWER BOUNDS ON FORMULA SIZE OF BOOLEAN FUNCTIONS USING HYPERGRAPH ENTROPY*

ILAN NEWMAN† AND AVI WIGDERSON‡

**Abstract.** Körner defined the notion of graph entropy. He used it to simplify the proof of the Fredman–Komlos lower bound for the family size of perfect hash functions.

We use this information-theoretic notion to obtain a general method for formula size lower bounds. This method can be applied to low-complexity functions for which the other known general methods do not apply.

**Key words.** Boolean functions, graph entropy, circuit complexity

**AMS subject classifications.** 68R05, 68Q10, 05C99

## 1. Introduction.

Körner [7] defined the notion of graph entropy. This notion is information theoretic and was used in [8] to simplify the proof of the Fredman–Komlos [2] lower bound for the family size of perfect hash functions.

We use this information-theoretic notion to obtain a general method for formula size lower bounds for Boolean functions. This method can be applied to low-complexity functions for which the other known general methods ([11], [12], [3], and see also [18]) do not apply. Specifically the results are as follows:

1. a new general lower bound on the formula size of quadratic Boolean functions;
2. as a corollary we get an $\Omega(n^2 \log n)$ lower bound for the function that decides whether a graph of n vertices has a cycle of length 4, and to the function that decides whether a graph has a vertex of degree at least 2;
3. a simple proof of a result of Krichevskii [10], stating that the formula size for the threshold-2 Boolean function with $n$ variables is at least $n \log n$;
4. a simple proof of a lower bound first proved by Snir [17], stating that a $\vee \wedge \vee$ formula for an $n$-variable threshold-$k$ function, where all $\wedge$ gates have fan in $k$, has a size of

$$\Omega\left( n \frac{\log n - \log(k-1)}{\log k - \log(k-1)} \right) = \Omega\left(nk \log \frac{n}{k}\right).$$

## 2. Definitions and notation.

Let $X$ be a finite set, interpreted as Boolean variables. A *formula* is a rooted tree whose leaves are labeled with members of $X$ or their negations and whose internal nodes are labeled with the Boolean operations AND, OR. The root of the tree computes a Boolean function $f; \{0,1\}^X \mapsto \{0,1\}$ in the natural way. If no negations appear, we say that the formula and the function computed are both monotone. The size of a formula is the number of leaves in the tree.

Let $f$ be any Boolean function. The *formula size* of $f$ is the minimum size of a formula that computes $f$; it is denoted by $L(f)$. For a monotone function $f$, the minimum size of a monotone formula for $f$ is denoted by $L_M(f)$.

* Received by the editors October 15, 1990; accepted for publication (in revised form) January 27, 1992.

† Computer Science Department, Hebrew University, Jerusalem 91904, Israel and Department of Mathematics and Computer Science, University of Haifa, Haifa 31905, Israel.

‡ Computer Science Departement, Hebrew University, Jerusalem 91904, Israel. The research of this author was partially supported by American-Israeli Binational Science Foundation grant 87-00082.

The threshold-$k$ Boolean function, denoted by $T_k^n$, is a Boolean function on $n$ variables that gets value 1 if and only if the input has at least $k$ variables assigned 1.

The set $\{1, \ldots, n\}$ is denoted as [n].

For a Boolean function $f$ on $n$ variables, we will assume that the variables are numbered from 1 to $n$, and by writing $f(T) = 0$ $(f(T) = 1)$, $T \subseteq$[n], we mean that $f(x) = 0$ $(f(x) = 1)$ for the characteristic vector $x$ of $T$.

A hypergraph $G = (V, E)$ is a set of "vertices" $V$, and a set $E$ of subsets of $V$ (also called the set of "edges"). If all edges $e \in E$ have a constant size $k$, the hypergraph is called $k$-uniform. A 2-uniform hypergraph is simply a graph. The $s$-uniform hypergraph on $n$ vertices that contains all subsets of size $s$ is called the complete $s$-uniform hypergraph and is denoted by $K_n^s$. The complete graph on $n$ vertices is denoted by $K_n$.

For a graph or hypergraph, an independent set $I$ is a subset of $V$ that contains no edges of $E$.

For a probability distribution $Q_{XY}$ on a cross product $A \times B$, $Q_X$ $(Q_Y)$ denotes the marginal distribution of $Q_{XY}$ on $A$ $(B)$.

All logarithms in this paper are to the base 2.

## 3. Definition and basic properties of entropy and hypergraph entropy.

1. Let $X$ and $Y$ be random variables in some probability space. The entropy of $X$ is defined as

$$H(X) = \sum_x p(x) \log \frac{1}{p(x)}.$$

The mutual information between $X$ and $Y$ is defined as $I(X, Y) = H(X) + H(Y) - H((X, Y))$ and it may be written also as:

$$I(X, Y) = H(X) - \sum_{x,y} p(x, y) \log \frac{1}{p(x|y)}.$$

Further information on information theory may be found in [1].

2. Körner [7], [8] defined the notion of hypergraph entropy as follows: Let $G(V, E)$ be a hypergraph and $P$ a probability distribution on $V$. Let $A(G)$ be the collection of all maximal independent sets of $G$.

Define $\mathcal{Q}(G, P)$ to be the set of all probability distributions $Q_{XY}$ on $V \times A(G)$ such that, for every $v \in V$,

(a) $Q_{XY}(v, I) = 0$ if $v \notin I$.

(b) $Q_X(v) = P(v)$ (where $Q_X(.)$ is the marginal distribution of $Q_{XY}$ on $V$).

The hypergraph entropy $H(G, P)$ is defined as

$$H(G, P) = \min\{I(X, Y) | Q_{XY} \in \mathcal{Q}(G, P)\},$$

where $I(X, Y)$ is the mutual information between two random variables $X$ and $Y$ that are distributed according to the marginal distributions $Q_X$ and $Q_Y$.

3. Hereafter, we consider only uniform distributions on $V$, so we refer to the hypergraph entropy of $G$ as $H(G)$.

4. We shall need the following basic properties of $H(G)$ proved by Körner and Marton [7]–[9].

(i) For two hypergraphs on the same vertex set $G(V, E_G), F(V, E_F)$, let $K = G \cup F$ be the hypergraph on V with $E = E_G \cup E_F$; then $H(K) \leq H(G) + H(F)$.

(ii) The hypergraph entropy is monotone—that is, deleting an edge can only decrease the entropy.

(iii) The entropy of the complete $k$-uniform hypergraph is $H(K_n^k) = \log n - \log(k-1)$.

(iv) The entropy of a bipartite graph on $m$ (out of $n$) vertices does not exceed $\frac{m}{n}$.

(v) The generalization for hypergraphs is as follows: Let $G = (V, E)$ be a $k$-uniform hypergraph. We call $G$ a $k$-partite hypergraph if there is a partition of $V$ into $k$ parts $V_1, \ldots, V_k$ such that, for every edge $e \in E$ and every $V_i$, $i \in$[k], $|e \cap V_i| = 1$.

We have that the entropy of the $k$-partite hypergraph on $m$ (out of $n$) vertices is no more than $\frac{m}{n}(\log k - \log(k-1))$ .

**4. A lower bound for formula size of Boolean functions.** In this section, we develop a general technique for formula lower bounds. A natural approach is to associate a nonnegative cost function $\mu$ to each Boolean function with the property that if $f = g \diamond h$, then $\mu(f) \leq \mu(g) + \mu(h)$ (where $\diamond$ is either $\wedge$ or $\vee$). Such a cost function is called an "abstract complexity measure" [18], [16]. It directly gives a lower bound on the formula size of a Boolean function in terms of its cost. We find that for monotone formulae, graph entropy is a natural choice for such a measure. It leads to nontrivial lower bounds for monotone formulae for quadratic functions (§4.1). We then extend it to the nonmonotone case using a lemma of Krichevskii.

**4.1. A lower bound for monotone formula size.** We assign each monotone function a cost and prove that the cost of a function computed by $\vee$, $\wedge$ gates is no more than the sum of costs of the inputs (thus the cost function is an "abstract complexity measure" for monotone formulae). The cost of a single variable will be $\frac{1}{n}$. Hence, (by induction on the formula) for a function of cost $\mu$ one gets a lower bound of $n\mu$.

The definition of the cost function is given below.

DEFINITION 4.1. *Let $g : \{0,1\}^n \mapsto \{0,1\}$ be Boolean function $g$ on $n$ variables. We identify the variable set with the set $[n]$. Define the following:*

1. *$(g)_k$ is the set of "minterms" of $g$ of size $k$, formally; $(g)_k = \{S | \ S \subseteq [n], |S| = k, \ g(S) = 1, \ \forall T \subset S \ g(T) = 0\}$.*

2. *We are interested only in $(g)_1$ and $(g)_2$. Observe that $(g)_1$ is a subset of $[n]$ and $(g)_2$ is a set of unordered pairs on $[n]$. $(g_2)$ is identified with the graph $G(g) = (V, E)$, $V = [n]$, $E = (g)_2$.*

3. *The cost $\mu$ of a function $g$ is defined as*

$$\mu(g) = H(G(g)) + \frac{|(g)_1|}{n}.$$

THEOREM 4.2. *Let $g$ be a monotone Boolean function. Let $L_M(g)$ be the monotone formula size of $g$. Then $L_M(g) \geq n\mu(g)$.*

*Proof.* We note the following:

1. For a variable $x_i$ (a leaf of a formula), $(x_i)_1 = \{i\}$, $G(x_i) = \phi$ (the empty graph), and so $\mu(x_i) = \frac{1}{n}$.

2. The cost function is monotone with respect to inclusion; if $(g)_1 \subseteq (h)_1$ and $(g)_2 \subseteq (h)_2$ , then $\mu(g) \leq \mu(h)$.

*Subadditivity for $\vee$ gate.* Let $g = h \vee f$. We have $(g)_1 = (h)_1 \cup (f)_1$ and $(g)_2 \subseteq G(h) \cup G(f)$; thus

$$\mu(g) \leq \frac{|(h)_1 \cup (f)_1|}{n} + H(G(h) \cup G(f))$$

$$\leq \frac{|(h)_1|}{n} + \frac{|(f)_1|}{n} + H(G(h)) + H(G(f)) = \mu(h) + \mu(f).$$

The first inequality follows from the monotonicity of $\mu$. The second follows from 4(i) in §3.

*Subadditivity for $\wedge$ gate.* Let $g = h \wedge f$. Denote $A = (h)_1$, $B = (f)_1$.

We get that $(g)_1 = A \cap B$ and $(g)_2 \subseteq G(h) \cup G(f) \cup G((A-B),(B-A))$, where $G(L,M)$ denotes the complete bipartite graph $G$ with parts $L$ and $M$. Thus

$$\begin{aligned}\mu(g) &\leq H(G(h) \cup G(f) \cup G((A-B),(B-A))) + \frac{|A\cap B|}{n}\\ &\leq H(G(h)) + H(G(f)) + \frac{|A-B|+|B-A|}{n} + \frac{|A\cap B|}{n}\\ &\leq H(G(h)) + H(G(f)) + \frac{|A|+|B|}{n} = \mu(h)+\mu(f).\end{aligned}$$

The first inequality follows from the monotonicity of $\mu$. The second follows from 4(i) and 4(iv) in §3.

The theorem now follows, since $\mu(t) = \frac{1}{n}$ for any leaf $t$ of the formula, and the cost of the output function does not exceed the sum of costs of all the leaves. ☐

*Remark.* We note here that the best this method can give (by direct application) are lower bounds of at most $n \log n$.

**4.2. The general lower bound.** We use a lemma (Krichevskii [10]) to extend our monotone lower bound method to nonmonotone formulae.

LEMMA 4.3 (see [10]). *Let $f(x_1,\dots,x_n)$ be any Boolean function for which $f(S) = 0$ for every $S$, $|S|=1$. Then there is a monotone function $\psi_f$ such that the following hold:*

1. $\psi_f(S) = 0$ *for any* $S$, $|S| = 1$.
2. $\psi_f(S) \geq f(S)$ *for every* $S$, $|S| = 2$.
3. $L_M(\psi_f) \leq L(f)$.

For completeness, we present a proof.

*Proof.* The proof is by induction on the formula size $L(f)$. For $L(f) = 2$, the claim is true. Let $F$ be an optimal formula for $f$. If $F = G \vee H$, where $G$ $(H)$ is the optimal formula for $g$ $(h)$, then by induction there are $\psi_g$ and $\psi_h$ for which conditions 1–3 of Lemma 4.3 are satisfied. It is easy to see that $\psi_f = \psi_g \vee \psi_h$ satisfies conditions 1–3 of Lemma 4.3 for $f$.

If $F = G \wedge H$ with the functions $g$ and $h$, respectively, then define $G_1 = \{x_i | g(\{x_i\}) = 1, \textit{ and } x_i \textit{ appears in } G\}$. Define $H_1$ similarly. By the assumption on $f$, it follows that $G_1 \cap H_1 = \phi$. Assume (w.l.o.g) that $G_1 = \{x_1,\dots,x_k\}$ and $H_1 = \{x_{k+1},\dots,x_{k+l}\}$. Let

$$\begin{aligned}F^* = (&x_1 \vee \cdots \vee x_k \vee G(0,\dots,0,x_{k+1},\dots,x_n))\\ &\wedge(x_{k+1} \vee \cdots \vee x_{k+l} \vee H(x_1,\dots,x_k,0,\dots,0,x_{k+l+1},\dots,x_n)).\end{aligned}$$

We have that $F^*$ is a formula for some function $f^*$. It is easy to verify that, for any $S$ with $|S| = 1$, $f^*(S) = 0$, and, for any $S$ with $|S| = 2$, $f^*(S) \geq f(S)$. In addition, $G(0,\dots,0,x_{k+1},\dots,x_n)$ and $H(x_1,\dots,x_k,0,\dots,0,x_{k+l+1},\dots,x_n)$ are formulae of some functions $g^*$ and $h^*$ that meet the requirements of the lemma, so by induction there are monotone functions $\psi_{g^*}$ and $\psi_{h^*}$ with monotone formulae $G^*$ and $H^*$ as required. Observe that by plugging $G^*$ and $H^*$ into $F^*$, we get a monotone formula for $\psi_f$ that satisfies conditions 1–3 of Lemma 4.3. ☐

We state now the main theorem for nonmonotone formulae.

THEOREM 4.4. *Let $f$ be a Boolean function with $f(S) = 0$ for every $S$, $|S| = 1$. Then $L(f) \geq n\mu(f)$.*

*Proof.* By the previous lemma, there is a monotone function $\psi_f$ for which $L(f) \geq L_M(\psi_f)$. From Theorem 4.2, we get that $L_M(\psi_f) \geq n\mu(\psi_f)$. Since $\psi_f(S) \geq f(S)$ for $|S| \leq 2$, the monotonicity of the cost function $\mu$ implies the result. □

**4.3. Application to specific functions.** Let $C4(n)$ be the Boolean function that decides "1" on a graph of $n$ vertices if the graph contains a cycle of length 4. Let $D2(n)$ be the Boolean function that decides "1" on a graph of $n$ vertices if the graph contains a vertex of degree at least 2.

Note that $C4(n)$ and $D2(n)$ are Boolean functions on $N = \binom{n}{2}$ variables.

COROLLARY 4.5. *Any formula for $D2(n)$ has size $\Omega(n^2 \log n)$.*

*Proof.* By Theorem 4.4, it is enough to show that $\mu(D2(n)) = \Omega(\log n)$. Observe that $(D2(n))_1 = \phi$ and $(D2(n))_2 = L(K_n)$, the line graph of $K_n$ (the graph whose vertices are the edges of $K_n$; two edges are connected if they have a common vertex in $K_n$).

We show that $\mu(D2(n)) = \Omega(\log n)$ by explicitly specifying the optimal distributions according to the definition of graph entropy. We do this by showing an upper bound of $\log(n-1)$ and $\log \frac{n}{2}$ on the graph entropies of $L(K_n)$ and its complement, respectively. (Note that the sum of these two numbers is $\log \binom{n}{2}$.) However, by 4(i) and 4(iii) in §3, the sum of the two graph entropies must be at least $\log \binom{n}{2}$; thus the upper bounds are, in fact, tight.

The independent sets of $L(K_n)$ are matchings (in $K_n$). The cliques in $L(K_n)$ are stars and triangles (in $K_n$). (A star is a set of all edges adjacent to a vertex.) Let $\mathcal{M}$ denote the set of perfect matchings, $\mathcal{S}$ denote the set of maximal stars, and $E$ denote the edge set of $K_n$. Define the probability $Q_1$ on $E \times \mathcal{M}$; $Q_1(e|M) = \frac{2}{n}$ for every matching $M \in \mathcal{M}$, $e \in M$, and such that the induced probability on $\mathcal{M}$ is uniform. Let $(X_1, Y_1)$ be a random variable on $E \times \mathcal{M}$ distributed according to $Q_1$. Define a probability distribution $Q_2$ on $E \times \mathcal{S}$; $Q_2(e|S) = \frac{1}{n-1}$ for every star in $\mathcal{S}$, $e \in \mathcal{S}$, and such that the induced probability on $\mathcal{S}$ is uniform. Let $(X_2, Y_2)$ be a random variable on $E \times \mathcal{S}$ distributed according to $Q_2$.

It is easy to check, using the definitions in §3, that $I_1(X_1, Y_1) = \log(n-1)$ and $I_2(X_2, Y_2) = \log \frac{n}{2}$. However,

$$\log \binom{n}{2} \leq H(L(G)) + H(L(G)^C) \leq I_1 + I_2 = \log \binom{n}{2}.$$

The first inequality follows from 4(i) in §3 and the fact that $L(G) \cup L(G)^C$ is the complete graph on $\binom{n}{2}$ vertices. The second inequality follows from the definition of graph entropy. Thus we get equality all the way and $H(L(G)) = I_1 = \Omega(\log n)$. □

COROLLARY 4.6. *Any formula for $C4(n)$ has size $\Omega(n \log n)$.*

*Proof.* Let $f$ be the restriction of $C4(n+1)$ obtained by the following procedure: Take a special vertex $z$ and set all edges adjacent to it to "1." Clearly, if there is a vertex of degree at least 2 in the remaining graph (the graph induced by unset edges), then there is a cycle of length 4 in the original graph. We have $(f)_1 = \phi$ and $(f)_2 \supseteq (D2)_2$, so by the monotonicity of the cost, we get that $\mu(D2) \leq \mu(f)$, and the result follows by Corollary 4.5. □

COROLLARY 4.7 [10]. *Let $T$ be any formula that computes $T_2^n$; then the size of $T$ is at least* nlogn. (*We note here that this is the best possible result.*)

*Proof.* $(T_2^n)_1 = \phi$ and $G(T_2^n) = K_n$ (the complete graph on $n$ vertices); thus by 4(iii) in §3, $\mu(T_2^n) = \log n$. □

*Remark.* A proof of the monotone formula lower bound for $T_2^n$ was given also by Hansel [5]. (See also [13].)

## 5. A lower bound on the size of a $\vee\wedge\vee$ formula for a threshold-$k$ function, where $\wedge$ gates' fan-in is $k$.

A $\vee\wedge\vee$ formula, where $\wedge$ gates have fan-in $k$, is a formula of the form $\vee_{i=1}^{p} \wedge_{j=1}^{k} \vee_{q\in S_{ij}} t_q$, where $t_q \in \{x_q, \neg x_q\}$ for every $q$.

The notion of hypergraph entropy can be used here to give a simple proof to a lower bound of Snir for such restricted formulae.

THEOREM 5.1 [17]. *The size of a $\vee\wedge\vee$ formula for $T_k^n$, where $\wedge$ gates have fan-in $k$, is at least*

$$s \geq \frac{n \log \frac{n}{k-1}}{\log \frac{k}{k-1}}.$$

The original proof was based on some ad hoc combinatorial considerations. We will follow the lines of the proof of the previous section.

*Proof.* Consider a minimum-size $\vee\wedge\vee$ formula for $T_k^n$—that is, of the form $\vee_{i=1}^{p} \wedge_{j=1}^{k} \vee_{q\in S_{ij}} t_q$. Let $\{g_i,\ i = 1, \dots, p\}$ be the functions computed at the $\wedge$ gates. Clearly, the formula must be monotone (that is, no negations), and for every fixed $i$, $1 \leq i \leq p$, the sets $S_{ij}$, $1 \leq j \leq k$ are pairwise disjoint.

Let $g$ be any Boolean function; define, as in the previous section, $(g)_k = \{S \subseteq [\mathrm{n}] : |S| = k,\ g(S) = 1$, and, for all $T \subset S$, $g(T) = 0\}$. Define the hypergraph $G_i$ whose edge set is $(g_i)_k$. We get that $G_i$ is a $k$-partite hypergraph on vertex set $\bigcup_j S_{ij}$. (The "parts" are $S_{ij}$, $1 \leq j \leq k$.) Similarly, define $T$ to be the hypergraph whose edge set is $(T_k^n)_k$. $T$ is the complete $k$-regular hypergraph $K_n^k$. Since $T_k^n = \vee_i g_i$, we get that $(T_k^n)_k = \cup_i (g_i)_k$—that is, a formula of this kind for $T_k^n$ defines a way to decompose the complete $k$-regular hypergraph to a union of $k$-partite hypergraphs. The size of the formula is

$$s = \sum_{i=1}^{p} \sum_{j=1}^{k} |S_{ij}| = \sum_{i=1}^{p} |V(G_i)|.$$

The hypergraph entropy of $K_n^k$ is $\log n - \log(k-1)$ (from 4(iii) in §3). For each $G_i$, we have $H(G_i) = \leq \frac{|V(G_i)|}{n}(\log k - \log(k-1))$ (by 4(v) in §3). Thus, by the subadditivity of the hypergraph entropy (4(i) in §3),

$$\begin{aligned} H(K_n^k) = \log n - \log(k-1) &\leq \sum_{i=1}^{p} H(G_i) \\ &\leq \frac{1}{n}(\log k - \log(k-1)) \sum_{i=1}^{p} |V(G_i)| = \frac{s}{n} \log \frac{k}{k-1}, \end{aligned}$$

and we get the desired lower bound

$$s \geq \frac{n \log \frac{n}{k-1}}{\log \frac{k}{k-1}}. \quad \square$$

Some remarks are due here.

1. This result was significantly improved recently by J. Radhakrishnan [14] using graph-entropy methods. He proved a near-optimal lower bound for any $\vee\wedge\vee$ formulae for $T_k^n$ of the order $e^{\delta(k)}n\log n$, where $\delta(k) = \Omega(\frac{\sqrt{k}}{\log^2 k})$ and $k < \log n$ (see also [15]).

2. For constant $k$, there is an (optimal) construction of $O(n\log n)$ $\vee\wedge\vee$ formulae for $T_k^n$ [6], [4].

**Acknowledgments.** We are grateful to Mauricio Karchmer and Aviad Cohen for helpful discussion.

## REFERENCES

[1] I. Csiszar and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*, Academic Press, New York, 1982.

[2] M. Fredman and J. Komlos, *On the size of separating systems and perfect Hash functions*, SIAM J. Algorithms Discrete Methods, 5 (1984), pp. 61–68.

[3] M. J. Fischer, A. R. Meyer, and M. S. Paterson, *$\Omega(n\log n)$ lower bounds on length of Boolean formulas*, SIAM J. Comput., 11 (1982), pp. 416–427.

[4] J. Friedman, *Constructing $O(n\log n)$-size monotone formulae for the kth elementary symmetric polynomial of n Boolean variables*, SIAM J. Comput., 15 (1986), pp. 641–654.

[5] G. Hansel, *Nombre minimal de contacts de fermature nessecaires pour realiser une function booleenne symetrique de n variables*, C. R. Acad. Sci. Paris, 258 (1964), pp. 6037–6040.

[6] L. S. Khasin, *Complexity bounds for the realization of monotone symmetrical functions by means of formulas in the basis $+ * -$*, Soviet Phys. Dokl., 14 (1970), pp. 1149–1151.

[7] J. Körner, *Coding of an information source having ambiguous alphabet and the entropy of graphs*, Trans. 6th Prague Conf. on Information Theory, Academia, Prague, 1973, pp. 441–425.

[8] ———, *Fredman–Komlos bounds and information theory*, SIAM J. Algorithms Discrete Methods, 7 (1986), pp. 560–570.

[9] J. Körner and K. Marton, *New bounds for perfect hashing via information theory*, European J. Combin., 9 (1988), pp. 523–530.

[10] R. E. Krichevskii, *Complexity of contact circuits realizing a function of logical algebra*, Soviet Phys. Dokl., 8 (1964), pp. 770–772.

[11] V. M. Krapchenko, *A method of obtaining lower bounds for the complexity of $\pi$-schemes*, Math. Notes Acad. Sci. USSR, 11 (1972), pp. 474–479.

[12] E. I. Nechiporuk, *A Boolean function*, Soviet Math. Dokl., 7 (1966), pp. 999–1000.

[13] N. Pippenger, *An information-theoretic method in combinatorial theory*, J. Combin. Theory, Ser. A, 23 (1977), pp. 99–104.

[14] J. Radhakrishnan, *Improved bounds for covering complete uniform hypergraphs*, Inform. Process. Lett., 41 (1992), pp. 203–207.

[15] ———, *Better bounds for threshold formulas*, Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991, pp. 314–323.

[16] A. A. Razborov, *On submodular complexity measures*, in Boolean Function Complexity: Selected Papers from LMS Symp. Durham, M. S. Paterson, ed., Cambridge University Press, Cambridge, 1990, pp. 76–83.

[17] M. Snir, *The covering problem of complete uniform hypergraphs: A note*, Discrete Math., 27 (1979), pp. 103–105.

[18] I. Wegener, *The Complexity of Boolean Functions*, Wiley–Teubner Series in Computer Science, B. G. Teubner, Stuttgart, 1987.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 543–554, November 1995

# FEASIBLE OFFSET AND OPTIMAL OFFSET FOR GENERAL SINGLE-LAYER CHANNEL ROUTING*

RONALD I. GREENBERG† AND JAU-DER SHIH‡

**Abstract.** This paper provides an efficient method to find all feasible offsets for a given separation in a very large-scale integration (VLSI) channel-routing problem in one layer. The previous literature considers this task only for problems with no single-sided nets. When single-sided nets are included, the worst-case solution time increases from $\Theta(n)$ to $\Omega(n^2)$, where $n$ is the number of nets. But if the number of columns $c$ is $O(n)$, the problem can be solved in time $O(n^{1.5} \lg n)$, which improves upon a "naive" $O(cn)$ approach. As a corollary of this result, the same time bound suffices to find the optimal offset (the one that minimizes separation). Better running times result when there are no two-sided nets or all single-sided nets are on one side of the channel. This paper also gives improvements upon the naive approach for $c \neq O(n)$, including an algorithm with running time independent of $c$. An interesting algorithmic aspect of the paper is a connection to discrete convolution.

**Key words.** VLSI layout, channel routing, single-layer wire routing, discrete convolution, combinatorial algorithms

**AMS subject classifications.** 68Q35, 68Q25

## 1. Introduction.

Much attention has been given to planar or single-layer wire routing for very large-scale integration (VLSI) chips. Most popular has been river routing in the restricted sense of the term, the connection of two parallel rows of corresponding points,[1] e.g., [11] and the references therein. Other works have considered routing within a rectangle [2], placement and routing within a ring of pads [1], or routing between very general arrangements of modules [10], [4].

Ironically, single-layer routing may become more relevant as technology evolves toward chips with increasing numbers of layers. With many layers, it becomes more likely that an individual layer can be dedicated to a coplanar subset of the original collection of nets. For example, the heuristic multilayer channel router MulCh [7] improved upon previous multilayer channel routers by dividing the problem into essentially independent subproblems of one, two, or three layers.

In this paper, we consider the single-layer channel-routing problem, which is more general than the more heavily studied river-routing problem. Channel routing is similar to river routing in that both deal with the interconnection of terminals lying in two parallel rows (sides of the channel); also, for simplicity, we restrict attention to two-point nets as in river routing.[2] But we allow nets that have both their terminals on the same side of the channel, contrary to river routing. The existence of these *single-sided* nets is both realistic (as in the example problems of [7]) and a significant algorithmic complication. As shown in Fig. 1, the usual convention is to draw the rows of terminals horizontally; only the region between these rows is available for routing.

* Received by the editors July 12, 1993; accepted for publication (in revised form) June 9, 1994. This research was supported in part by National Science Foundation grant CCR-9109550.

† Electrical Engineering Department, University of Maryland, College Park, Maryland 20742 (rig@eng.umd.edu).

‡ Department of Information Engineering, Kaohsiung Polytechnic Institute, Ta-Hsu, Kaohsiung 84008, Taiwan, Republic of China (jdshih@nas04.kpi.edu.tw).

[1] This is the only use of the term "river routing" in this paper; we refer to more complicated variations of the problem as "single-layer" or "planar" routing.

[2] Multiterminal nets can be handled by a transformation that might be considered "folklore." It is described in [8] in the context of showing that minimum separation problems can be solved even more easily than by actually applying the transformation.

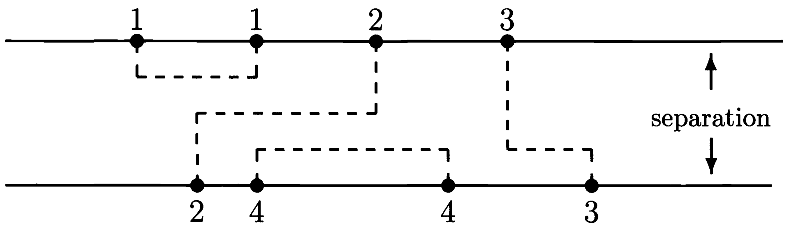


FIG. 1. *An example of a routed single-layer channel.*

We refer to single-sided nets that have their two terminals on the top (bottom) as *upper* (*lower*) *nets*. Nets with terminals on opposite sides are referred to as *two-sided* nets. We restrict attention to a rectilinear, grid-based model in which terminals lie on gridpoints and wires are disjoint paths through grid edges. We use $c$ to denote the total number of grid columns from the leftmost terminal to the rightmost terminal and $n$ to denote the number of nets.

The greatest attention has been given to the *minimum separation* version of the problem. In this case, we assume that the horizontal positions of the terminals are completely fixed, but we seek the minimum vertical distance between the two rows of terminals that allows the routing to be completed. An $O(n)$ time solution in the river-routing case was given in [6]. Though some erroneous solutions have been published for the general channel-routing case, a simple and correct $O(n)$ algorithm is provided in [8].

In this paper, other important versions of the river-routing problem are solved in the context of channel routing; in these problems, we allow the rows of terminals to be offset relative to one another. That is, we allow the upper row of terminals to be slid as a block to the left or right, though individual terminals do not shift position relative to one another. (This models the situation in which we are trying to wire together two modules, each having terminals on one side, and we have substantial freedom on how to place the modules.) The *optimal offset problem* involves finding the offset that minimizes the amount of separation necessary to route the problem. A related problem, which we refer to as the *feasible offset problem*, is to determine all offsets that are feasible (i.e., give enough room to route) at a given separation. In the river-routing context, the second problem is usually called the *offset range problem*, since the feasible offsets always constitute a single continuous range, but this property does not hold for channels with single-sided nets.

Mirzaian [11] showed that feasible offset and optimal offset can be computed in $O(n)$ time in the river-routing case, but we are not aware of any published solutions for channels with single-sided nets. One complication that arises when single-sided nets are included is that the solution time is no longer insensitive to the number of columns in the problem (at least for feasible offset). As illustrated in Fig. 2, if the number of columns is large, the number of disjoint intervals of feasible offsets may be $\Omega(n^2)$. But if $c = O(n)$, we show that feasible offset can be solved in $O(n^{1.5} \lg n)$ time. This improves on the naive $O(cn)$ time obtained by running the $O(n)$ algorithm for the minimum separation problem at each of the $2c$ offsets that may need to be checked. In the remainder of this paper, we express our running times in terms of $c$ as well as $n$ where necessary but concentrate on obtaining a good running time when $c = O(n)$. Later, we give an algorithm that is less efficient for $c = O(n)$ but has a running time independent of $c$.

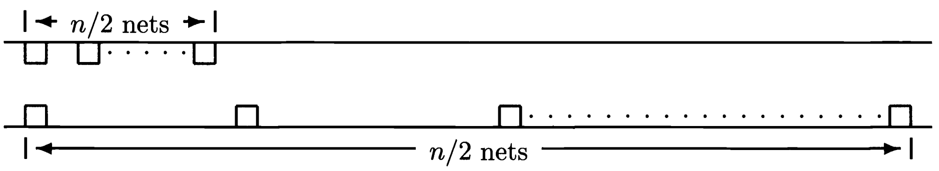


FIG. 2. *For small separation, the number of disjoint intervals of feasible offsets of the channel above is* $\Omega(n^2)$.

The remainder of this paper is organized as follows. In §2, we introduce some additional terminology and notation and show how to solve the feasible offset problem for a channel in which all nets are single sided. In this case, the running time with $c = O(n)$ is $O(n^{1.5}\sqrt{\lg n})$, which leads to an $O(n^{1.5}\sqrt{\lg n})$ algorithm for optimal offset. (The optimal offset problem as defined above is trivial when all nets are single sided; large offset minimizes separation. But we can handle a nontrivial generalization of the problem in which certain offsets are disallowed.) In §3, we show how to combine ideas from §2 with some new ideas to obtain solutions for channels with both single-sided and two-sided nets. For the general channel, the running time to solve either feasible offset or optimal offset is $O(n^{1.5}\lg n)$. Section 4 provides concluding remarks and some additional results. In particular, feasible offset and optimal offset can be solved in time $O(n^2 \lg n)$ independent of $c$. Also, the *optimal placement problem*, involving multiple modules on each side of the channel, can be handled in $O(n^3)$ time.

## 2. Channels with single-sided nets only.

In this section, we deal with the special case of channels with only single-sided nets. Much of the work we do here will help us in the next section where we consider channels that have both single-sided and two-sided nets.

We begin by explaining some notation and terminology that we use throughout this paper. First, we use $L$, $U$, and $T$ for the sets of lower, upper, and two-sided nets, respectively, and $N$ for the complete set of nets in the channel. In addition, we often use the same notation interchangeably for a set of nets or for a lower or upper *contour*. The contour of a set of lower nets is the upper boundary of the routing region consumed in the routing of those nets that minimizes total wire length. That is, when the nets are routed as tightly as possible against the bottom of the channel, the contour is formed by the uppermost nets and portions of the channel boundary. The contour of a set of upper nets is defined similarly. We also refer often to subsets of contours, which simply means restricting the contour to certain columns (even though there may be no set of nets that would generate the resulting contour). We use the notation FOP and OOP to refer to the feasible offset problem and optimal offset problem, respectively. We also use the more precise notation FOP$(s, A)$ to represent the set of solutions to the feasible offset problem with separation $s$ and the set $A$ of nets (or contours or contour fragments). We also use analogous notation SSFOP and SSOOP for the corresponding problems when all nets are single sided. (For optimal offset, we permit the problem specification to disallow some set of offsets, e.g., all offsets $\geq c/2$; otherwise SSOOP is trivial.)

Our first step in solving SSFOP is to find the contours of the upper and lower nets. We use Pinter's result that $O(n)$ time suffices to find a contour (i.e., the coordinates of all the bends in the contour) [12].

LEMMA 2.1 (Pinter). *The contour of a set of $n$ single-sided nets can be found in $O(n)$ time.* $\square$

Once we find the contours of the upper and lower nets, SSFOP can be expressed simply in terms of these contours. At each column, we define the *extension* of a contour to be the distance that the contour extends into the channel at that column. Then we are simply seeking all offsets for which no vertical cut corresponds to extensions of the upper and lower contours that sum to more than the separation under consideration. One way to solve this problem would be to compute the discrete convolution of the two sequences of extensions with the max and + operators substituted for the usual + and $\times$. It is unknown whether max, + convolution for vectors of length $n$ can be computed in better than $\Theta(n^2)$ time; still it will be seen that there is some relationship between convolution and our solution technique for SSFOP.

We begin with a general lemma that allows us to decompose SSFOP into smaller instances of the problem. In each of the smaller problems, we use only a portion of the lower contour, while retaining the entire upper contour. In fact, the lemma applies even when there are also two-sided nets. (Naturally, we also could switch the roles of the lower and upper contours.)

LEMMA 2.2. *Let $L_1, L_2, \ldots, L_k$ be any subsets of the contour $L$ of the lower nets such that $L_1 \cup L_2 \cup \cdots \cup L_k = L$, and let $A$ be an additional set of nets. Then* $\mathrm{FOP}(s, L \cup A) = \bigcap_{i=1}^{k} \mathrm{FOP}(s, L_i \cup A)$.

*Proof.* This follows from the fact that routing is possible if and only if each line segment from the top of the channel to the bottom of the channel is long enough (in the $L_\infty$ metric) to accommodate the number of nets that must cross it (i.e., each *cut* is *safe*). More details on the theory of single-layer routability can be found in [10]; see especially §2.1. $\square$

We now proceed to decompose the lower contour into pieces that are easier to handle and not too numerous. The next three lemmas are directed toward handling pieces of the contour that have large extension, and the following two lemmas handle portions of the contour in which there are not too many distinct extensions. Then we show how to put these two ideas together to solve the entire problem.

For the next lemma, we define a special type of contour fragment such that if it comprises the entire lower contour, then SSFOP is particularly easy to solve. A *monotonic* subset of the lower contour $L$ is a subset of $L$, such that the extensions within the selected columns are monotonically nondecreasing or monotonically nonincreasing as we move across the columns.

LEMMA 2.3. *If $L_m$ is a monotonic subset of the lower contour and $U$ is the upper contour, then we can solve* $\mathrm{SSFOP}(s, L_m \cup U)$ *in $O(n)$ time.*

*Proof.* Without loss of generality, assume the (nonzero portion of the) lower contour has nondecreasing extensions from left to right. We need only march across the columns of the upper contour once from left to right. Initially, we consider a far left position for the lower contour (highly negative offset). The check for each column of the upper contour involves adding the upper extension to the lower extension for the corresponding column of the lower contour, based on the current offset, and comparing to the upper bound on separation. After any unsuccessful check, the current offset is incremented and we do not yet advance to the next column of the upper contour. After each successful check, we move to the next column of the upper contour; prior columns never need to be rechecked at larger offsets since the lower contour is nondecreasing. When the rightmost column of the lower contour is involved in a successful check, a feasible offset has been found and, again, the current offset is incremented. The $O(c)$ approach just described can actually be improved to $O(n)$ time because of the following two facts. First, we really only need to look at columns of the upper contour where the upper contour bends. Second, there are at most $n$ places where the extension of the lower contour changes, and preprocessing of the lower contour will allow us to increment offset sufficiently after each unsuccessful check so that we can proceed immediately to the next bend point of the upper contour. $\square$

In the next lemma, we show that not only are monotonic pieces of contour easy to handle but that we don't have to check too many of them as long as we restrict attention to sections of contour with large extension. Here we define a monotonic subset to be *maximal* if no other monotonic subset contains it. Now we bound the number of maximal monotonic subsets in the portion of the contour with extension of at least $h$.

LEMMA 2.4. *Let $L_g$ be the subset of the lower contour containing only extensions greater than or equal to $h$. Then $L_g$ contains at most $c/2h$ maximal monotonic pieces.*

*Proof.* To have a maximal monotonic piece of the lower contour with extensions of at least $h$, there must be $h$ lower nets nested one inside the next. Therefore, a maximal monotonic piece with extensions greater than or equal to $h$ must span at least $2h$ columns, so $L_g$ contains at most $c/2h$ maximal monotonic pieces. $\square$

We can now put together Lemmas 2.2, 2.3, and 2.4 to solve SSFOP efficiently for any piece of lower contour in which all extensions are large enough.

LEMMA 2.5. *If $L_g$ is a subset of the lower contour containing only extensions greater than or equal to $h$ and if $U$ is the upper contour, then we can solve* SSFOP$(s, L_g \cup U)$ *in $O(cn/h)$ time.*

*Proof.* By Lemma 2.2, we know that it suffices to solve the problem independently for each of the maximal monotonic pieces of $L_g$. By Lemma 2.4 there are $O(c/h)$ such pieces, and by Lemma 2.3 $O(n)$ time suffices for each piece. $\square$

What remains is to solve the SSFOP for the portion of the lower contour with small extensions. (Later we'll show how to choose $h$, the dividing point between large and small extensions.) The next lemma handles the simplified case in which all extensions on the lower contour are 0 or 1. The following lemma goes on to handle $h$ distinct extensions.

LEMMA 2.6. *If all extensions are* 0 *or* 1, *we can solve* SSFOP *in $O(c \lg c)$ time.*

*Proof.* The only interesting case is separation 1, and the feasible offsets correspond to the zero entries in the convolution of the upper and lower extensions. The convolution can be computed in $O(c \lg c)$ time by using the Fast Fourier Transform method. (See [5], for example.) $\square$

LEMMA 2.7. *If all extensions of the lower contour are at most $h$, we can solve* SSFOP *in $O(hc \lg c)$ time.*

*Proof.* From Lemma 2.2, SSFOP$(s, L \cup U) = \bigcap_{i=1}^{h}$ SSFOP$(s, L_i \cup U)$, where $L_i$ is the subset of the lower contour with extension $i$. We can now solve SSFOP$(s, L_i \cup U)$ using Lemma 2.6 after assigning 1 to the lower extensions in $L_i$ and those upper extensions exceeding $s - i$ and 0 to the other extensions. Since we have a total of $h$ problems, each solvable in $O(c \lg c)$ time, the total time is $O(hc \lg c)$. $\square$

Now we can provide an overall solution to SSFOP by combining our results for contours with large extensions and contours with small extensions.

THEOREM 2.8. SSFOP *can be solved in $O(c\sqrt{n \lg c})$ time.*

*Proof.* SSFOP$(s, L \cup U)$ = SSFOP$(s, L_l \cup U) \cap$ SSFOP$(s, L_g \cup U)$, where $L_l$ is the subset of $L$ with extensions less than $h$ and $L_g$ is the subset of $L$ with extensions greater than or equal to $h$. Using Lemmas 2.7 and 2.5 with $h = \sqrt{n/\lg c}$, the solution time is $O(c\sqrt{n \lg c})$. $\square$

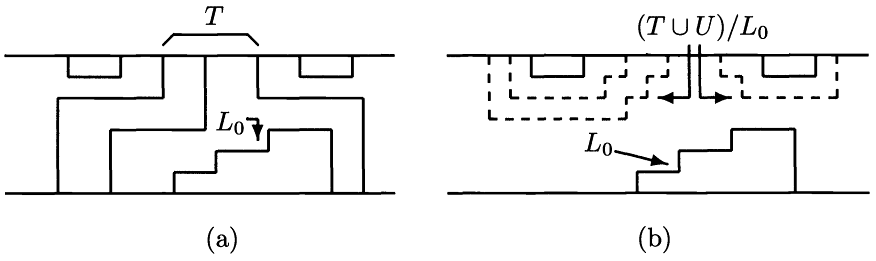


FIG. 3. *The effect of two-sided nets in* (a) *is incorporated into the top contour in* (b). *In this figure,* $L_0$ *is a monotonic portion of the lower contour.*

We can adapt the *halving* technique of [11] to solve SSOOP in the same time as SSFOP. The details will be shown in §3. The following theorem is just a simplified version of Theorem 3.8:

THEOREM 2.9. SSOOP *can be solved in* $O(c\sqrt{n}\lg c)$ *time.* □

COROLLARY 2.10. SSFOP *and* SSOOP *can be solved in* $O(n^{1.5}\sqrt{\lg n})$ *time for* $c = O(n)$. □

**3. General single-layer channel.** In this section, we use the ideas of §2 to solve FOP and OOP when there are two-sided as well as single-sided nets. As before, we begin by computing the contours of the upper and lower nets. Also as before, we consider separately the portions of the lower contour with large extensions and the portions with small extensions and then show how to put these ideas together. But first we consider an intermediate case, that is, when there are both single-sided and two-sided nets but all the single-sided nets are on one side.

LEMMA 3.1. *When all single-sided nets are on one side,* FOP *and* OOP *can be solved in* $O(n)$ *time.*

*Proof.* When the single-sided nets are on one side, we can extend the method of Mirzaian [11]. The basic idea is that as in river routing, the feasible offsets at a given separation form a continuous range whose bounds are determined by $O(n)$ *cut conditions.* More details can be found in [9]. □

To deal with the extra complications of two-sided nets, we also must introduce two new definitions.

First, let $L_0$ be a subset of the contour of the lower nets and $T$ a set of two-sided nets whose lower terminals are to the left of $L_0$. Define $T/L_0$ as the upper contour obtained by pulling up the lower terminals of the nets in $T$ and reconnecting them to the upper side to the left of preexisting terminals. That is, we convert the nets in $T$ to single-sided nets without violating planarity and without moving what were their lower terminals to the wrong side of $L_0$. This notation is also used analogously for any set $A$ of upper and two-sided nets as long as the lower terminal of each two-sided net is to the left or right of all nonzero extensions in $L_0$. In all cases, $A/L_0$ is the upper contour formed by moving lower terminals in $T$ away from $L_0$ and to the upper side. Finally, the notation can also be used with a portion of the upper contour, in which case "upper" and "lower" are reversed throughout the definition. Figure 3 illustrates the definition of $(T \cup U)/L_0$.

For the second definition, let $M$ be a subset of the contour of the upper or lower nets. We define $M|_s$ to be a new contour in which we replace all extensions exceeding $s - 1$ with extension $s - 1$.

We now proceed in the next two lemmas to handle a portion of lower contour with only large extensions. As before, the first lemma shows how to handle a monotonic piece of lower contour, and the second lemma handles a contour portion with large extensions by dividing it into maximal monotonic pieces.

LEMMA 3.2. *Let $A$ be a set of upper and two-sided nets. Then we can solve* FOP$(s, L_m \cup A)$ *in $O(n)$ time, where $L_m$ is a monotonic portion of the lower contour.*

*Proof.* The solution is the intersection of the feasible offsets from two subproblems. In the first subproblem, we solve FOP without $L_m$ (using Lemma 3.1). In the second subproblem, we retain $L_m$ and reroute the two-sided nets in the fashion shown in Fig. 3, i.e., we determine $(U \cup T)/L_m$. Since we have already determined the infeasible offsets in the absence of $L_m$ (in the first subproblem), we now ignore those portions of $(U \cup T)/L_m$ with extension exceeding $s-1$; we need only determine those offsets for which a vertical cut through $(U \cup T)/L_m$ *and* $L_m$ has too large a sum of upper and lower extensions. So the second subproblem is SSFOP$(s, L_m \cup ((U \cup T)/L_m)|_s)$, which can be constructed and solved in $O(n)$ time by Lemmas 2.1 and 2.3. □

Now we combine Lemmas 2.2, 3.2, and 2.4 to solve FOP for a subset of $L$ with large extensions. As before, we define large as exceeding $h$ and specify the value of $h$ later.

LEMMA 3.3. *If $L_g$ is a subset of $L$ containing only extensions greater than or equal to $h$, then we can solve* FOP$(s, L_g \cup U \cup T)$ *in $O(cn/h)$ time.* □

Now that we have taken care of FOP with large extensions, we use the next two lemmas to deal with small extensions. The next lemma tells us how to transform certain instances of FOP into SSFOP and will be used in handling general instances of FOP with small extensions.

LEMMA 3.4. *Let $T_l$ and $T_r$ be two sets of two-sided nets such that all the nets in $T_l$ are to the left of those in $T_r$. (That is, the upper terminals in $T_l$ are to the left of those in $T_r$ and similarly for the lower terminals.) Also let $U_l$ be a set of upper nets in which all terminals are to the left of (the upper terminals of) $T_r$, and let $L_r$ be a set of lower nets in which all terminals are to the right of $T_l$. (See Fig. 4.) Then*

$$\begin{aligned}\mathrm{FOP}(s, U_l \cup T_l \cup T_r \cup L_r) = {}& \mathrm{FOP}(s, U_l \cup T_l \cup T_r) \cap \mathrm{FOP}(s, L_r \cup T_l \cup T_r) \\ & \cap \mathrm{FOP}(s, ((U_l \cup T_l)/L_r)|_s \cup ((L_r \cup T_r)/U_l)|_s)\ .\end{aligned}$$

*Proof.* The argument is similar to the one for Lemma 3.2. At any given offset that is infeasible, either there is a vertical cut demonstrating infeasibility that goes through both $U_l$ and $L_r$ or there is not. In the former case, we know that we can incorporate the effect of the two-sided nets into the upper and lower contours; i.e., solving FOP$(s, ((U_l \cup T_l)/L_r)|_s \cup ((L_r \cup T_r)/U_l)|_s)$, as illustrated in Fig. 4, will rule out the infeasible offsets of the first type. On the other hand, if the infeasibility does not result from interaction between $U_l$ and $L_r$, it suffices to solve FOP$(s, U_l \cup T_l \cup T_r)$ and FOP$(s, L_r \cup T_l \cup T_r)$. Thus, intersection of the feasible offsets from these three problems provides the feasible offsets for the original problem. □

We can now solve FOP with small extensions.

LEMMA 3.5. *If the extensions of the upper and lower contours are all less than $h$, then we can solve* FOP *in $O(hc \lg^2 c)$ time.*

*Proof.* Let $t$ be the number of two-sided nets. We first consider the case when $s < 4h$. Divide the channel into $t/4h$ blocks $B_1, B_2, \ldots, B_{t/4h}$, each spanning $4h$ two-sided nets as shown in Fig. 5. Let $L_i, U_i$, and $T_i$ denote the lower nets, upper nets, and two-sided nets in block $i$. (Single-sided nets at a boundary between blocks of two-sided nets are assigned to exactly one of those blocks.) Since the upper side and

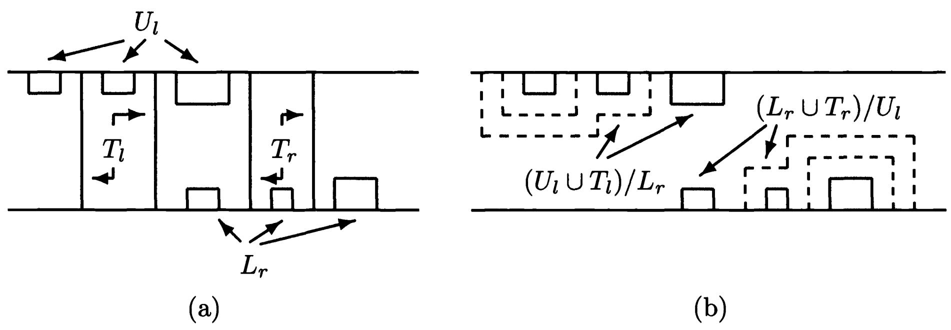


FIG. 4. *The effect of two-sided nets in* (a) *is incorporated into the upper and lower contours in* (b).

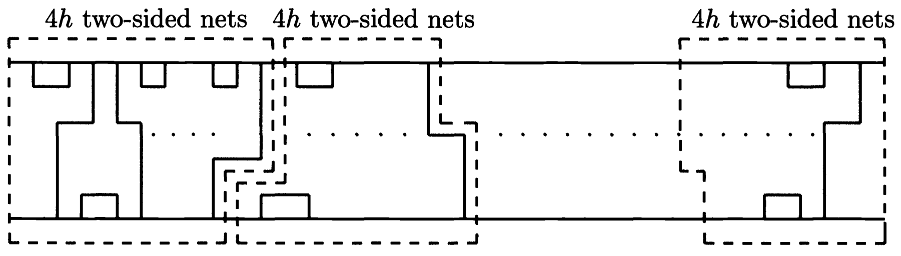


FIG. 5. *The partition for $s < 4h$.*

lower side of a block may not be of the same length, we define $c_i$ to be the sum of the number of columns spanned by the upper side and the number of columns spanned by the lower side.

From Lemma 2.2, $\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/4h} \mathrm{FOP}(s, L_i \cup T \cup U)$. Since $s < 4h$, there must be fewer than $4h$ two-sided nets through any vertical cut at any feasible offset. Therefore, any offset with vertical cuts through $L_i$ and $U_j$ for $j > i + 1$ or $j < i - 1$ would be an infeasible offset, because such a cut would be crossed by all the nets in $T_{i+1}$ or $T_{i-1}$. Thus we can write

$$\mathrm{FOP}(s,N) = \bigcap_{i=1}^{t/4h} [\mathrm{FOP}(s, L_i \cup T \cup U_i) \cap \mathrm{FOP}(s, L_i \cup T \cup U_{i+1}) \cap \mathrm{FOP}(s, L_i \cup T \cup U_{i-1})] .$$

Note also that no vertical cuts through both $L_i$ and $U_j$ can cut any two-sided nets outside blocks $i$ through $j$, so we can rewrite $\mathrm{FOP}(s, N)$ as

$$\bigcap_{i=1}^{t/4h} [\mathrm{FOP}(s, L_i \cup T_i \cup U_i) \cap \mathrm{FOP}(s, L_i \cup T_i \cup T_{i+1} \cup U_{i+1}) \cap \mathrm{FOP}(s, L_i \cup T_{i-1} \cup T_i \cup U_{i-1})] .$$

Now we can solve each of $\mathrm{FOP}(s, L_i \cup T_i \cup T_{i+1} \cup U_{i+1})$ and $\mathrm{FOP}(s, L_i \cup T_{i-1} \cup T_i \cup U_{i-1})$ in time $O(h(c_{i-1} + c_i + c_{i+1}) \lg(c_{i-1} + c_i + c_{i+1}))$ as follows. We use Lemma 3.4 to decompose the problem further, Lemma 2.1 for the computation of new contours (which will still have $O(h)$ extensions), and Lemmas 3.1 and 2.7 to solve the subproblems.

To solve $\mathrm{FOP}(s, L_i \cup T_i \cup U_i)$, we use a recursive method, for which we consider the general problem of solving $\mathrm{FOP}(s, L^* \cup T^* \cup U^*)$ with $|T^*| = t^* \leq 4h$. We decompose such a problem into left and right blocks, each having half as many two-sided nets as the original. Using subscripts $l$ and $r$ to denote the portions of $L^*$, $T^*$, and $U^*$ falling in the left and right blocks, we know from Lemma 2.2 that

$$\begin{aligned}\mathrm{FOP}(s, L^* \cup T^* \cup U^*) &= \mathrm{FOP}(s, L_l^* \cup T^* \cup U^*) \cap \mathrm{FOP}(s, L_r^* \cup T^* \cup U^*) \\ &= \mathrm{FOP}(s, L_l^* \cup T_l^* \cup U_l^*) \cap \mathrm{FOP}(s, L_l^* \cup T^* \cup U_r^*) \\ &\quad \cap \mathrm{FOP}(s, L_r^* \cup T^* \cup U_l^*) \cap \mathrm{FOP}(s, L_r^* \cup T_r^* \cup U_r^*).\end{aligned}$$

The restrictions from $T^*$ to $T_l^*$ and $T_r^*$ are determined by reasoning similar to that used above. Also as above, we use Lemma 3.4, Lemma 2.1, Lemma 3.1, and Lemma 2.7 to solve $\mathrm{FOP}(s, L_l^* \cup T^* \cup U_r^*)$ and $\mathrm{FOP}(s, L_r^* \cup T^* \cup U_l^*)$ in time $O(hc^* \lg c^*)$, where $c^*$ is the sum of the number of top and bottom columns spanned by $L^*$, $T^*$, and $U^*$. $\mathrm{FOP}(s, L_l^* \cup T_l^* \cup U_l^*)$ and $\mathrm{FOP}(s, L_r^* \cup T_r^* \cup U_r^*)$ are just recursive calls; we denote the sum of the number of top and bottom columns in these subproblems as $c_l^*$ and $c_r^*$. Then, the time $T(t^*, c^*)$ to compute $\mathrm{FOP}(s, L^* \cup T^* \cup U^*)$ can be written as

$$T(t^*, c^*) = T(t^*/2, c_l^*) + T(t^*/2, c_r^*) + O(hc^* \lg c^*)$$

for all $t^* \leq 4h$, where $c_l^* + c_r^* = c^*$, and $T(1, c) = O(hc \lg c)$. There are $O(\lg t^*)$ stages in the recursion, and the total work on all subproblems at any given stage is $O(hc^* \lg c^*)$ time. Thus, $T(t^*, c^*) = O(hc^* \lg c^* \lg t^*)$, and, in particular, $T(4h, c_i) = O(hc_i \lg c_i \lg h) = O(hc_i \lg^2 c)$.

Putting everything together, the time to solve $\mathrm{FOP}(s, N)$ is

$$\begin{aligned}T(t, c) &= \sum_{i=1}^{t/4h} T(4h, c_i) + \sum_{i=1}^{t/4h} O((c_{i-1} + c_i + c_{i+1})h \lg c) \\ &= \sum_{i=1}^{t/4h} O(hc_i \lg^2 c) + O(hc \lg c) \\ &= O(hc \lg^2 c) + O(hc \lg c) \\ &= O(hc \lg^2 c).\end{aligned}$$

Finally, we must return to solving the original problem in the case that $s \geq 4h$. We divide the channel into $t/2h$ blocks, each spanning $2h$ two-sided nets. From Lemma 2.2, $\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/2h} \mathrm{FOP}(s, L_i \cup T \cup U)$. Furthermore, at any offset, we need not consider any vertical cut for which the number of two-sided nets crossing the cut is less than $s - 2h$ or greater than $s$. In the former case, we know the cut cannot provide evidence of infeasibility; in the latter case infeasibility is guaranteed. Thus, we can write

$$\begin{aligned}\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/2h} [&\mathrm{FOP}(s, L_i \cup T_i \cup T_{i+1} \cup \cdots \cup T_{i+s/2h} \cup U_{i+s/2h-1} \cup U_{i+s/2h}) \\ &\cap \mathrm{FOP}(s, L_i \cup T_i \cup T_{i-1} \cup \cdots \cup T_{i-s/2h} \cup U_{i-s/2h+1} \cup U_{i-s/2h})].\end{aligned}$$

At this point, we could proceed as when $s < 4h$ by incorporating two-sided nets into the upper and lower contours, but there are too many two-sided nets to get the desired running time; we might end up with more than $O(h)$ distinct extensions in the contours. Instead, we take out the $s - 4h$ two-sided nets between $L_i$ and $U_{i+s/2h-1}$ and the $s - 4h$ two-sided nets between $L_i$ and $U_{i-s/2h+1}$, and we decrease $s$ by $s - 4h$.

Each infeasible offset in this new problem actually denotes the center of a range of $2(s-4h)+1$ infeasible offsets of the original problem, but with this proviso, the task is to solve

$$\bigcap_{i=1}^{t/2h} [\mathrm{FOP}(4h, L_i \cup T_i \cup T_{i+s/2h-1} \cup T_{i+s/2h} \cup U_{i+s/2h-1} \cup U_{i+s/2h}) \cap \mathrm{FOP}(4h, L_i \cup T_i \cup T_{i-s/2h+1} \cup T_{i-s/2h} \cup U_{i-s/2h+1} \cup U_{i-s/2h})].$$

We can solve these subproblems using Lemmas 3.4, 2.1, 3.1, and 2.7 as before. Also, with a similar analysis for the combined running time of the subproblems, we get a total time of $O(hc \lg c)$. □

THEOREM 3.6. *FOP can be solved in* $O(c\sqrt{n} \lg c)$ *time.*

*Proof.* We can use Lemma 2.2 to write $\mathrm{FOP}(s, N) = \mathrm{FOP}(s, L_l \cup T \cup U_l) \cap \mathrm{FOP}(s, L_g \cup T \cup U) \cap \mathrm{FOP}(s, U_g \cup T \cup L)$, where $L_l$ is the subset of $L$ with extensions less than $h$ and $L_g$ is the subset of $L$ with extensions greater than or equal to $h$, and similarly for $U_l$ and $U_g$. The first subproblem can be solved in $O(hc \lg^2 c)$ time using Lemma 3.5, and the latter two subproblems can be solved in $O(cn/h)$ time using Lemma 3.3. By letting $h = \sqrt{n}/\lg c$, FOP can be solved in $O(c\sqrt{n} \lg c)$ time. □

We now show how to use a halving technique similar to that of [11] and [9] to solve OOP in the same time as FOP. We actually focus here on finding optsep($P$), the minimum separation attainable with an optimal offset for the routing problem $P$; once optsep($P$) is determined, the solution of the feasible offset problem can be used to determine the optimal offsets. From the original problem $P$, we create a simpler problem $P^e$ that has about half the separation of $P$. The basic idea is to halve the extensions of the contours of single-sided nets, remove every other two-sided net, and compact the channel horizontally to eliminate the freed space. More precisely, if the two-sided nets are numbered 1 through $t$ from left to right, we remove all the odd-numbered nets and move the terminals of net $2i$ to the left by $i$ units. The nonzero portions of the single-sided contours are also shifted left so that they stay the same distance from their nearest two-sided nets. This is the same transformation used in [9], but the effect on optsep is slightly different here due to the general arrangement of single-sided nets and the disallowance of routing on the channel boundaries, and the timing analysis for computing optsep($P$) differs more substantially. The following lemma states the relationship between optsep($P$) and optsep($P^e$).

LEMMA 3.7. *Let* $s = \mathrm{optsep}(P)$ *and* $s^e = \mathrm{optsep}(P^e)$. *Then* $2s^e - 2 \le s \le 2s^e + 2$.

*Proof.* We again use the theory of single-layer routability from [10]. In our context, the *flow* of a cut is the number of nets that *must* cross it, and the cut is *safe* if its flow is no greater than one less than the maximum of the horizontal and vertical extents of the cut. A cut $\chi$ in $P$ that crosses $f$ nets, $p$ of which are lower nets, $q$ of which are two-sided nets, and $r$ of which are upper nets, can be seen to correspond to a cut $\chi^e$ with the following properties: (1) The flow of $\chi^e$ is in the range $[\frac{p-1}{2} + \frac{q-1}{2} + \frac{r-1}{2}, \frac{p}{2} + \frac{q+1}{2} + \frac{r}{2}] = [\frac{f}{2} - \frac{3}{2}, \frac{f}{2} + \frac{1}{2}]$, and (2) the horizontal extent of $\chi^e$ is diminished relative to $\chi$ to the same extent as the flow. Thus $s^e - 1 \in [\frac{s-1}{2} - \frac{3}{2}, \frac{s-1}{2} + \frac{1}{2}]$, i.e., $2s^e \in [s-2, s+2]$. □

THEOREM 3.8. *OOP can be solved in* $O(c\sqrt{n} \lg c)$ *time.*

*Proof.* To find optsep($P$), we recursively find $s^e = \mathrm{optsep}(P^e)$. Then we need only determine which of the five separations in $[2s^e - 2, 2s^e + 2]$ have feasible offsets for $P$. Let $T(m)$ be the solution time for optsep($P$) where $P$ has maximum extension $m$ but has been derived by repeated application of the halving transformation to a problem with maximum extension $M$. Then any extension $h$ in $P$ has been derived from an extension $hM/m$ in the original problem. An argument as in the proof of Theorem 3.6 then tells us that we can solve FOP for $P$ in $O(\sqrt{m}\sqrt{n/M}c \lg c)$ time. Thus we have $T(m) = T(m/2) + \sqrt{m}\sqrt{n/M}c \lg c$, which yields $T(m) = \sqrt{m}\sqrt{n/M}c \lg c \leq \sqrt{n}c \lg c$. $\square$

COROLLARY 3.9. FOP *and* OOP *can be solved in* $O(n^{1.5} \lg n)$ *time for* $c = O(n)$. $\square$

**4. Conclusion and further results.** We have shown how to solve the feasible offset and optimal offset problems for single-layer channel routing in time $O(c\sqrt{n} \lg c)$. (The time is $O(c\sqrt{n \lg c})$ when all nets are single sided and $O(n)$ when all single-sided nets are on one side of the channel.) This result is unattractive for large values of $c$, but there is a superior alternative when $c$ is larger than $n^{1.5}$. Essentially all the necessary machinery is already in place for the following result, which states that FOP can be solved in $O(n^2 \lg n)$ time independent of the number of columns.

THEOREM 4.1. FOP *can be solved in* $O(n^2 \lg n)$ *time.*

*Proof.* Using Lemma 2.2, we decompose the lower contour into maximal monotonic subsets. Since there are only $n$ nets, we have at most $n$ monotonic subsets. We can find the feasible offsets for each subset in $O(n)$ time using Lemma 3.2. The total time required to find the feasible offsets for all of the subsets of the lower contour is $O(n^2)$. Furthermore, for each subset, the set of feasible offsets can be output as a list of at most $n$ nonoverlapping intervals with all the interval endpoints in sorted order. Two sets of nonoverlapping intervals with endpoints in sorted order can be intersected in time proportional to the total number of intervals, which is an upper bound on the output size. We intersect the $O(n)$ sets of intervals in a tournament style, i.e., we go from $n$ sets with $n$ intervals in each set to $n/2$ sets with $2n$ intervals in each set, ..., to 1 set with $n^2$ intervals. There are $\lg n$ stages, with $O(n^2)$ work at each stage, yielding a total time of $O(n^2 \lg n)$. $\square$

One direction for further research is to improve the time for feasible offset when the number of columns is large. We know that $\Omega(n^2)$ is a lower bound on the worst-case running time, but we suspect that it may not be difficult to obtain an $O(n^2)$ upper bound as well. Another remaining open question is whether our upper bounds for feasible offset with smaller $c$ can be improved. We know of no nontrivial lower bounds, i.e., better than $\Omega(\min\{c, n^2\})$. It also might be possible to improve the time required to solve optimal offset without making further progress on feasible offset. Though it seems unlikely that optimal offset would be much easier than feasible offset, optimal offset has a much smaller output size, and output size is the only basis for our lower bounds on feasible offset.

It is also desirable to handle the situation in which there are multiple modules. Within each module, the positions of the terminals are fixed, but on each side of the channel the modules can slide back and forth as long as their order does not change. In the *optimal placement problem*, the goal is to minimize the channel length given a channel width. We can solve this problem in $O(n^3)$ time by adapting ideas used by Chao and LaPaugh [3] for density minimization; more details can be found in [13]. A further direction for research is to improve this $O(n^3)$ time when there is a reasonable bound on the number of columns.

Finally, an interesting open problem related to SSFOP is efficient computation of the max, + convolution. The technique in Lemma 2.7 can be extended to yield a solution to max, + convolution for $n$-vectors of small integers (e.g., $\leq n^{1/4}$) in less than $\Theta(n^2)$ time. If the range of integers could be extended to 1 through $n$, improved solutions for several VLSI routing problems would result (e.g., see [11]).

## REFERENCES

[1] B. S. Baker and R. Y. Pinter, *An algorithm for the optimal placement and routing of a circuit within a ring of pads*, in 24th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Piscataway, NJ, 1983, pp. 360–370.

[2] S.-C. Chang, J. JáJá, and K. W. Ryu, *Optimal Parallel Algorithms for One-Layer Routing*, Tech. report UMIACS-TR-89-46, University of Maryland, Institute for Advanced Computer Studies, College Park, MD, April 1989.

[3] L.-F. Chao and A. S. LaPaugh, *Finding All Minimal Shapes in a Routing Channel*, Tech. report CS-TR-384-92, Princeton University, Department of Computer Science, Princeton, NJ, August 1992.

[4] R. Cole and A. Siegel, *River routing every which way, but loose*, in 25th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Piscataway, NJ, 1984, pp. 65–73.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, McGraw-Hill, New York, 1990.

[6] D. Dolev, K. Karplus, A. Siegel, A. Strong, and J. D. Ullman, *Optimal algorithms for structural assembly*, VLSI Design (1982), pp. 38–43. Earlier version in Proc. of the 13th ACM Symposium on Theory of Computing, ACM, New York, 1981.

[7] R. I. Greenberg, A. T. Ishii, and A. L. Sangiovanni-Vincentelli, *MulCh: A multi-layer channel router using one, two, and three layer partitions*, in IEEE International Conference on Computer-Aided Design (ICCAD-88), IEEE Computer Society Press, Piscataway, NJ, 1988, pp. 88–91.

[8] R. I. Greenberg and F. M. Maley, *Minimum separation for single-layer channel routing*, Inform. Process. Lett., 43 (1992), pp. 201–205.

[9] R. I. Greenberg and J.-D. Shih, *Single-Layer Channel Routing and Placement with Single-Sided Nets on One Side*, Tech. report UMIACS-TR-93-6, University of Maryland, Institute for Advanced Computer Studies, College Park, MD, January 1993; Revised version, submitted.

[10] F. M. Maley, *Single-Layer Wire Routing and Compaction*, MIT Press, Cambridge, MA, 1990.

[11] A. Mirzaian, *River routing in VLSI*, J. Comput. System Sci., 34 (1987), pp. 43–54.

[12] R. Y. Pinter, *The Impact of Layer Assignment Methods on Layout Algorithms for Integrated Circuits*, Ph.D. thesis and report MIT/LCS/TR-291, Department of Electrical Engineering & Computer Science, Massachusetts Institute of Technology, Cambridge, MA, August 1982.

[13] J.-D. Shih, *Efficient Algorithms for Channel Routing and Placement Problems*, Ph.D. thesis, University of Maryland, Electrical Engineering Department, College Park, MD, 1993.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 555–605, November 1995

# CANONICAL ENCODERS FOR SLIDING BLOCK DECODERS*

JONATHAN ASHLEY† AND BRIAN MARCUS†

**Abstract.** The existence and uniqueness of a canonical minimal encoder for any given sliding block decoder are proven. The structure of this encoder is given in terms of an explicit sequence of state splittings. The universality of the state splitting algorithm for code construction is clarified.

**Key words.** sliding block code, state splitting, symbolic dynamics

**AMS subject classifications.** 68R10, 94A24

## 1. Introduction.

Consider a finite directed graph whose edges are labeled by a finite alphabet. We call such an object a labeled graph. Now, together with a given subset of initial states and a given subset of terminal states, a labeled graph presents a regular language, defined as the set of all blocks obtained by reading the labels of paths which begin at an initial state and end at a terminal state. When a labeled graph is viewed as describing a regular language, it is usually called an automaton.

A labeled graph also presents a sofic shift, which is defined as the set of biinfinite sequences obtained by reading the labels of biinfinite paths in the graph. When viewed in this way, a labeled graph is usually called a presentation.

Regular languages form a prominent class of languages that is studied in automata theory. Sofic shifts form a prominent class of systems that is studied in symbolic dynamics. As can be seen just from the definitions, there is a strong connection between regular languages and sofic shifts. And many results for regular languages have analogues for sofic shifts.

The best-known example of this is the uniqueness of minimal presentations. These are fundamental results which show how regular langauages and sofic shifts can be presented in a canonical and minimal way.

For regular languages the result is as follows: Every regular language has a unique minimal deterministic (i.e., at each state, the outgoing edges are labeled distinctly) automaton; here, minimality is taken either in terms of number of states or in a functorial sense (see [17]).

The analogous result for sofic shifts would be that every sofic shift has a unique minimal deterministic presentation; this is not quite true. However, sofic shifts can, in some sense, be broken down into irreducible pieces, and it is true that every every irreducible sofic shift has a unique minimal deterministic presentation; again, minimality can be taken either in terms of number of states or in a functorial sense ([13]).

These minimality results (one for regular languages and the other for sofic shifts) are closely related, but neither implies the other.

It is natural to wonder if there is a notion of deterministic mapping between regular languages or deterministic mapping between sofic shifts, and if such mappings can also be presented in a canonical, minimal way. Indeed, for mappings between regular languages, this was done some time ago (see [23, p. 290]). By considering the identity mapping on a regular language, this result generalizes the minimality result, mentioned above, for regular languages.

* Received by the editors February 16, 1993; accepted for publication (in revised form) August 18, 1994.

† IBM Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099 (jashley@almaden.ibm.com, marcus@almaden.ibm.com).

We offer the notion of a sliding block decoder as the notion of deterministic mapping between sofic shifts. A sliding block decoder is a special kind of sliding block code; the latter is a mapping from one sofic shift into another such that each symbol in the image sequence is a function of only a fixed number $m$ (the memory) of past symbols, the present symbol, and a fixed number $a$ (the anticipation) of future symbols in the domain sequence, and this function does not change with time. In §6, we define a sliding block decoder as a sliding block code that can be presented by what we call a finite-state code or encoder (finite-state codes belong to a class of objects known in automata theory as transducers). In §7, we give intrinsic and other equivalent definitions of sliding block decoders.

In Theorem 6.2, we show that every sliding block decoder on an irreducible sofic shift has a canonical, minimal encoder which is unique in a strong functorial sense. And, again by considering the identity map, this result generalizes the fact that each irreducible sofic shift has a unique minimal presentation. The canonical encoder comes from a general construction for sliding block codes that is given in §5. In §8, we describe how our canonical encoders can be constructed using specific splitting and amalgamation operations on graphs. As part of our development in §8, we show (Lemma 8.5) how any 1-block conjugacy (i.e., invertible sliding block code) between two graph shifts (special kinds of sofic shifts) can be canonically decomposed into splitting and amalgamation operations. This result may be of independent interest within symbolic dynamics.

Aside from the connection with automata theory, there is a strong connection between sofic shifts and the theory of constrained coding, a subfield of coding theory. A typical problem in constrained coding is that of encoding a user's digital data stream into another digital data stream tailored to a channel across which it is to be transmitted. No assumptions are made about the user's data stream other than it is binary or ternary or $k$-ary. It is the encoder's job to encode the user's arbitrary stream of symbols into a stream of symbols conforming to specific constraints. We assume that the constrained sequences, into which the user's stream is encoded, lie in some sofic shift $W$. The constraints are imposed either because of the physical limitations of the channel or in order to improve the reliability of the transmission process. For instance, $W$ might be the $(d, k)$ run-length limited sofic shift: between any two consecutive 1's, there must be at least $d$ and at most $k$ 0's. Such a constraint arises in magnetic and optical data storage channels where very short runs of 0's and very long runs of 0's are prone to error (see [18], [33]). Or $W$ might be an error-correcting code i.e., a set of binary sequences such that any two distinct sequences differ in a "large" number of coordinates; so, if a "small" number of errors are made, then a sequence in $W$ cannot be corrupted to look like a different sequence in $W$ (see [29]).

In these practical applications, the encoding is done via a finite-state code. The finite-state code is used as a finite-state machine to encode information in a symbol-by-symbol manner. One can also use the finite-state code to decode, but the decoding is done with bounded delay—not symbol-by-symbol (see the definition of finite-state code in §6). Finite-state codes can be implemented in hardware.

Now, if the noise in the channel causes an error in the encoded information, then because of the state-dependence of the decoding, such an error could propagate indefinitely. For this reason, it is desirable that the decoding be implemented via a sliding block code; by definition, such a code is a sliding block decoder. As a consequence, any particular constrained-stream symbol figures into the computation of only a bounded block of user symbols; so, an error caused by the channel gives rise to only a bounded burst of errors in the user's sequence.

Given a sofic shift $W$ and a positive integer $k$, a necessary and sufficient condition for the existence of a finite-state code which encodes arbitrary $k$-ary data to sequences in $W$ is the entropy inequality: $h(W) \geq \log(k)$; here, $h(W)$ denotes the *entropy* of $W$, i.e., the asymptotic growth rate of the number of $n$-blocks that appear in $W$; this is a consequence of the state-splitting algorithm ([1]); see §9.

For finite-state codes with sliding block decoders, stronger assumptions are needed. It is sufficient that $h(W) > \log(k)$ or that $W$ be a shift of finite type (a special kind of sofic shift) and $h(W) \geq \log(k)$ (see [1], [20], [26]). Methods for constructing such codes are contained in [1], [9], [14], [18], [28], among many others.

The state-splitting algorithm requires the choice of a deterministic presentation and an auxiliary vector, called an approximate eigenvector; see, for example, the formulation given in [28, §E].

If we allow arbitrary choices in both the presentation and approximate eigenvector, then Corollary 11.7 shows that the algorithm is strong enough to find every sliding block decoder. This is a broad and not-so-algorithmic interpretation of the algorithm. See the discussion after Corollary 11.7.

If we fix the presentation to be the minimal deterministic presentation but allow arbitrary choice in the approximate eigenvector, then Corollary 12.2 shows that, up to a change in the domain of the decoder and a shift of the decoding function (but no change in the decoding function itself), the algorithm is strong enough to find every sliding block decoder. See the discussion after Corollary 12.2. In particular, it will find the sliding block decoder with smallest window length; but Example 11.4 shows that, in some cases, it will not find the sliding block decoder with smallest number of encoder states ("state expansion" and "in-splitting" are additional necessary tools).

The narrowest and most algorithmic interpretation of the state-splitting algorithm fixes the presentation to be the minimal deterministic presentation and the approximate eigenvector to be a smallest such vector (in terms of maximal component). Example 10.1 shows that, with this intrepretation, the algorithm need not find the sliding block decoder with minimal window length. This is the first such example we know of where $W$ is a shift of finite type and $h(W) = \log(k)$; examples of this phenomenon, where $h(W) > \log(k)$, were found in [19] and [18].

Sections 2–4 provide most of the necessary background from symbolic dynamics. There may be some overlap between our work and that of Hollmann ([16]).

**2. Background.** In this section, we review definitions of some of the basic concepts in symbolic dynamics.

The *full shift* over a finite alphabet $\mathcal{A}$ is the set of all biinfinite sequences over $\mathcal{A}$. A *block* (over $\mathcal{A}$) is a finite sequence of symbols (in $\mathcal{A}$). An *$n$-block* is a block of length $n$. The *shift map* $\sigma$ is defined by $\sigma(x) = y$, where $y_i = x_{i+1}$. For an element $x = \ldots x_{-1}x_0x_1\ldots$ in a full shift, $x_{[m,n]}$ denotes the block which appears in coordinates $m$ through $n$:

$$x_{[m,n]} \equiv x_m \cdots x_n.$$

And

$$x_{[n,\infty)} \equiv x_n x_{n+1} \cdots, \qquad x_{(-\infty,n]} \equiv \cdots x_{n-1}x_n.$$

The *full $k$-shift* is the full shift over $\mathcal{A} \equiv \{0, \ldots, k-1\}$.

A *shift space* or a *shift* is a shift-invariant subset of a full shift obtained by forbidding the appearance of a collection of blocks.

A *shift of finite type* (abbreviated SFT) is a shift space obtained by forbidding the appearance of a finite collection of blocks.

By a *graph* we mean a finite directed graph. For a graph $G$, we write $G = (\mathcal{V}(G), \mathcal{E}(G))$, where $\mathcal{V}(G)$ is the set of vertices (sometimes called states) and $\mathcal{E}(G)$ is the set of edges. We write a finite path as a sequence of edges

$$u = e_1 \dots e_n.$$

$s(u)$ denotes the initial state of $u$ and $t(u)$ denotes the terminal state of $u$.

A *graph homomorphism* from a graph $G$ to a graph $H$ is a pair of mappings, the *state mapping*

$$L^* : \mathcal{V}(G) \to \mathcal{V}(H),$$

and the *edge mapping*

$$L : \mathcal{E}(G) \to \mathcal{E}(H)$$

that are consistent in the sense that initial states and terminal states of edges are preserved. We often refer to $L$ itself, rather than the pair $(L^*, L)$, as a graph homomorphism. This makes sense because if $L$ is the edge mapping of a graph homomorphism, then $L$ determines $L^*$. A graph homomorphism such that $L$ and $L^*$ are 1-1 is called a *graph isomorphism.*

The *graph shift*, sometimes called an edge shift, $X_G$ is the set of all biinfinite paths on $G$:

$$X_G \equiv \{\dots e_{-1}e_0e_1 \dots : e_i \in \mathcal{E}(G) \text{ and } e_{i+1} \text{ follows } e_i \text{ in } G\}.$$

We often regard graph shifts based on isomorphic graphs as being identical. Note that any graph shift is an SFT.

For a square nonnegative integral matrix $A$, we take $X_A$ to be a graph shift defined by a graph whose adjacency matrix is $A$. For instance, $X_{[k]}$ may be regarded as the full $k$-shift.

A *labeled graph* $(G, L)$ is a graph $G$ together with a labeling $L$ of its *edges.* A *sofic shift* $X$ is the set of all biinfinite sequences obtained by reading the labels of a labeled graph $(G, L)$:

$$X = \{\dots L(e_{-1})L(e_0)L(e_1) \dots \quad : \quad \dots e_{-1}e_0e_1 \dots \in X_G\}.$$

$(G, L)$ is called a *presentation* of $X$. Note that any graph shift is a sofic shift. In fact, it is not hard to see that any SFT is a sofic shift. A sofic shift which is not an SFT is called *strictly sofic.*

Let $X$ and $Y$ be shift spaces. Let $m, a$ be integers such that $m + a \geq 0$ and $\Phi$ a map from $(m+a+1)$-blocks of $X$ to 1-blocks (i.e., symbols) of $Y$. Let $\phi : X \to Y$ be the map defined by

$$\phi(x)_i = \Phi(x_{[i-m,i+a]}).$$

$\phi$ is called a *sliding block code* with *memory* $m$ and *anticipation* $a$. The *window length* is defined to be $m + a + 1$. We write

$$\phi = \Phi_\infty^{m,a}.$$

By an *$(m,a)$-block code*, we mean a sliding block code that can be expressed with memory $m$ and anticipation $a$. By an *$n$-block code* we mean a sliding block code that can be expressed with window length $n$. Unless otherwise specified, an $n$-block code $\phi$ is assumed to have memory 0, i.e., is a $(0, n-1)$-block code; when we write

$$\phi = \Phi_\infty,$$

we make this tacit assumption. In particular, unless otherwise specified, a 1-block code has memory 0 and anticipation 0.

Note that the memory, anticipation, and window length of a sliding block code are not well defined. For instance, a sliding block code with memory $m$ and anticipation $a$ can also be expressed as a sliding block code with memory $m$ and anticipation $a+1$. One might hope that the ambiguity could be removed if we were to restrict ourselves to a consideration of $(m,a)$ with minimum window length $m+a+1$. If $X$ is a graph shift, then this does the trick—see Proposition 7.2. There are a unique memory and anticipation which achieve the minimum window length; but even for SFT's, this does not hold in general (see [21]).

If $X$ is a shift space and $n$ is a positive integer, then the *higher block system* $X^{[n]}$ is the shift space obtained by breaking the sequences in $X$ into overlapping blocks of length $n$: the alphabet of $X^{[n]}$ is the set of $n$-blocks in $X$, and each sequence $x \in X$ gives rise to the sequence $\dots(x_{-1}\dots x_{n-2})(x_0\dots x_{n-1})(x_1\dots x_n)\dots \in X^{[n]}$.

Note that an $n$-block code on $X$ may be viewed as a 1-block code on $X^{[n]}$.

There is an analogous notion for graphs: $G^{[n]}$ denotes the graph whose states are paths of length $n-1$ in $G$ and edges are paths of length $n$ in $G$ with the same kind of overlapping condition as in $X^{[n]}$. There is a natural identification between the graph shift $X_{G^{[n]}}$ and $(X_G)^{[n]}$.

Observe that for a labeled graph $(G,L)$, $L_\infty$ is a 1-block code. In fact, every 1-block code on a graph shift is of this form. One special case of this is as follows. Suppose that $(G,L)$ is a presentation of a sofic subshift contained in a graph shift $X_H$. Then $L$ can be regarded as (the edge mapping of) a graph homomorphism: $(L^*, L) : (\mathcal{V}(G), \mathcal{E}(G)) \to (\mathcal{V}(H), \mathcal{E}(H))$.

A *factor map*, sometimes called a factor code, is a sliding block code which is onto. We say that the image is a *factor* of the domain and the domain is an *extension* of the image. An *imbedding* is a sliding block code which is 1-1. A *conjugacy* is a sliding block code which is 1-1 and onto. Conjugacies are the fundamental mappings of symbolic dynamics. If there is a conjugacy from one shift space $X$ to another shift space $Y$, then $X$ and $Y$ are forced to share many properties; for instance, if one is an SFT (resp., sofic), then so is the other. However, the property of being a graph shift is not preserved under conjugacy; in fact, it is well known that a shift space is an SFT if and only if it is conjugate to a graph shift.

**3. The structure of conjugacies betweeen graph shifts.** It is well known that every conjugacy between graph shifts can be broken down into conjugacies obtained by the *basic graph operations*: out-splitting; in-splitting; out-amalgamation; in-amalgamation; and graph isomorphism. We briefly review these operations.

An *out-splitting* $H$ of a graph $G$ is obtained as follows: for each state $I$ of $G$, partition the outgoing edges from $I$ into sets $\{P_I^1, \dots, P_I^{m_I}\}$. The graph $H$ has vertices $\{I^1, \dots, I^{m_I}\}$ and an edge, called $e^j$, from state $I^i$ to state $J^j$ for each edge $e \in P_I^i$ from $I$ to $J$. The states $I^i$ are called *descendants* of state $I$, and the edges $e^j$ are called *descendants* of edge $e$. The state $I^i$ and its outgoing edges in $H$ are shown in Fig. 1.

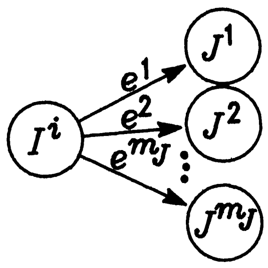


FIG. 1.

Note that an out-splitting can split several states simultaneously, each into several pieces (sometimes, an out-splitting is called a *round* of out-splitting to emphasize that several things are done at the same time).

And an *in-splitting* is defined by partitioning incoming edges rather than outgoing edges. When $H$ is an out-splitting of $G$, we say that $G$ is an *out-amalgamation* of $H$. When $H$ is an in-splitting of $G$, we say that $G$ is an *in-amalgamation* of $H$. We include graph isomorphism as a basic graph operation as well.

The notion of "state combining," as in [18], is a combination of in-splitting and out-amalgamation.

Each basic graph operation creates a graph $H$ from a graph $G$. Associated to such an operation is a *basic graph conjugacy* $X_G \to X_H$. For instance, if $H$ is an out-splitting of $G$, then the basic graph conjugacy is the 2-block code $\phi = \Phi_\infty : X_G \to X_H$ defined by

$$\Phi(ef) = e^j,$$

where $j$ is the index of the partition element to which $f$ belongs. Now, $G$ is an out-amalgamation of $H$ and the associated basic graph conjugacy is the 1-block code $\psi = \Psi_\infty : X_H \to X_G$ defined by

$$\Psi(e^j) = e.$$

The basic graph conjugacy associated to a graph isomorphism is simply a relabeling of the symbols of the shift; for our purposes, this is not very important, and we will often forget about it (on the other hand, it is extremely crucial in the theory of automorphisms of SFTs—see [34]).

We sometimes refer to the conjugacy associated with a basic graph operation as that basic graph operation itself.

The following is a cornerstone of symbolic dynamics, due to R. F. Williams.

THEOREM 3.1 ([35], [32, Chap. 5, §3]). *Every conjugacy between graph shifts is the composition of basic graph conjugacies.*

Now, let $(G, L)$ and $(H, M)$ be labeled graphs which present the same sofic shift $X$. We say that $(H, M)$ *is obtained from* $(G, L)$ *by basic graph operations* if there is a sequence of presentations

$$(G, L) = (G_0, L_0), (G_1, L_1), \dots, (G_n, L_n) = (H, M)$$

of $X$ and basic graph conjugacies

$$\psi_i : X_{G_i} \to X_{G_{i+1}}$$

$$\begin{tikzcd}
X_G = X_{G_0} \arrow[r, "\psi_0"] \arrow[d, "L_\infty = (L_0)_\infty"'] & X_{G_1} \arrow[r, "\psi_1"] \arrow[dl, "(L_1)_\infty"] & \cdots \arrow[r, "\psi_{n-1}"] & X_{G_n} = X_H \arrow[dlll, "(L_n)_\infty"] \\
X & & &
\end{tikzcd}$$

FIG. 2.

$$\begin{tikzcd}
X_G \arrow[rr, "\psi"] \arrow[dr, "L_\infty"'] & & X_H \arrow[dl, "M_\infty"] \\
& X &
\end{tikzcd}$$

FIG. 3.

such that the diagram in Fig. 2 commutes. Notice what happens to the labeling $L$ of $(G, L)$ if $\psi : X_G \to X_H$ is a splitting: the label of each edge in $G$ is *copied* to all of its descendants in $H$. Viewed the other way, this says that if $\psi^{-1} : X_H \to X_G$ is an amalgamation, then any two edges of $H$ amalgamating to the same edge of $G$ must have the same label.

By focusing on the perimeter of the commutative diagram of Fig. 2, we see that if $(H, M)$ is obtained from $(G, L)$ by basic graph operations, then there is a conjugacy $\psi : X_G \to X_H$ such that the diagram in Fig. 3 commutes.

In fact, the converse is true.

THEOREM 3.2. *Let $(G, L)$ and $(H, M)$ be labeled graphs which present the same sofic shift $X$. Then $(H, M)$ is obtained from $(G, L)$ by basic graph operations if and only if there is a conjugacy $\psi : X_G \to X_H$ such that the diagram in Fig.* 3 *commutes.*

*Proof.* It remains to verify the "*If*" direction. Decompose $\psi$ into basic graph conjugacies $\psi_i : X_{G_i} \to X_{G_{i+1}}$. We claim that this can be done so that all of the splittings come first. To see this, it suffices to show that if $H$ is an amalgamation (out or in) of $G$ and $K$ is a splitting (out or in) of $H$, then there is a graph $A$ such that $A$ is a splitting of $G$ and $K$ is an amalgamation of $A$ and the diagram of basic graph conjugacies in Fig. 4 commutes. This can be done by mimicking the splitting of $H$ by a splitting of $G$ or by using the fiber product construction—for the latter, see Proposition 8.6 and the definition of fiber product before it; we leave the proof as an exercise for the reader.

So, we may suppose that $G = G_0, G_1, \dots, G_\ell$ are constructed by out-splitings/in-splittings, and $G_{\ell+1}, \dots, G_n = H$ are constructed by out-amalgamations/in-amalgamations. For $0 \le i \le \ell - 1$, let $\theta_i = (\Theta_i)_\infty = (\psi_0)^{-1} \circ \cdots \circ (\psi_i)^{-1}$; then $\theta_i$ is a 1-block code, and we define $L_i = L \circ \Theta_i$. For $\ell \le i \le n-1$, let $\theta_i = (\Theta_i)_\infty = (\psi_{n-1}) \circ \cdots \circ (\psi_i)$; then $\theta_i$ is a 1-block code, and we define $L_i = M \circ \Theta_i$. The reader can verify that, with these choices, the diagram in Fig. 3 above commutes. □

In later sections, we will have more to say about the specific sequence of basic graph operations used to pass from $(G, L)$ to $(H, M)$ in terms of the conjugacy $\psi$. See Proposition 8.2 and Lemma 8.5.

$$\begin{tikzcd} & A \arrow[dr] & \\ G \arrow[ur] \arrow[dr] & & K \\ & H \arrow[ur] & \end{tikzcd}$$

FIG. 4.

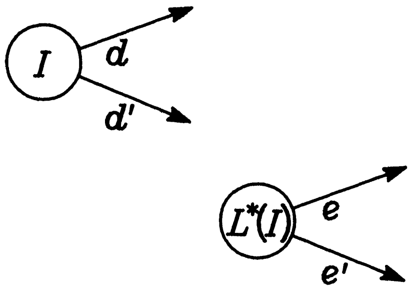


FIG. 5.

## 4. Right-resolving labelings and the Fischer cover.

A labeled graph is called *right resolving* if, at each state $I$, all of the outgoing edges from $I$ have distinct labels. Intuitively, this means that, given knowledge of an initial state, the label resolves the outgoing edge. In automata theory, "right resolving" is usually called "deterministic." When $(G, L)$ is right resolving, we will sometimes say that the labeling $L$, or the 1-block code $L_\infty$ that it generates, is right resolving.

Likewise, we have the notion of left resolving: replace "outgoing" with "incoming" in the definition above.

Consider the special case in which the following conditions hold: $G$ is irreducible, $(G, L)$ presents a graph shift $X_H$, and $L$ is right resolving; then, we have the following *unique lifting property*: $L$ is the edge mapping of a graph homomorphism $(L^*, L)$ from $G$ to $H$, and for each state $I$ of $G$ and each outgoing edge $e$ in $H$ from $L^*(I)$, there is a unique edge $d$ in $G$ such that $d$ is outgoing from $I$ and $L(d) = e$ as shown in Fig. 5. See [11, p. 8].

Every sofic shift $X$ has a right-resolving presentation. We define one such presentation below.

Let $X^+$ (resp., $X^-$) denote the set of all right (resp., left) semiinfinite sequences which appear in elements of $X$:

$$X^+ = \{x_{[0,\infty)} : x \in X\};$$

$$X^- = \{x_{(-\infty,-1]} : x \in X\}.$$

For a left semiinfinite sequence $x^- \in X^-$, define the *follower set* of $x^-$ to be the set of all right semiinfinite sequences that can follow it:

$$\mathcal{F}(x^-) = \{y^+ : x^- y^+ \in X\}.$$

Now, the *Krieger cover* $(K_X, M_X)$ of $X$ is the right-resolving labeled graph whose vertices are the follower sets $\{\mathcal{F}(x^-) : x \in X\}$ with an edge

$$\mathcal{F}_1 \to \mathcal{F}_2$$

labeled $u$ whenever there is an $x \in X$ such that $x_0 = u$, $\mathcal{F}_1 = \mathcal{F}(x^-)$ and $\mathcal{F}_2 = \mathcal{F}(x^-u)$. Such an edge is denoted $e(\mathcal{F}_1, u)$. It is well known that a shift space is sofic if and only if it has only finitely many follower sets, and if so, $(K_X, M_X)$ is a (right-resolving) presentation of $X$.

A shift space $X$ is called *irreducible* if for every (ordered) pair of blocks $u, v$ that appear in $X$, there is a block $w$ such that $uwv$ appears in $X$. An element $x \in X$ is called *left transitive* if every block that appears in $X$ appears infinitely often to the left in $x$. It is not hard to see that a shift space is irreducible if and only if it has a left-transitive point. In fact, if $X$ is irreducible, then the set of left-transitive points is dense in the sense that for every $x \in X$ and positive integer $n$, there is a left-transitive point $x' \in X$ such that $x'_{[-n,n]} = x_{[-n,n]}$.

For an irreducible sofic shift $X$, let $F_X$ be the subgraph of $K_X$ whose edges are incident to follower sets of left-transitive points (here, we mean a vertex which can be expressed as the follower set of a left-transitive point; it may also be expressable as the follower set of a point which is not left transitive). Clearly, $F_X$ is a sink component of $K_X$, i.e., every outgoing edge in $K_X$ from a vertex in $F_X$ is an edge in $F_X$. So, $F_X$ is irreducible. Let $L_X = M_X|F_X$. The labeled graph $(F_X, L_X)$ is called the *Fischer cover* of $X$.[1]

Let $X$ be an irreducible sofic shift. According to [13], $X$ is an SFT if and only if $(L_X)_\infty$ is 1-1. It is easy to see that $X$ is a graph shift if and only if $L_X$ is 1-1.

For a sofic shift $X$, a *magic word* $w$ is a block in $X$ such that whenever $uw$ and $wv$ are blocks in $X$, so is $uwv$. If a sofic shift is irreducible, then it has a magic word ([13]). Observe that if $x^- = (x')^-$ and this semiinfinite sequence contains a magic word, then $\mathcal{F}(x^-) = \mathcal{F}((x')^-)$. The only important thing to remember about left-transitive points is that they contain magic words infinitely often to the left; this, together with the fact that $(L_X)_\infty$ is right resolving, yields the following: If $x$ is left transitive, $x_{(-\infty,i]} = x'_{(-\infty,i]}$, $(L_X)_\infty(z) = x$, $(L_X)_\infty(z') = x'$, then $z_{(-\infty,i]} = z'_{(-\infty,i]}$. In particular, $(L_X)_\infty^{-1}(x)$ consists of a single point.

The next result summarizes some basic facts regarding the Fischer cover. In particular, the Fischer cover is the minimal right-resolving presentation.

THEOREM 4.1. *Let $X$ be an irreducible sofic shift with Fischer cover $(F_X, L_X)$.*

(1) ([13], *see also* [25]) *For every right-resolving presentation $(G, L)$ of $X$, with $G$ irreducible, there is a right-resolving graph homomorphism $\Psi : G \to F_X$ such that $\Psi_\infty$ is onto and the diagram in Fig. 6 commutes.*

$$\begin{tikzcd}
 & X_G \arrow[dl, "\Psi_\infty"'] \arrow[dr, "L_\infty"] & \\
X_{F_X} \arrow[rr, "(L_X)_\infty"] & & X
\end{tikzcd}$$

FIG. 6.

[1] Sometimes this is called the *Shannon cover* of $X$.

(2) ([24], *see also* [10]) *If $\phi : X \to Y$ is a conjugacy, then there is a unique conjugacy $c_\phi : X_{F_X} \to X_{F_Y}$ (called the lift of $\phi$) such that the diagram in Fig. 7 commutes.*

$$\begin{tikzcd}
X_{F_X} \arrow[r, "c_\phi"] \arrow[d, "(L_X)"'] & X_{F_Y} \arrow[d, "(L_Y)_\infty"] \\
X \arrow[r, "\phi"] & Y
\end{tikzcd}$$

FIG. 7.

The following is a useful fact regarding right-resolving labelings.

PROPOSITION 4.2. *Let $G, H$ be irreducible graphs, and let $f_1 : X_G \to X_H$, $f_2 : X_H \to Y$ be 1-block codes with $f_1$ onto. Then $f_2 \circ f_1$ is right (resp., left) resolving if and only if $f_2$ and $f_1$ are right (resp., left) resolving.*

*Proof.* Clearly, the composition of two right-resolving codes is right resolving. If $f_2 \circ f_1$ is right resolving, then $f_1$ cannot collapse two edges of $G$ having the same initial state. Thus $f_1$ is right resolving. So, it remains to prove:

$$f_2 \circ f_1 \text{ right resolving } \Rightarrow f_2 \text{ right resolving.}$$

Write $f_1 = (F_1)_\infty$ and $f_2 = (F_2)_\infty$. If $f_2$ were not right resolving, then there would be two distinct edges $e, e'$ in $H$, with the same initial state $J$ such that $F_2(e) = F_2(e')$.

Since $f_1$ is a 1-block code from a graph shift into another, $F_1$ is the edge mapping of a graph homomorphism: $(F_1^*, F_1)$. Since $f_1$ is a right-resolving code from one irreducible graph shift onto another, it has the unique lifting property. So, we can lift the edges $e, e'$ to distinct edges $d, d'$ in $G$ with the same initial state, namely any $F_1^*$-preimage $I$ of $J$, such that $F_1(d) = e, F_1(d') = e'$. This is shown in Fig. 8. However,

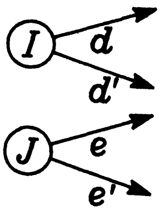


FIG. 8.

then $F_2 \circ F_1(d) = F_2 \circ F_1(d')$, contradicting the fact that $f_2 \circ f_1$ is right resolving. □

There is a "delayed" version of right resolving: a labeled graph $(G, L)$ is called *right closing* if there is a nonnegative integer $d$ such that for each state $I$ all paths of length $d + 1$ which begin at $I$ and have the same $L$-label must also have the same initial edge. We denote the smallest such $d$ by $d(G, L) = d(L) = d(L_\infty)$, and we call this the *delay* of $L$. If $(G, L)$ is not right closing, then we declare $d = \infty$. We sometimes call the labeling $L$ itself, or the 1-block code $L_\infty$, right closing. Observe that $L$ is right resolving if and only if $L$ is right closing with delay zero.

Both notions, right resolving and right closing, can be formulated in the more general context of sliding block codes on shift spaces as follows.

Let $\phi : X \to Y$ be a sliding block code on a shift space. We say that $\phi$ is right resolving if it is a 1-block code $\phi = \Phi_\infty$ and whenever $uv$ and $uv'$ are 2-blocks in $X$ (with the same initial symbol $u$) and $\Phi(uv) = \Phi(uv')$, then $v = v'$; we say that $\phi$ is right closing if whenever $x, x' \in X$, $x_{(-\infty,i]} = x'_{(-\infty,i]}$ for some $i$, and $\phi(x) = \phi(x')$, then $x = x'$.

We leave it to the reader to verify that these definitions reduce to the definitions that we gave earlier in the case that $\phi$ is a 1-block code on a graph shift (i.e., induced by a labeled graph).

The following is an analogue of Proposition 4.2 for right-closing codes.

PROPOSITION 4.3 (see [10]). *Let $X, Y$ be irreducible SFTs and $f : X \to Y$, $g : Y \to Z$ be sliding block codes with $f$ onto. Then $g \circ f$ is right closing (resp., left closing) if and only if $g$ and $f$ are right closing (resp., left closing). Also, if $X$ and $Y$ are merely sofic and $g \circ f$ is right closing (resp., left closing), then $f$ is right closing (resp., left closing).*

The proof of this is analogous to that of Proposition 4.2; it is contained in [10, Lem. 12].

For right-closing labelings, we have the following more specific version of Proposition 4.3, which generalizes Proposition 4.2. Since it is not used directly in the remainder of this paper, we defer the proof to Appendix I.

PROPOSITION 4.4. *Let $G, H$ be irreducible graphs, and let $f_1 : X_G \to X_H$, $f_2 : X_H \to Y$ be 1-block codes with $f_1$ onto. Then $d(f_2 \circ f_1) \leq d(f_2) + d(f_1)$, $d(f_1) \leq d(f_2 \circ f_1)$, and $d(f_2) \leq d(f_2 \circ f_1)$. In particular, $f_2 \circ f_1$ is right closing if and only if $f_2$ and $f_1$ are.*

There is a version of the notion of delay for sliding block codes on irreducible sofic shifts; this is given in §7. Proposition 4.4 actually generalizes to this setting.

## 5. A basic construction.

In this section, we associate some apparatus to sliding block codes—especially sliding block codes on sofic shifts.

Let $X, Y$ be shift spaces over the alphabets $\mathcal{U}, \mathcal{V}$. Let $\phi : X \to Y$ be a sliding block code. Define

$$S_\phi = \{(x, \phi(x)) : x \in X\}.$$

We naturally identify a pair of sequences with a sequence of pairs, and regard $S_\phi$ as a subshift over the alphabet $\mathcal{U} \times \mathcal{V}$. Define

$$\mathcal{I}_\phi : \mathcal{U} \times \mathcal{V} \to \mathcal{V}, \ \ \mathcal{I}_\phi(u, v) = v$$

and

$$\mathcal{O}_\phi : \mathcal{U} \times \mathcal{V} \to \mathcal{U} \ \ \mathcal{O}_\phi(u, v) = u.$$

Then $(\mathcal{I}_\phi)_\infty : S_\phi \to Y$ and $(\mathcal{O}_\phi)_\infty : S_\phi \to X$ are 1-block codes, and we have

$$(\mathcal{O}_\phi)_\infty(x, \phi(x)) = x, \ \ (\mathcal{I}_\phi)_\infty(x, \phi(x)) = \phi(x).$$

Note that $\phi$ is a factor map (i.e., onto $Y$), if and only if $(\mathcal{I}_\phi)_\infty$ is.

The following is a consequence of the definitions.

PROPOSITION 5.1. *Let $X$ and $Y$ be shift spaces and $\phi : X \to Y$ be a sliding block code. Then $(\mathcal{O}_\phi)_\infty$ is a conjugacy, and we have the commutative diagram in Fig. 9.*

*Proof.* $(\mathcal{O}_\phi)_\infty$ is a conjugacy since $x$ determines $\phi(x)$. The commutativity of the diagram is immediate. □

$$\begin{tikzcd}
& S_\phi \arrow[dl, "(\mathcal{O}_\phi)_\infty"'] \arrow[dr, "(\mathcal{I}_\phi)_\infty"] & \\
X \arrow[rr, "\phi"] & & Y
\end{tikzcd}$$

FIG. 9.

So, $X$ and $S_\phi$ share many properties, such as the following: $X$ is an SFT if and only if $S_\phi$ is an SFT; $X$ is sofic if and only if $S_\phi$ is sofic; $X$ is irreducible if and only if $S_\phi$ is irreducible. In particular, if $X$ is an irreducible sofic shift, then, so is $S_\phi$, and thus it too has a Fischer cover. We denote the Fischer cover $(F_{S_\phi}, L_{S_\phi})$ by simply $(F_\phi, L_\phi)$. We write

$$\bar{\mathcal{I}}_\phi = \mathcal{I}_\phi \circ L_\phi, \quad \bar{\mathcal{O}}_\phi = \mathcal{O}_\phi \circ L_\phi.$$

Applying the basic properties of the Fischer cover in this setting, we obtain the following theorem.

THEOREM 5.2. *Let $X$ be an irreducible sofic shift, and let $\phi : X \to Y$ be a sliding block code.*

(1) *The diagram of Fig.* 10 *commutes.*

$$\begin{tikzcd}
& & X_{F_\phi} \arrow[d, "(L_\phi)_\infty"] \arrow[ddll, bend right=40, "c(\mathcal{O}_\phi)_\infty"'] \arrow[ddl, bend right, "(\bar{\mathcal{O}}_\phi)_\infty"'] \arrow[ddr, bend left, "(\bar{\mathcal{I}}_\phi)_\infty"] & \\
& & S_\phi \arrow[dl, "(\mathcal{O}_\phi)_\infty"'] \arrow[dr, "(\mathcal{I}_\phi)_\infty"] & \\
X_{F_X} \arrow[r, "(L_X)_\infty"] & X \arrow[rr, "\phi"] & & Y
\end{tikzcd}$$

FIG. 10.

(2) $(F_\phi, \bar{\mathcal{O}}_\phi) = (F_\phi,\ \mathcal{O}_\phi \circ L_\phi)$ *is obtained from* $(F_X, L_X)$ *by a sequence of basic graph operations.*

*Proof.* (1): this follows from Proposition 5.1, Theorem 4.1 (part (2)), and the definitions of $\bar{\mathcal{I}}_\phi, \bar{\mathcal{O}}_\phi$. (2): this comes from the commutativity of the left-most triangle in part (1) and Theorem 3.2. □

When $\bar{\mathcal{I}}_\phi$ is right resolving, the apparatus given in Theorem 5.2 has more meaning. This is developed in the next section.

**6. Encoders, decoders, and the canonical encoder.** A *finite-state code* $(G, \mathcal{I}, \mathcal{O})$ is a graph $G$, together with two labelings $\mathcal{I}, \mathcal{O}$ (the *input* and *output* labelings) such that $\mathcal{I}_\infty$ is right resolving and $\mathcal{O}_\infty$ is right closing.

The idea is that we can use a finite-state code as a finite-state machine to encode right semiinfinite sequences presented by $(G, \mathcal{I})$ (well, at least all sequences that are presented by the $\mathcal{I}$-labeling of $G$ beginning at some fixed state of $G$) to right semiinfinite sequences presented by $(G, \mathcal{O})$; since $\mathcal{I}$ is right resolving, the encoding is accomplished symbol-by-symbol, and since $\mathcal{O}$ is right closing, the decoding is accomplished with bounded delay.

Recall from §1 that we imagine that encoded information is transmitted across a noisy channel and then decoded. Recall also that it is desirable that each decoded symbol be a function of only a bounded number of encoded symbols, and thus that decoding be implemented by a sliding block decoder.

A *sliding block decoder* $\phi : X \to Y$ is a sliding block code for which there is a finite-state code $(G,\mathcal{I},\mathcal{O})$ such that $X = \mathcal{O}_\infty(X_G)$, $\mathcal{I}_\infty(X_G) = Y$, and the diagram in Fig. 11 commutes. We say that $\phi$ is a *decoder* for $(G,\mathcal{I},\mathcal{O})$ and that $(G,\mathcal{I},\mathcal{O})$ is an *encoder* for $\phi$. Note that, by definition, $\phi(X) = \mathcal{I}_\infty(X_G) = Y$.

$$\begin{tikzcd}
 & X_G \arrow[dl, "\mathcal{O}_\infty"'] \arrow[dr, "\mathcal{I}_\infty"] & \\
X \arrow[rr, "\phi"] & & Y
\end{tikzcd}$$

FIG. 11.

If the diagram in Fig. 11 commutes, and $\mathcal{I}$ is right resolving, then by Proposition 4.3, $\mathcal{O}$ is automatically right closing. So, we have the following proposition.

PROPOSITION 6.1. *An encoder for a sliding block decoder $\phi$ is a $(G,\mathcal{I},\mathcal{O})$ such that the diagram in Fig.* 11 *commutes and $\mathcal{I}_\infty$ is right resolving.*

The following result shows that every sliding block decoder on an irreducible sofic shift has a canonical minimum encoder. The proof is based on an idea of M. Nasu's. In [31, §7], he views a pair of graph homomorphisms $p : K \to G$ and $q : K \to H$ as a presentation $(K,(p,q))$ of a sofic shift, and then considers a canonical cover of this sofic shift. Here, we consider the Fischer cover $X_{F_\phi}$ of the sofic shift $S_\phi$.

THEOREM 6.2. *Let $X$ be an irreducible sofic shift, and let $\phi : X \to Y$ be a sliding block decoder. Then $(F_\phi,\bar{\mathcal{I}}_\phi,\bar{\mathcal{O}}_\phi)$ is an encoder for $\phi$ and is the unique minimum encoder in the following sense:*

*Let $(G,\mathcal{I},\mathcal{O})$ be any encoder for $\phi$ with $G$ irreducible. Then there is a right-resolving graph homomorphism $M : G \to F_\phi$ such that $M_\infty$ is onto and the diagram in Fig.* 12 *commutes.*

$$\begin{tikzcd}
 & X_G \arrow[d, "M_\infty"] \arrow[ddl, bend right, "\mathcal{O}_\infty"'] \arrow[ddr, bend left, "\mathcal{I}_\infty"] & \\
 & X_{F_\phi} \arrow[dl, "(\bar{\mathcal{O}}_\phi)_\infty"'] \arrow[dr, "(\bar{\mathcal{I}}_\phi)_\infty"] & \\
X \arrow[rr, "\phi"] & & Y
\end{tikzcd}$$

FIG. 12.

*Proof.* Let $(G,\mathcal{I},\mathcal{O})$ be an encoder for $\phi$ with $G$ irreducible. Let $\Psi$ be the labeling of the graph $G$ defined by

$$\Psi(e) \equiv (\mathcal{O}(e),\mathcal{I}(e)).$$

Then

$$\psi \equiv \Psi_\infty(z) = (\mathcal{O}_\infty(z), \mathcal{I}_\infty(z)),$$

and we have the commutative diagram in Fig. 13. Now, since $\psi, (\mathcal{I}_\phi)_\infty$ and $\mathcal{I}_\infty$ are

$$\begin{tikzcd}
& X_G \arrow[d, "\psi"] \arrow[ddl, bend right, "\mathcal{O}_\infty"'] \arrow[ddr, bend left, "\mathcal{I}_\infty"] & \\
& S_\phi \arrow[dl, "(\mathcal{O}_\phi)_\infty"'] \arrow[dr, "(\mathcal{I}_\phi)_\infty"] & \\
X \arrow[rr, "\phi"] & & Y
\end{tikzcd}$$

FIG. 13.

1-block codes and $\mathcal{I}$ is right resolving, so is $\Psi$. Since $\mathcal{O}_\infty$ maps onto $X$, $(G, \Psi)$ is a right-resolving presentation of $S_\phi$. Thus, by Theorem 4.1 (part 1), there is a right-resolving graph homomorphism $M : G \to F_\phi$ such that $M_\infty$ is onto and the diagram in Fig. 14 commutes. Thus, the diagram in Fig. 12 commutes.

$$\begin{tikzcd}
X_G \arrow[d, "M_\infty"] \arrow[dd, bend right=60, "\psi"'] \\
X_{F_\phi} \arrow[d, "(L_\phi)_\infty"] \\
S_\phi
\end{tikzcd}$$

FIG. 14.

By Proposition 4.2, since $\mathcal{I}$ is right resolving, so is $\bar{\mathcal{I}}_\phi$; thus, $(F_\phi, \bar{\mathcal{I}}_\phi, \bar{\mathcal{O}}_\phi)$ is indeed an encoder for $\phi$. $\square$

Note that the minimality of the Fischer cover (Theorem 4.1 (part (1))) amounts to the special case of Theorem 6.2 where $X = Y$ and $\phi = \text{id}$ (here, and in the remainder of the paper, id denotes the identity map).

We now consider sliding block decoders that have an encoder $(G, \mathcal{I}, \mathcal{O})$ such that $\mathcal{O}$ is left resolving and $X$, the domain of $\phi$, is a graph shift. In the following, we will see that in such a case $\bar{\mathcal{O}}_\phi$ itself is left resolving.

PROPOSITION 6.3. *Let $\phi$ be a sliding block decoder on an irreducible graph shift $X = X_H$ which has an encoder $(G, \mathcal{I}, \mathcal{O})$ such that $G$ is irreducible and $\mathcal{O}$ is left resolving. Then, $(\bar{\mathcal{O}}_\phi)_\infty$ is a left-resolving conjugacy.*

*Proof.* By Theorem 6.2, we may write

$$\mathcal{O}_\infty = (\bar{\mathcal{O}}_\phi)_\infty \circ M_\infty, \tag{1}$$

where $M : G \to F_\phi$ is a graph homomorphism with $M_\infty$ onto. By Proposition 4.2, since $\mathcal{O}$ is left resolving, so is $\bar{\mathcal{O}}_\phi$.

Since $X$ is an SFT, $(L_\phi)_\infty$ is a conjugacy. Thus, since $(\mathcal{O}_\phi)_\infty$ is always a conjugacy, $(\bar{\mathcal{O}}_\phi)_\infty$ is a conjugacy. So, $(\bar{\mathcal{O}}_\phi)_\infty$ is a left-resolving conjugacy. □

If, in the preceding result, $\mathcal{O}_\infty$ is a left-resolving *conjugacy*, then by (1) $M_\infty : X_G \to X_{F_\phi}$ is a left-resolving conjugacy. But since $\mathcal{I}_\infty = (\bar{\mathcal{I}}_\phi)_\infty \circ M_\infty$ and $\mathcal{I}$ is right resolving, $M_\infty$ is a biresolving (i.e., both right- and left-resolving) conjugacy. Since the domain and range of this code are both irreducible graph shifts, $M$ is a graph isomorphism (see [30, §2], [5, Lem. 4]). Thus, we have recovered the result [5, Thm. 2] that whenever a conjugacy $\phi : X_G \to X_H$ between graph shifts can be expressed as

$$\phi = g \circ f^{-1}, \tag{2}$$

where $f : X_K \to X_G$ is a left-resolving conjugacy and $g : X_K \to X_H$ is a right-resolving conjugacy, then expression (2) is unique (up to codes that simply relabel the alphabet). M. Nasu has informed us that this also follows directly from [31, Lem. 3.12].

**7. Characterization of sliding block decoders.** Our next result characterizes when a sliding block code is a sliding block decoder in terms of a "right-resolving" type of property. The proof of this result pretty much boils down to an understanding of what $(F_\phi, L_\phi)$ really is. Well, $(F_\phi, L_\phi)$, is the right-resolving labeled graph whose vertices are the follower sets $\mathcal{F}((x,y)_{(-\infty,-1]})$ where $(x,y) \in S_\phi$ is left transitive, with an edge

$$\mathcal{F}((x,y)_{(-\infty,-1]}) \to \mathcal{F}((x',y')_{(-\infty,-1]})$$

labeled $(u,v)$ whenever $(x',y')_{(-\infty,-1]} = (x,y)_{(-\infty,-1]}(u,v)$; such an edge is denoted $e(\mathcal{F},(u,v))$, where $\mathcal{F} = \mathcal{F}((x,y)_{(-\infty,-1]})$ is the initial vertex.

THEOREM 7.1. *Let $\phi$ be a sliding block code on an irreducible sofic shift $X$. Then $\phi$ is a sliding block decoder if and only if for all left-transitive points $x \in X$ and $y = \phi(x)$, $x_{(-\infty,-1]}$ and $y_{(-\infty,0]}$ determine $x_0$ (that is, if $x, x' \in X$ are left transitive, $y = \phi(x), y' = \phi(x')$, $x_{(-\infty,-1]} = x'_{(-\infty,-1]}$, and $y_{(-\infty,0]} = y'_{(-\infty,0]}$, then $x_0 = x'_0$).*

*Proof.* Suppose that $\phi$ is a sliding block decoder.

By Theorem 6.2, $(F_\phi, \bar{\mathcal{I}}_\phi, \bar{\mathcal{O}}_\phi)$ is an encoder for $\phi$. So, $\bar{\mathcal{I}}_\phi$ is right resolving, and this means that $\mathcal{F}$ and $v$ uniquely determine the edge $e(\mathcal{F},(u,v))$; equivalently, $\mathcal{F}$ and $v$ uniquely determine $u$.

Now, if $x \in X$ is left transitive and $y = \phi(x)$, then $(x,y) \in S_\phi$ is left transitive too. So, $\mathcal{F} \equiv \mathcal{F}((x,y)_{(-\infty,-1]}) \in \mathcal{V}(F_\phi)$. Now, $x_{(-\infty,-1]}$ and $y_{(-\infty,-1]}$ uniquely determine the vertex $\mathcal{F}$. Since $\bar{\mathcal{I}}_\phi$ is right resolving, $\mathcal{F}$ and $y_0$ uniquely determine the edge $e(\mathcal{F},(x_0,y_0))$ and therefore $x_0$. Thus, $x_{(-\infty,-1]}$ and $y_{(-\infty,0]}$ determine $x_0$.

For the converse, we will show that if the condition holds, then $\bar{\mathcal{I}}_\phi$ is right resolving and so $\phi$ is a sliding block decoder.

Let $\mathcal{F}$ be a vertex in $F_\phi$, and let $e(\mathcal{F},(u,v))$ be an outgoing edge from $\mathcal{F}$. Then there is a left-transitive point $(x,y) \in S_\phi$ such that $\mathcal{F} = \mathcal{F}((x,y)_{(-\infty,-1]})$ and $(u,v) = (x_0,y_0)$. Now, $x$ is left transitive, and $y = \phi(x)$. So, $x_{(-\infty,-1]}$ and $y_{(-\infty,0]}$ determine $x_0$. So, $\mathcal{F}$ and $v$ determine $e(\mathcal{F},(u,v))$, and so $\bar{\mathcal{I}}_\phi$ is right resolving, as desired. □

The goal of the remainder of this section is to give a version of the condition in the preceding result that is more concrete and is based on graph shifts.

Recall that the memory and anticipation of a sliding block code are not uniquely defined. Nevertheless, it is natural to consider the minimum memory and anticipation of a sliding block code:

$$a_{\min} \equiv a_{\min}(\phi) \equiv \min\{a : \text{ for some } m,\ \phi \text{ is an } (m,a)\text{-block code}\}$$
$$m_{\min} \equiv m_{\min}(\phi) \equiv \min\{m : \text{ for some } a,\ \phi \text{ is an } (m,a)\text{-block code}\}.$$

The following result shows that the minimum memory is actually independent of the minimum anticipation for sliding block codes on graph shifts. See [21] for an example which shows that this result is false for sliding block codes on sofic shifts—in fact on SFTs—in general.

PROPOSITION 7.2. *Let $\phi$ be a sliding block code on a graph shift $X_G$.*

(1) *$\phi$ is an $(m_{\min}, a_{\min})$-block code.*

(2) *If $\phi$ is an $(m,a)$-block code, then $m + a + 1 \geq m_{\min} + a_{\min} + 1$ with equality if and only if $m = m_{\min}$ and $a = a_{\min}$. In particular, the minimum window length, $m_{\min} + a_{\min} + 1$, is divided uniquely between memory and anticipation.*

*Proof.* (1): Let

$$m^* = \min\{m : \phi \text{ is an } (m, a_{\min})\text{-block code}\}.$$

Clearly $m^* \geq m_{\min}$. We must show that $m^* = m_{\min}$. Suppose not. Then

$$m_{\min} < m^*.$$

Write $\phi = \Phi_\infty^{m^*, a_{\min}}$. By the definition of $m^*$, there are paths in $G$

$$\gamma = e_{-m^*} \dots e_{a_{\min}}$$

and

$$\eta = f_{-m^*} \dots f_{a_{\min}}$$

of length $m^* + a_{\min} + 1$ such that

$$\Phi(\gamma) \neq \Phi(\eta) \quad \text{and} \quad e_i = f_i, \quad -m^* + 1 \leq i \leq a_{\min}.$$

Now, we can also write $\phi = \Psi_\infty^{m_{\min}, a}$ for some $a \geq a_{\min}$. Let $\omega$ be a path of length $a - a_{\min}$ which begins at $t(\gamma) = t(\eta)$. Let $x, y \in X_G$ such that

$$x_{[-m^*, a]} = \gamma\omega \quad \text{and} \quad y_{[-m^*, a]} = \eta\omega.$$

Then,

$$\phi(x)_0 = \Phi(\gamma) \neq \Phi(\eta) = \phi(y)_0. \tag{3}$$

But since $m_{\min} < m^*$,

$$\phi(x)_0 = \Psi(x_{[-m_{\min}, a]}) = \Psi(y_{[-m_{\min}, a]}) = \phi(y)_0. \tag{4}$$

Equations (3) and (4) contradict one another. This yields part (1).

(2): If $m + a + 1 = m_{\min} + a_{\min} + 1$ and $m > m_{\min}$, then $a < a_{\min}$, contrary to the definition of $a_{\min}$. This gives Part (2). □

Write $\gamma = \phi \circ (L_X)_\infty$. Define

$$a = a(\phi) \equiv a_{\min}(\gamma), \quad m = m(\phi) \equiv m_{\min}(\gamma)$$

and define $d(\phi)$ as the smallest nonnegative integer such that for $z \in X_{F_X}$ and $y = \gamma(z)$, $z_{[-m,a-1]}$ and $y_{[0,d]}$ determine $z_a$. It is easy to see that this generalizes the notion of delay for labelings that we gave in §4.

The following proposition characterizes sliding block decoders in terms of $d(\phi)$ and $a(\phi)$.

PROPOSITION 7.3. *Let $\phi : X \to Y$ be a sliding block code on an irreducible sofic shift $X$. Then $\phi$ is a sliding block decoder if and only if*

$$d(\phi) \leq a(\phi).$$

*Proof.* Suppose that $\phi$ is a sliding block decoder. Let $z, z' \in X_{F_X}$, $y = \gamma(z)$, $y' = \gamma(z')$ be such that

$$z_{[-m,a-1]} = z'_{[-m,a-1]} \quad \text{and} \quad y_{[0,a]} = y'_{[0,a]}.$$

We will show that $z_a = z'_a$. Then $d(\phi) \leq a(\phi)$.

We may assume $z$ and $z'$ are left transitive and $z_{(-\infty,a-1]} = z'_{(-\infty,a-1]}$. Let

$$x = (L_X)_\infty(z), \qquad x' = (L_X)_\infty(z').$$

Then $x$ and $x'$ are left transitive. Now $x_{(-\infty,a-1]} = x'_{(-\infty,a-1]}$ and $y_{(-\infty,a]} = y'_{(-\infty,a]}$. Thus by Theorem 7.1, $x_a = x'_a$. Since $L_X$ is right resolving, $z_a = z'_a$.

Suppose $d \leq a$. Let $x, x' \in X$ be left transitive, $y = \phi(x)$, $y' = \phi(x')$, $x_{(-\infty,-1]} = x'_{(-\infty,-1]}$, and $y_{(-\infty,0]} = y'_{(-\infty,0]}$. We will show $x_0 = x'_0$. Choose $z, z' \in X_{F_X}$ such that $(L_X)_\infty(z) = x$ and $(L_X)_\infty(z') = x'$. Then $z_{(-\infty,-1]} = z'_{(-\infty,-1]}$. Since $y_{(-\infty,0]} = y'_{(-\infty,0]}$ and $d \leq a$, we have $z_0 = z'_0$. Therefore $x_0 = x'_0$. Therefore $\phi$ is a sliding block decoder by Theorem 7.1. □

In the next proposition, we characterize the delay of a sliding block decoder $\phi : X \to Y$ in terms of the output labeling $\mathcal{O}$ of any finite-state code $(G, \mathcal{I}, \mathcal{O})$ having sliding block decoder $\phi$.

PROPOSITION 7.4. *Let $G$ be an irreducible graph. Let $(G, \mathcal{I}, \mathcal{O})$ be a finite-state code having a sliding block decoder $\phi : X \to Y$. Then*

$$d(\phi) = \min\{d : \forall u \in X_G,\ u_0 \text{ determines } \mathcal{O}_\infty(u)_{[0,a-d]}\}.$$

*Proof.* Write $d = d(\phi)$ and

$$d' = \min\{d : \forall u \in X_G,\ u_0 \text{ determines } \mathcal{O}_\infty(u)_{[0,a-d]}\}.$$

$d' \leq d$: Let $u \in X_G$. We must show that $u_0$ determines $\mathcal{O}_\infty(u)_{[0,a-d]}$. We can assume that $u$ is left transitive. Let $z \in X_{F_X}$ be such that $(L_X)_\infty(z) = \mathcal{O}_\infty(u)$. Set $y = \mathcal{I}_\infty(u)$. By the definition of delay, $z_{(-\infty,-d+i]}$ and $y_{(-\infty,1-a+i]}$ together determine $z_{1-d+i}$. Thus $z_{(-\infty,-d]}$ and $y_{(-\infty,0]}$ determine $z_{[1-d,a-d]}$. Now $u_{(-\infty,0]}$ determines both $z_{(-\infty,a-d]}$ and $y_{(-\infty,0]}$; so $u_{(-\infty,0]}$ determines $(L_X)_\infty(z)_{[1-d,a-d]} = \mathcal{O}_\infty(u)_{[1-d,a-d]}$, so $u_{(-\infty,0]}$ determines $\mathcal{O}_\infty(u)_{[0,a-d]}$. But $\mathcal{O}_\infty$ is a 1-block code on $X_G$; so actually, the edge $u_0$ determines $\mathcal{O}_\infty(u)_{[0,a-d]}$, as was to be shown.

$d \leq d'$: Fix $z \in X_{F_X}$ and set $y = \phi \circ (L_X)_\infty(z)$. We must show that $z_{[-m,a-1]}$ and $y_{[0,d']}$ determine $z_a$. Since $X_{F_X}$ is a graph shift and $\phi \circ (L_X)_\infty$ has memory $m$, it suffices to show that $z_{(-\infty,a-1]}$ and $y_{[0,d']}$ determine $z_a$. By Theorem 6.2

$$d' = \min\{d : \forall u \in X_{F_\phi},\ u_0 \text{ determines } (\bar{\mathcal{O}}_\phi)_\infty(u)_{[0,a-d]}\}.$$

Now $d(\bar{\mathcal{O}}_\phi) \leq a$. (Otherwise there would be two $(m+a+1)$-blocks $u_{-m}\dots u_{-1}u_0\dots u_a$ and $u_{-m}\dots u_{-1}u'_0\dots u'_a$ in $F_\phi$ with $u_0 \neq u'_0$ (and therefore with $\bar{\mathcal{I}}(u_0) \neq \bar{\mathcal{I}}(u'_0)$) but with $\bar{\mathcal{O}}_\phi(u_{-m}\dots u_{-1}u_0\dots u_a) = \bar{\mathcal{O}}_\phi(u_{-m}\dots u_{-1}u'_0\dots u'_a)$.) By Theorem 5.2, there is a conjugacy $\theta : X_{F_X} \to X_{F_\phi}$. Now $z_{(-\infty,a-1]}$ determines $(L_X)_\infty(z)_{(-\infty,a-1]} = (\bar{\mathcal{O}}_\phi)_\infty(\theta(z))_{(-\infty,a-1]}$. Using that $d(\bar{\mathcal{O}}_\phi) \leq a$, $(\bar{\mathcal{O}}_\phi)_\infty(\theta(z))_{(-\infty,a-1]}$ together with $z_{(-\infty,a-1]}$ determines $\theta(z)_{(-\infty,-1]}$. Use that $\bar{\mathcal{I}}_\phi$ is right resolving to see that $\theta(z)_{(-\infty,-1]}$ together with $y_{[0,d']}$ determines $\theta(z)_{(-\infty,d']}$. By the definition of $d'$, $\theta(z)_{(-\infty,d']}$ determines $(\bar{\mathcal{O}}_\phi)_\infty(\theta(z))_{(-\infty,a-d'+d']} = (L_X)_\infty(z)_{(-\infty,a]}$. But this (together with $z_{(-\infty,a-1]}$ again) determines $z_a$, as was to be shown. $\square$

Sliding block decoders are intimately related to the following class of codes introduced in [11, §4].

A sliding block code $\phi$ on an irreducible sofic shift is *right closing a.e.* if for each *left-transitive* $x \in X$, the tail $x_{(-\infty,-1]}$ and the image $\phi(x)$ of $x$, taken together, determine $x$.

From [11, §4], one can see that $\phi$ is right closing a.e. if and only if $\phi \circ (L_X)_\infty$ is right closing (so, if $X$ is an SFT, then right closing a.e. $\Rightarrow$ right closing). And from this, one can show that $\phi$ is right closing a.e. if and only if there is some $k \geq 0$ such that for each left-transitive $x \in X$, $x_{(-\infty,-1]}$ and $\phi(x)_{(-\infty,k]}$, taken together, determine $x_0$.

From this and Theorem 7.1, we obtain the following corollary.

COROLLARY 7.5. *A sliding block code $\phi$ on an irreducible sofic shift $X$ is right closing a.e. if and only if for some $k \geq 0$, $\phi \circ \sigma^k$ is a sliding block decoder.*

## 8. Construction of $(F_\phi, \bar{\mathcal{O}}_\phi)$ by basic graph operations.

We view the canonical encoder $(F_\phi, \bar{\mathcal{I}}_\phi, \bar{\mathcal{O}}_\phi)$ as the labeled graph $(F_\phi, \bar{\mathcal{O}}_\phi)$ together with a choice of right-resolving labeling $\bar{\mathcal{I}}_\phi$. In the main case of practical interest, $Y$ is the full $k$-shift. In this case, $F_\phi$ has out-degree $k$ at each state, and $\bar{\mathcal{I}}_\phi$ is simply a 1-1 assignment, for each state $I$ in $F_\phi$, of each of the $k$-ary symbols to the edges outgoing from state $I$ (such a labeling is sometimes called a road coloring). So, in some sense, the heart of the construction of $(F_\phi, \bar{\mathcal{I}}_\phi, \bar{\mathcal{O}}_\phi)$ is the construction of $(F_\phi, \bar{\mathcal{O}}_\phi)$. This is why we focus, in this section, on how $(F_\phi, \bar{\mathcal{O}}_\phi)$ can be obtained from the Fischer cover $(F_X, L_X)$ by an explicit sequence of basic graph operations. This strengthens Theorem 5.2 (part (2)).

This section is somewhat technical. So, we recommend that the reader skim it first, paying particular attention to the statements of Theorem 8.1, the left and right Markov properties, Proposition 8.2, and Proposition 8.11. After reading the remaining sections, the reader can then come back to this section in detail.

THEOREM 8.1. *Let $\phi$ be a sliding block decoder on an irreducible sofic shift $X$. Then $(F_\phi, \bar{\mathcal{O}}_\phi)$ is obtained from $(F_X, L_X)$ by a sequence of at most $m(\phi) + d(\phi)$ in-splittings, followed by at most $a(\phi)$ out-splittings, followed by a sequence of in-amalgamations. That is, there are presentations*

$$(F_X, L_X) = (G_0, L_0), \dots, (G_n, L_n) = (F_\phi, \bar{\mathcal{O}}_\phi)$$

*of $X$ and basic graph conjugacies $\psi_i : X_{G_i} \to X_{G_{i+1}}$ such that for $i = 0, \dots, m(\phi) + d(\phi) - 1$, $\psi_i$ are in-splittings, for $i = m(\phi) + d(\phi), \dots, m(\phi) + d(\phi) + a(\phi) - 1$, $\psi_i$ are out-splittings, for $i = m(\phi) + d(\phi) + a(\phi), \dots, n-1$, $\psi_i$ are in-amalgamations, and the diagram in Fig. 15 commutes.*

Observe that if one is interested in the construction of only *some* encoder for $\phi$ (rather than the canonical encoder), then the in-amalgamations can be eliminated.

$$\begin{tikzcd}
X_{F_X} \arrow[r, "\psi_0"] \arrow[dr, "(L_X)_\infty"'] & X_{G_1} \arrow[r, "\psi_1"] & \cdots \arrow[r, "\psi_{n-1}"] & X_{F_\emptyset} \arrow[dll, "(\overline{\mathcal{O}}_\emptyset)_\infty"] \\
 & X & &
\end{tikzcd}$$

FIG. 15.

We break the proof of Theorem 8.1 down into Proposition 8.2 and Lemmas 8.3, 8.4, and 8.5.

Proposition 8.2 shows that all resolving conjugacies $\psi : X_G \to X_H$ between graph shifts can be constructed by state splitting. The proposition also gives an interpretation of the states of $G$ as the atoms of a partition $\mathcal{P}$ of the length-$m$ paths of $H$, where $m+1$ is the block length of $\psi^{-1}$. These atoms are related to subsets of paths called *independent paths* in [15]. The partition $\mathcal{P}$ satisfies a property that we now explain.

Let $\mathcal{P}$ be a partition of paths in $H$ of length $m$. We use the notation $[w]$ to denote the atom of $\mathcal{P}$ to which $w$ belongs, and $w \sim w'$ to mean that $w$ and $w'$ belong to the same atom.

We say that $\mathcal{P}$ satisfies the *right Markov property* if the following holds:

$$\text{(5)} \qquad \begin{array}{c}\text{If } w \sim w' \text{ and } e \text{ is an edge that follows } w \text{ in } H,\\ \text{then } w_2 \dots w_m e \sim w'_2 \dots w'_m e.\end{array}$$

Note, in particular, that if $w \sim w'$, then $t(w) = t(w')$.

Define a graph $A^{\mathcal{P}}$ as follows. The vertices of the graph $A^{\mathcal{P}}$ are the atoms $[w]$ of $\mathcal{P}$. For each $[w] \in \mathcal{V}(A^{\mathcal{P}})$ and each edge $e$ that follows $w$ in $H$, we endow $A^{\mathcal{P}}$ with an edge, called $([w], e)$, from $[w]$ to $[w_2 \dots w_m e]$. By virtue of the right Markov property, this makes sense independent of the choice of representative of $[w]$.

We define the 1-block code $\psi^{\mathcal{P}} = (\Psi^{\mathcal{P}})_\infty$ by

$$\Psi^{\mathcal{P}}(([w], e)) = e.$$

Notice that $\Psi^{\mathcal{P}}$ is right resolving.

Symmetrically, we can consider partitions that satisfy the *left Markov property*:

$$\text{(6)} \qquad \begin{array}{c}\text{If } w \sim w' \text{ and } e \text{ is an edge that precedes } w \text{ in } H,\\ \text{then } e w_1 \dots w_{m-1} \sim e w'_1 \dots w'_{m-1}.\end{array}$$

And we have the analogous graph construction $A^{\mathcal{P}}$ and 1-block code $\psi^{\mathcal{P}}$.

PROPOSITION 8.2. *Let $\psi = \Psi_\infty : X_G \to X_H$ be a 1-block code, and let $m$ be a positive integer. Then the following conditions are equivalent:*

(1) *$\psi$ is a 1-block conjugacy with an $(m, 0)$-block inverse (resp., $(0, m)$-block inverse).*
(2) *$(G, \Psi)$ is obtained from $(H, \mathrm{id})$ by a sequence of $m$ in- (resp., out-) splittings.*
(3) *There is a partition $\mathcal{P}$ of the set of blocks of length $m$ in $H$ that satisfies the right Markov property (resp., left Markov property) such that $G$ is graph isomorphic to $A^{\mathcal{P}}$ and via this isomorphism, $\psi = \psi^{\mathcal{P}}$.*

*Moreover, a 1-block conjugacy $\psi : X_G \to X_H$ has an $(m, 0)$ (resp., $(0, m)$)-block inverse for some $m$ if and only if $\psi$ is right (resp., left) resolving.*

LEMMA 8.3. *Let $\phi$ be a sliding block decoder on an irreducible sofic shift $X$. Let*

$$\gamma = \phi \circ (L_X)_\infty : X_{F_X} \to Y. \tag{7}$$

*Then there is a right-resolving conjugacy*

$$\pi : X_{F_\gamma} \to X_{F_\phi}$$

*such that the diagram in Fig. 16 commutes.*

$$\begin{tikzcd}
X_{F_\gamma} \arrow[r, "\pi"] \arrow[d, "(\bar{O}_\gamma)_\infty"'] & X_{F_\phi} \arrow[d, "(\bar{O}_\phi)_\infty"] \\
X_{F_X} \arrow[r, "(L_X)_\infty"] & X
\end{tikzcd}$$

FIG. 16.

LEMMA 8.4. *Let $\gamma$ be a sliding block decoder on an irreducible graph shift $X_G$. Then $(\bar{O}_\gamma)_\infty : X_{F_\gamma} \to X_G$ is a 1-block conjugacy whose inverse is an $(m(\gamma) + d(\gamma), a(\gamma))$-block code.*

LEMMA 8.5. *Let $\theta = \Theta_\infty : X_G \to X_H$ be a 1-block conjugacy from one graph shift to another such that $\theta^{-1}$ is an $(m, a)$-block code. Then $(G, \Theta)$ is obtained from $(H, \mathrm{id})$ by a sequence of at most $m$ in-splittings, at most $a$ out-splittings and at most $m$ in-amalgamations.*

Before we prove the lemmas, we show how to use them to prove Theorem 8.1.

*Proof of Theorem* 8.1. Let $\gamma$ be as in (7). By Proposition 8.2 and Lemma 8.3, $(F_\phi, \bar{O}_\phi)$ is obtained from $(F_\gamma, L_X \circ \bar{O}_\gamma)$ by a sequence of in-amalgamations.

By definition, $m(\gamma) = m(\phi), a(\gamma) = a(\phi)$ and $d(\gamma) = d(\phi)$. In particular, we see from Proposition 7.3, that $\gamma$ is also a sliding block decoder. We have, by Lemmas 8.4 and 8.5 (applied to $\theta = (\bar{O}_\gamma)_\infty$), that $(F_\gamma, \bar{O}_\gamma)$ is obtained from $(F_X, \mathrm{id})$ by a sequence of at most $m(\phi) + d(\phi)$ in-splittings, followed by at most $a(\phi)$ out-splittings, followed by at most $m(\phi) + d(\phi)$ in-amalgamations.

So, $(F_\phi, \bar{O}_\phi)$ is obtained from $(F_X, L_X)$ by a sequence of at most $m(\phi) + d(\phi)$ in-splittings, followed by at most $a(\phi)$ out-splittings, followed by a sequence of in-amalgamations, as desired. □

Thus we have the commutative diagram of Fig. 17.

$$\begin{tikzcd}
\bullet \arrow[rr, "\pi", "\text{in-amal.}"'] & & X_{F_\gamma} \arrow[rr, "\pi", "\text{in-amal.}"'] \arrow[dd, "(\bar{O}_\gamma)_\infty"] & & X_{F_\phi} \arrow[dd, "(\bar{O}_\phi)_\infty"] \\
& \bullet \arrow[ul, "\text{out-split}"] & & & \\
& & X_{F_X} \arrow[ul, "\text{in-split}"] \arrow[rr, "(L_X)_\infty"] & & X
\end{tikzcd}$$

FIG. 17.

*Proof of Proposition* 8.2. We will prove Proposition 8.2 presently, but first we remark that a proof is already contained in the literature. In [4, Lem. 2.6], it was shown that conditions (2) and (3) are equivalent. (Actually, in that paper, this result was proven only for full shifts, but essentially the same proof works in general.) In [5, Lem. 5], it was shown that conditions (1) and (3) are equivalent. So, putting these two results together we obtain Proposition 8.2. The "if" half of the "moreover" is contained in [5, Lem. 1]. The other half is easy to see.

(1) $\Rightarrow$ (3). First we show that any two paths $x_{-m} \dots x_{-1}$ and $x'_{-m} \dots x'_{-1}$ in $G$ with $\Psi(x_{-m} \dots x_{-1}) = \Psi(x'_{-m} \dots x'_{-1})$ have the same terminal state. Let

$$y_{-m} \cdots y_{-1} = \Psi(x_{-m} \dots x_{-1}) = \Psi(x'_{-m} \dots x'_{-1}).$$

Let $x'_{-m-1}$ be any edge preceding $x'_{m}$ in $G$ and let $x_0$ be any edge following $x_{-1}$ in $G$. Denote $\psi^{-1} = \Phi_{\infty}^{(m,0)}$. Then $x'_{-1} = \Phi^{(m,0)}(\Psi(x'_{-m-1})y_{-m} \cdots y_{-1})$ and $x_0 = \Phi^{(m,0)}(y_{-m} \dots y_{-1}\Psi(x_0))$. Now $\Psi(x'_{-m-1})y_{-m} \cdots y_{-1}\Psi(x_0)$ is a path in $H$, so $t(x'_{-1}) = s(x_0) = t(x_{-1})$.

Let $\mathcal{P}$ be the partition of the length-$m$ blocks of $X_H$ having, for each state $I$ of $G$, an atom

$$P_I = \{y_{-m} \cdots y_{-1} : y_{-m} \cdots y_{-1} = \Psi(x_{-m} \dots x_{-1}) \text{ and } t(x_{-1}) = I \ \}.$$

It is not hard to show that $\mathcal{P}$ is a right Markov partition. We define a label-preserving graph homomorphism $\Theta$ from $(G, \Psi)$ to $(A^{\mathcal{P}}, \Psi^{\mathcal{P}})$ by mapping states via $I \mapsto P_I$ and mapping edges via $x_0 \mapsto (P_{s(x_0)}, \Psi(x_0))$. It is not hard to show that this graph homomorphism has an inverse that maps edges by $(P_I, y_0) \mapsto \Phi^{(m,0)}(y_{-m} \dots y_{-1}y_0)$, where $y_{-m} \cdots y_{-1}$ is any element of the atom $P_I$. It follows that $\Theta$ is a graph isomorphism and that $\Psi = \Psi^{\mathcal{P}} \circ \Theta$.

(3) $\Rightarrow$ (2). We use induction on $m$. We must show that $(A^{\mathcal{P}}, \Psi^{\mathcal{P}})$ is obtained from $(H, \mathrm{id})$ by a sequence of $m$ in-splittings. A degenerate base case $m = 0$ is easy and we omit it. Suppose $m \geq 1$ and we are given a right Markov partition $\mathcal{P}$ of the length-$m$ blocks of $H$. We define a partition $\mathcal{Q}$ of the length-$(m-1)$ blocks of $H$ as follows. Declare $w_{-m+1} \dots w_{-1} \approx w'_{-m+1} \cdots w'_{-1}$ if there are edges $w_{-m}$ and $w'_{-m}$ with $w_{-m}w_{-m+1} \cdots w_{-1} \sim w'_{-m}w'_{-m+1} \cdots w'_{-1}$. (Two length-0 blocks are equivalent if and only if they have the same terminal state.) Now define $\mathcal{Q}$ to be the equivalence classes of the transitive closure $\overset{\star}{\approx}$ of the relation $\approx$. It is not hard to verify that $\mathcal{Q}$ is a right Markov partition.

We show that $(A^{\mathcal{P}}, \Psi^{\mathcal{P}})$ is obtained from $(A^{\mathcal{Q}}, \Psi^{\mathcal{Q}})$ by one round of in-splitting. In fact, given a state $[u_{-m+1} \dots u_{-1}]$ in $A^{\mathcal{Q}}$, the set

$$\{[w_{-m} \dots w_{-1}] \in \mathcal{P} : [w_{-m+1} \dots w_{-1}] = [u_{-m+1} \dots u_{-1}]\}$$

corresponds to a partition of the in-coming edges to state $[u_{-m+1} \dots u_{-1}]$. To be precise, $[w_{-m} \dots w_{-1}] \in \mathcal{P}$ corresponds to an atom

$$\{([v_{-m} \cdots v_{-2}], v_{-1}) : v_{-m} \cdots v_{-2}v_{-1} \in [w_{-m} \cdots w_{-1}]\}$$

in the partition of the in-coming edges to state $[w_{-m+1} \dots w_{-1}]$ in $A^{\mathcal{Q}}$. This gives the inductive step.

(2) $\Rightarrow$ (1). It suffices to prove the case $m = 1$, since the composition of an $(m, 0)$-block map with a $(1, 0)$-block map is a $(m + 1, 0)$-block map. Suppose $(G, \Psi)$ is obtained from $(H, \mathrm{id})$ by one round of in-splitting. Then $\Psi$ is the basic graph conjugacy associated with the in-amalgamation leading from $(G, \Psi)$ to $(H, \mathrm{id})$. Now a length-2 block $fe$ of $H$ determines the index $j$ for which $f \in P_J^j$, and therefore

determines the edge $e^j$ (the unique edge in $G$ terminating length-2 blocks $x_{-1}x_0$ with $\Psi(x_{-1}x_0) = fe$). Thus $\Psi$ has a $(1,0)$-block inverse.

Finally, we prove the "moreover." Suppose first that the 1-block conjugacy $\psi : X_G \to X_H$ has an $(m,0)$-block inverse. Then the implication (1) $\Rightarrow$ (3) shows that $\psi$ is essentially $\psi^{\mathcal{P}}$ for some right Markov partition $\mathcal{P}$. But, as noted above, $\psi^{\mathcal{P}}$ is right resolving.

Now suppose that $\psi : X_G \to X_H$ is a right-resolving conjugacy, say with inverse $\psi^{-1} = \Phi_\infty^{(m,a)}$. To show that $a$ can be taken to be 0, we must show that

$$\Phi^{(m,a)}(y_{-m}\cdots y_0y_1\cdots y_a) = \Phi^{(m,a)}(y_{-m}\cdots y_0y_1'\cdots y_a')$$

for any pair of paths $y_{-m}\cdots y_0y_1\cdots y_a$ and $y_{-m}\cdots y_0y_1'\cdots y_a'$ in the graph $H$. If $a > 0$ and if $y_1\cdots y_{a-1} = y_1'\cdots y_{a-1}'$, then the edges

$$x_0 = \Phi^{(m,a)}(y_{-m}\cdots y_0y_1\cdots y_a)$$

and

$$x_0' = \Phi^{(m,a)}(y_{-m}\cdots y_0y_1'\cdots y_a')$$

share the same initial state, namely, the terminal state of any edge in $G$ that can be expressed as $\Phi^{(m,a)}(y_{-m-1}\cdots y_0y_1\cdots y_{a-1})$ for some edge $y_{-m-1}$ preceding $y_{-m}$. Now $\Psi(x_0) = y_0 = \Psi(x_0')$ and $\Psi$ is right resolving, so $x_0 = x_0'$. This shows that, so long as $a > 0$, $a$ can be replaced by $a - 1$. By induction, we conclude that $a$ can be taken to be 0. □

*Proof of Lemma* 8.3. Define

$$f = F_\infty : S_\gamma \to S_\phi, \qquad (z,\gamma(z)) \mapsto (x,\phi(x)) \text{ where } x = (L_X)_\infty(z).$$

Then we have the commutative diagram in Fig. 18. We claim that $f$ is a right-resolving

$$\begin{tikzcd} S_\gamma \arrow[r, "f"] \arrow[d, "(\mathcal{O}_\gamma)_\infty"'] & S_\phi \arrow[d, "(\mathcal{O}_\phi)_\infty"] \\ X_{F_X} \arrow[r, "(L_X)_\infty"] & X \end{tikzcd}$$

FIG. 18.

sliding block code (recall that we defined right-resolving sliding block codes—as opposed to only right-resolving labelings—at the end of §4). To see this, first observe that we may write $f = F_\infty$ where

$$F(u,v) = (L_X(u),v).$$

Now, suppose that $(u,v)(r,s)$ and $(u,v)(r',s')$ are 2-blocks (with the same initial symbol $(u,v)$) in $S_\phi$ such that $F((r,s)) = F((r',s'))$. Then $s = s'$ and $r,r'$ are edges in $F_X$ both of which follow the edge $u$ and have the same $L_X$-label; since $L_X$ is right resolving, $r = r'$ as well, and we have $(r,s) = (r',s')$. So, $f$ is right resolving.

Since $L_\gamma$ is also right resolving, we have that the composition $F \circ L_\gamma$ is also right resolving. Since both $(L_\gamma)_\infty$ and $F_\infty$ are onto, $(F_\gamma, F \circ L_\gamma)$ is a right-resolving

$$\begin{tikzcd}
X_{F_\gamma} \arrow[r, "\pi"'] \arrow[d, "(L_\gamma)_\infty"'] & X_{F_\phi} \arrow[d, "(L_\phi)_\infty"] \\
S_\gamma \arrow[r, "f"] & S_\phi
\end{tikzcd}$$

FIG. 19.

presentation of $S_\phi$. By Theorem 4.1(part (1)), there is a right-resolving factor map $\pi : X_{F_\gamma} \to X_{F_\phi}$ such that the diagram in Fig. 19 commutes. Putting the diagrams in Figs. 18 and 19 together, we get the desired commutative diagram, Fig. 16.

It remains now only to see that $\pi$ is actually a conjugacy. For this, recall from Theorem 5.2 (part (1)) that there is a conjugacy $c = c_{(\mathcal{O}_\phi)_\infty} : X_{F_\phi} \to X_{F_X}$ such that the diagram in Fig. 20 commutes. From the diagrams in Figs. 16 and 20, we get

$$\begin{tikzcd}
X_{F_\phi} \arrow[d, "c"'] \arrow[dr, "(\bar{\mathcal{O}}_\phi)_\infty"] & \\
X_{F_X} \arrow[r, "(L_X)_\infty"] & X
\end{tikzcd}$$

FIG. 20.

$$(L_X)_\infty \circ (\bar{\mathcal{O}}_\gamma)_\infty = (\bar{\mathcal{O}}_\phi)_\infty \circ \pi = (L_X)_\infty \circ c \circ \pi.$$

Now, since $(L_X)_\infty$ is 1-1 on the left-transitive points, and since sliding block codes preserve left-transitive points, we have that

$$c \circ \pi(z) = (\bar{\mathcal{O}}_\gamma)_\infty(z)$$

for every left-transitive point $z \in X_{F_\gamma}$. But since the left-transitive points are dense, this must hold on all of $X_{F_\gamma}$:

$$c \circ \pi = (\bar{\mathcal{O}}_\gamma)_\infty. \tag{8}$$

Now, since $(\mathcal{O}_\gamma)_\infty$ is a conjugacy, $S_\gamma$ is an SFT. So, $(L_\gamma)_\infty$ is a conjugacy, and therefore so is $(\bar{\mathcal{O}}_\gamma)_\infty$. By (8), so is $\pi$. □

*Proof of Lemma* 8.4. Since the domain of $\gamma$ is a graph shift $X = X_G$, we have $(F_X, L_X) = (G, \mathrm{id})$, so $a \equiv a(\gamma) = a_{\min}(\gamma)$ and $m \equiv m(\gamma) = m_{\min}(\gamma)$. Write $d \equiv d(\gamma)$. By Proposition 7.2, $\gamma$ is an $(m, a)$-block code.

$(\bar{\mathcal{O}}_\gamma)_\infty$ is a 1-block code by definition. To say that $(\bar{\mathcal{O}}_\gamma)_\infty$ is a conjugacy whose inverse is an $(m+d, a)$-block code means: For $x \in X$ and $y = \gamma(x)$, $x_{[-(m+d),a]}$ determines the edge $e(\mathcal{F}((x,y)_{(-\infty,-1]}), (x_0, y_0))$ in $F_\gamma$. Well, clearly $x_{[-(m+d),a]}$ determines $(x_0, y_0)$. It suffices to show that $x_{[-(m+d),a-1]}$ determines $\mathcal{F}((x,y)_{(-\infty,-1]})$.

Let $x, x' \in X$ such that

$$x_{[-(m+d),a-1]} = x'_{[-(m+d),a-1]}.$$

Let

$$y = \gamma(x), y' = \gamma(x').$$

Suppose that

$$(u,v)_{[0,\infty)} \in \mathcal{F}((x,y)_{(-\infty,-1]}).$$

We must show that

$$(u,v)_{[0,\infty)} \in \mathcal{F}((x',y')_{(-\infty,-1]}).$$

Let

$$z = x_{(-\infty,-1]}u_{[0,\infty)}.$$

Then $z \in X$ and

$$\gamma(z) = y_{(-\infty,-1]}v_{[0,\infty)}.$$

Let

$$z' = x'_{(-\infty,-1]}u_{[0,\infty)}.$$

Since $0 \leq d \leq a$, we have $x_{-1} = x'_{-1}$, and so $z' \in X$. It remains to show that

$$\gamma(z') = y'_{(-\infty,-1]}v_{[0,\infty)}. \tag{9}$$

We break up the integers into four regions pictured in Fig. 21 and check (9) for coordinates $i$ in each region separately.

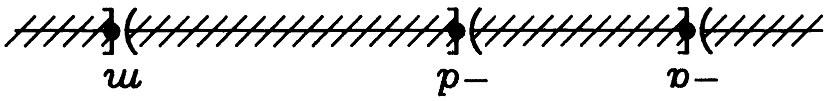


FIG. 21.

$i \in (-\infty,-a-1]$: Since $\gamma(x') = y'$, we have $\gamma(z')_{(-\infty,-a-1]} = y'_{(-\infty,-a-1]}$.

$i \in [m,\infty)$: Since $\gamma(u) = v$, we have $\gamma(z')_{[m,\infty)} = v_{[m,\infty)}$.

$i \in [-d,m-1]$: First, note that if $m+d \leq 0$, then this case does not occur. Now observe that

$$\begin{aligned} z'_{[-(m+d),a+m-1]} &= x'_{[-(m+d),-1]}u_{[0,a+m-1]} \\ &= x_{[-(m+d),-1]}u_{[0,a+m-1]} \\ &= z_{[-(m+d),a+m-1]}. \end{aligned}$$

Thus, since $\gamma$ is an $(m,a)$-block code,

$$\gamma(z')_{[-d,m-1]} = \gamma(z)_{[-d,m-1]} = y_{[-d,-1]}v_{[0,m-1]}. \tag{10}$$

Since $x_{[-(m+d),a-1]} = x'_{[-(m+d),a-1]}$, we have

$$y_{[-d,-1]} = y'_{[-d,-1]}. \tag{11}$$

Comparing (10) and (11), we see that

$$\gamma(z')_{[-d,m-1]} = y'_{[-d,-1]}v_{[0,m-1]}.$$

$i \in [-a, -d-1]$: Since $d = d(\gamma)$, $x_{(-\infty,-1]} = z_{(-\infty,-1]}$ and $\gamma(x)_{[-a,-1]} = y_{[-a,-1]} = \gamma(z)_{[-a,-1]}$, we have

$$x_{[0,a-d-1]} = z_{[0,a-d-1]}.$$

Thus,

$$x_{[0,a-d-1]} = u_{[0,a-d-1]}.$$

So,

$$\begin{aligned} z'_{[-m-a,a-d-1]} &= x'_{[-m-a,-1]}u_{[0,a-d-1]} \\ &= x'_{[-m-a,-1]}x_{[0,a-d-1]} \\ &= x'_{[-m-a,a-d-1]}. \end{aligned}$$

Thus, since $\gamma$ is an $(m,a)$-block code,

$$\gamma(z')_{[-a,-d-1]} = \gamma(x')_{[-a,-d-1]} = y'_{[-a,-d-1]}. \qquad \square$$

*Proof of Lemma* 8.5. The proof of this result will make use of the following construction.

Let $\phi_1 : X_1 \to Y$ and $\phi_2 : X_2 \to Y$ be sliding block codes on shift spaces. The *fiber product* $(Z, \psi_1, \psi_2)$ of $\phi_1, \phi_2$ is the shift space

$$Z \equiv \{(x_1, x_2) \in X_1 \times X_2 : \phi_1(x_1) = \phi_2(x_2)\}$$

together with

$$\psi_i : Z \to X_i, \qquad (x_1, x_2) \mapsto x_i, \qquad i = 1, 2.$$

We call $\phi_1, \phi_2$ the *legs* of the fiber product and $\psi_1, \psi_2$ the *arms* of the fiber product.

The following proposition is well known and easy to prove (see [3], [10]).

PROPOSITION 8.6. *Let $\phi_1 : X_1 \to Y$ and $\phi_2 : X_2 \to Y$ be sliding block codes on shift spaces. Let $(Z, \psi_1, \psi_2)$ be the fiber product of $\phi_1, \phi_2$.*

(1) *If $X_1$ and $X_2$ are graph shifts (resp., SFT, sofic), then $Z$ is a graph shift (resp., SFT, sofic).*

(2) *If a leg has any of the following properties—right resolving, right closing,* 1-1, *onto, conjugacy,* 1*-block code—then so does the opposite arm.*

For the proof of Lemma 8.5 we may assume $m > 0$, for otherwise, by Proposition 8.2, $\theta$ is a left-resolving conjugacy and $(G, \Theta)$ is obtained from $(H, \mathrm{id})$ by at most $a$ rounds of out-splitting. Now, the conclusion of Lemma 8.5 is, by virtue of Proposition 8.2, equivalent to the existence of graphs $A$ and $K$, a left-resolving conjugacy $\psi_1 : X_K \to X_A$ with a $(0,a)$-block inverse, and right-resolving conjugacies $\psi_2 : X_K \to X_G$, $\psi : X_A \to X_H$, both with an $(m,0)$-block inverses, such that the diagram in Fig. 22 commutes.

We first construct the right-resolving conjugacy $\psi : X_A \to X_H$. Then $(X_K, \psi_1, \psi_2)$ will be the fiber product of $\psi, \theta$.

The vertices of the graph $A$ will be atoms of a partition of $m$-blocks in $X_H$. In order to define this partition, we need the following result.

SUBLEMMA 8.7. *Let $\theta : X_G \to X_H$ be a* 1*-block conjugacy with an $(m,a)$-block inverse. Let $x, x' \in X_G$ such that $\theta(x)_{[-m,a-1]} = \theta(x')_{[-m,a-1]}$. Then $s(x_0) = s(x'_0)$.*

$$\begin{tikzcd}
& X_K \arrow[dl, "\psi_1"'] \arrow[dr, "\psi_2"] & \\
X_A \arrow[dr, "\psi"'] & & X_G \arrow[dl, "\theta"] \\
& X_H &
\end{tikzcd}$$

FIG. 22.

*Proof.* Let

$$y = \theta(x), y' = \theta(x'), \qquad \bar{y} = \theta(x)_{(-\infty,a-1]}\theta(x')_{[a,\infty)}.$$

Observe that $\bar{y} \in X_H$ since $X_H$ is a graph shift.

Let

$$\bar{x} = \theta^{-1}(\bar{y}).$$

Since $\theta^{-1}$ is an $(m, a)$-block code, it follows that

$$x_{-1} = \bar{x}_{-1} \quad \text{and} \quad x'_0 = \bar{x}_0.$$

Thus,

$$s(x_0) = t(x_{-1}) = t(\bar{x}_{-1}) = s(\bar{x}_0) = s(x'_0). \qquad \square$$

We continue the proof of Lemma 8.5. For an $(m + a)$-block $w$ of $X_H$, fix $y \in X_H$ such that $y_{[-m,a-1]} = w$, and let

$$s^*(w) = s(\theta^{-1}(y)_0).$$

Sublemma 8.7 shows that $s^*(w)$ is well defined independent of the choice of $y$.

For each $m$-block $w = w_1 \dots w_m$ of $X_H$, let $f_w$ denote the following function on $a$-blocks of $X_H$ outgoing from $t(w)$:

$$f_w(u) = s^*(wu).$$

Define a partition $\mathcal{P}$ on the set of $m$-blocks of $H$ by declaring $w, w'$ to belong to the same atom if and only if

$$f_w = f_{w'}.$$

Note that implicitly $f_w = f_{w'} \Rightarrow t(w) = t(w')$. As before, we say that $w \sim w'$ when $w$ and $w'$ belong to the same atom, and we denote this atom by $[w]$.

We now establish the right Markov property (5) for our partition $\mathcal{P}$. Suppose that $w \sim w'$ and $e$ follows $w$. Let $u = u_1 \dots u_a$ be an $a$-block such that $s(u) = t(e)$. We must show that

$$s^*(w_2 \dots w_m eu) = s^*(w'_2 \dots w'_m eu).$$

Since $w \sim w'$, we have

$$s^*(weu_{[1,a-1]}) = s^*(w'eu_{[1,a-1]}).$$

Now, let $v$ be any path in $H$ of length $m$ such that $s(v) = t(u)$.

Write

$$\theta = \Theta_\infty, \quad \theta^{-1} = \Pi_\infty^{m,a}.$$

For a 1-block code, such as $\theta = \Theta_\infty$, a *diamond* is a pair of distinct paths with the same initial state, terminal state, and $\Theta$-label. It is well-known that a 1-block conjugacy cannot have a diamond (see [12], [22]).

Now, the paths $\eta \equiv \Pi(weuv)$ and $\eta' \equiv \Pi(w'euv)$ in $G$ have the same initial state and the same terminal state; since $\Theta_\infty$ and $\Pi_\infty$ are inverses of one another, $\eta$ and $\eta'$ have the same $\Theta$-label. Thus, $\eta = \eta'$. In particular,

$$s^*(w_2 \dots w_m eu) = s^*(w'_2 \dots w'_m eu)$$

as desired. So, the right Markov property (5) holds. Let

$$A = A^{\mathcal{P}}, \qquad \psi = \psi^{\mathcal{P}}.$$

By Proposition 8.2, $\psi$ is a right-resolving conjugacy with a $(m, 0)$-block inverse.

Let $(X_K, \psi_1, \psi_2)$ be the fiber product of $\psi, \theta$—we are justified in writing the domain of the fiber product as a graph shift $X_K$ by Proposition 8.6 (part (1)).

By Proposition 8.6 (part (2)), since $\psi$ is a right-resolving conjugacy with an $(m, 0)$-block inverse, so is $\psi_2$, It remains to show that $\psi_1 : X_K \to X_A$ is a left-resolving conjugacy with an $(0, a)$-block inverse. This means that if $(z, x) \in X_K$, then $z_{[0,a]}$ determines $x_0$.

For this, first observe that, by the definition of $\psi = \psi^{\mathcal{P}}$, $\Psi(z_{[-m,-1]})$ is a path in $H$ that belongs to the state $s(z_0) \in \mathcal{V}(A)$, regarded as an atom of the partition $\mathcal{P}$. Thus, $s(z_0)$ and $\Psi(z_{[0,a-1]})$ uniquely determine the state $s(x_0) \in \mathcal{V}(G)$.

For the same reason, $s(z_1)$ and $\Psi(z_{[1,a]})$ uniquely determine the state $s(x_1) = t(x_0) \in \mathcal{V}(G)$.

Now, $\Psi(z_0) = \Theta(x_0)$. Thus, $z_{[0,a]}$ determines the initial state, terminal state, and $\Theta$-label of $x_0$. Since $\theta$ has no diamonds, we see that $z_{[0,a]}$ does indeed determine $x_0$.

This completes the proofs of the lemmas and therefore the proof of Theorem 8.1. □

The in-splitting corresponding to the conjugacy $\psi : X_A \to X_H$ in the proof of Lemma 8.5 is the minimal amount of in-splitting possible to reach $(G, \Theta)$ from $(H, \mathrm{id})$ by first in-splitting, then out-splitting, and finally in-amalgamating. A precise version of this is given in Proposition 8.8 below. In order to state this result, we need to say what it means for a partition $\mathcal{P}'$ on blocks of length $m'$ to refine a partition $\mathcal{P}$ on blocks of length $m$, when $m \neq m'$. Well, each partition induces a partition on blocks of length $m_0 \equiv \max(m, m')$: two blocks of length $m_0$ are in the same atom of the partition induced by $\mathcal{P}$ if their suffixes of length $m$ are in the same atom of $\mathcal{P}$ (and likewise for $\mathcal{P}'$). When we say that $\mathcal{P}'$ refines $\mathcal{P}$, we mean that the partition induced by $\mathcal{P}'$ on $m_0$-blocks partition refines the partition $\mathcal{P}$ on $m_0$-blocks.

PROPOSITION 8.8. *Let $\theta = \Theta_\infty : X_G \to X_H$ be a 1-block conjugacy from one graph shift to another such that $\theta^{-1}$ is an $(m, a)$-block code. Suppose that we have the commutative diagram of Fig. 23, where $\psi' : X_{A'} \to X_H$ is a right-resolving conjugacy, $\psi'_1 : X_{K'} \to X_{A'}$ is a left-resolving conjugacy, and $\psi'_2 : X_{K'} \to X_G$ is a 1-block code; then the partition $\mathcal{P}'$ corresponding to $\psi'$ via Proposition 8.2 refines the partition $\mathcal{P}$ corresponding to $\psi$ (where $\psi$ is as in the proof of Lemma 8.5).*

$$\begin{tikzcd} & X_{K'} \arrow[dl, "\psi'_1"'] \arrow[dr, "\psi'_2"] & \\ X_{A'} \arrow[dr, "\psi'"'] & & X_G \arrow[dl, "\theta"] \\ & X_H & \end{tikzcd}$$

FIG. 23.

*Proof.* We may assume that $\mathcal{P}'$ and $\mathcal{P}$ are partitions of blocks of the same length, and we call this length $m_0$. Let $a_0$ denote the maximum of the anticipation of $(\psi'_1)^{-1}$ and $\psi_1^{-1}$ (where $\psi_1$ is as in the proof of Lemma 8.5). For an $m_0$-block $w$ and an $a_0$-block $u$, by $s^*(wu)$, we mean $s^*(\bar{w}\bar{u})$, where $\bar{w}$ is the suffix of $w$ of length $m$ and where $\bar{u}$ is the prefix of $u$ of length $a$.

To show that $\mathcal{P}'$ refines $\mathcal{P}$, we choose two $m_0$-blocks $w$ and $w'$ in the same atom of $\mathcal{P}'$ and show that they are in the same atom of $\mathcal{P}$. To this end, let $u$ be any $a_0$-block in $X_H$ with $t(w) = s(u)$. We will show that $s^*(wu) = s^*(w'u)$, giving $f_w = f_{w'}$, whence $w$ and $w'$ are in the same atom of $\mathcal{P}$.

Suppose $y, y' \in X_{K'}$ satisfy $\psi' \circ \psi'_1(y)_{[-m_0, a_0-1]} = wu$ and $\psi' \circ \psi'_1(y')_{[-m_0, a_0-1]} = w'u$. Now

$$s(\psi'_1(y)_0) = s(\psi'_1(y')_0)$$

because both $\psi' \circ \psi'_1(y)_{[-m_0,-1]} = w$ and $\psi' \circ \psi'_1(y')_{[-m_0,-1]} = w'$ are in the same atom of $\mathcal{P}'$. Now $\psi'$ is right resolving, so

$$\psi'_1(y)_{[0,a_0-1]} = \psi'_1(y')_{[0,a_0-1]}.$$

But $(\psi'_1)^{-1}$ is an $(0, a_0)$-block map, so by Sublemma 8.7,

$$s(y_0) = s(y'_0).$$

Write $\psi'_2 = (\Psi'_2)_\infty$. Since the diagram in Fig. 23 above commutes,

$$s^*(wu) = (\Psi'_2)^*(s(y_0)) = (\Psi'_2)^*(s(y'_0)) = s^*(w'u),$$

completing the argument that $w$ and $w'$ are in the same atom of $\mathcal{P}$. □

The following corollary asserts that the in-splitting used in our proof of Theorem 8.1 is the minimal amount of in-splitting that suffices there.

COROLLARY 8.9. *Let $\phi$ be a sliding block decoder on an irreducible sofic shift $X$. Let $\gamma = \phi \circ (L_X)_\infty$. Suppose that we have the commutative diagram of Fig. 24, where $\psi' : X_{A'} \to X_{F_X}$ is a right-resolving conjugacy, $\psi'_1 : X_{K'} \to X_{A'}$ is a left-resolving conjugacy, and $\psi'_2 : X_{K'} \to X_{F_\phi}$ is a 1-block code; then the partition $\mathcal{P}'$ corresponding to $\psi'$ via Proposition 8.2 refines the partition $\mathcal{P}$ corresponding to $\psi$ (where $\psi$ is as in the proof of Lemma 8.5 for $\theta = (\bar{\mathcal{O}}_\gamma)_\infty$).*

*Proof.* $(K', \bar{\mathcal{I}}_\phi \circ \Psi'_2, \Psi' \circ \Psi'_1)$ is an encoder for $\gamma$. Thus, by Theorem 6.2, there is a graph homomorphism $M : K' \to F_\gamma$ such that the following diagram in Fig. 25 commutes. Now, apply Proposition 8.8 to $\theta = (\bar{\mathcal{O}}_\gamma)_\infty$. □

Suppose that for a finite-state code $(G, \mathcal{I}, \mathcal{O})$ with sliding block decoder $\phi$, $\mathcal{O}$ can be expressed as

$$\mathcal{O} = L_X \circ M,$$

$$\begin{tikzcd}
 & X_{K'} \arrow[dl, "\psi_1'"'] \arrow[drr, "\psi_2'"] & & \\
X_{A'} \arrow[d, "\psi'"'] & & & X_{F_\phi} \arrow[dl, "(\mathcal{O}_\phi)_\infty"'] \arrow[d, "(\bar{\mathcal{I}}_\phi)_\infty"] \\
X_{F_X} \arrow[rr, "(L_X)_\infty"] & & X \arrow[r, "\phi"'] & Y
\end{tikzcd}$$

FIG. 24.

$$\begin{tikzcd}
 & X_{K'} \arrow[dl, "\psi_1'"'] \arrow[dr, "M_\infty"] & \\
X_{A'} \arrow[dr, "\psi'"'] & & X_{F_\gamma} \arrow[dl, "(\bar{\mathcal{O}}_\gamma)_\infty"] \\
 & X_{F_X} &
\end{tikzcd}$$

FIG. 25.

where $X$ is the domain of $\phi$ and $M_\infty : X_G \to X_{F_X}$ is a left-resolving factor map. We call such a code a *left-resolving encoder.* Such codes arise in the state-splitting algorithm ([1]) and the stethering construction ([2], [8]). In the following, we see that if $\phi$ has a left resolving encoder, then $(F_\phi, \bar{\mathcal{O}}_\phi)$ is obtained from $(F_X, L_X)$ without using in-splittings.

PROPOSITION 8.10. *Let $\phi : X \to Y$ be a sliding block decoder on an irreducible sofic shift with a left-resolving encoder. Then $(F_\phi, \bar{\mathcal{O}}_\phi)$ is obtained from $(F_X, L_X)$ by a sequence of out-splittings followed by a sequence of in-amalgamations.*

*Proof.* By Proposition 8.2, the conclusion of this result is equivalent to the existence of a graph $K$, a left-resolving conjugacy $\psi_1 : X_K \to X_{F_X}$, and a right-resolving conjugacy $\psi_2 : X_K \to X_G$, such that the diagram in Fig. 26 commutes. Let $\gamma = \phi \circ (L_X)_\infty$. By Proposition 6.3, $(\bar{\mathcal{O}}_\gamma)_\infty : X_{F_\gamma} \to X_{F_X}$ is a left-resolving

$$\begin{tikzcd}
X_K \arrow[r, "\psi_2"] \arrow[d, "\psi_1"'] & X_{F_\phi} \arrow[d, "(\bar{\mathcal{O}}_\phi)_\infty"] \\
X_{F_X} \arrow[r, "(L_X)_\infty"] & X
\end{tikzcd}$$

FIG. 26.

conjugacy. Now, apply Lemma 8.3. □

Our next result shows that, in Theorem 8.1, the in-splittings can be exchanged for out-splittings if we are allowed to "shift" the sliding block decoder.

PROPOSITION 8.11. *Let $\phi$ be a sliding block decoder on an irreducible sofic shift $X$. Let $\alpha = \phi \circ \sigma^{m+d}$. Then*

(1) *There is an encoder $(F, \mathcal{I}', \mathcal{O}')$ for $\alpha$ such that $(F, \mathcal{O}')$ is obtained from $(F_X, L_X)$ by a sequence of at most $m + d + a$ out-splittings.*
(2) *$(F_\alpha, \bar{\mathcal{O}}_\alpha)$ is obtained from $(F_X, L_X)$ by a sequence of at most $m + d + a$ out-splittings, followed by a sequence of in-amalgamations.*

*Proof.* At the heart of the proof is the following simple fact: If $k$ rounds of *complete* out-splitting (resp., in-splitting) are performed on the pair $(G, \mathrm{id})$, then one obtains $(G^{[k+1]}, \Omega)$, where $\Omega(e_0 \dots e_k) = e_0$ (resp., $\Omega(e_0 \dots e_k) = e_k$).

As shown in Theorem 8.1, an encoder $(K, \mathcal{I}, \mathcal{O})$ for $\phi : X \to Y$ can be obtained from $(F_X, L_X)$ by $m + d$ rounds of in-splitting (resulting in right-resolving conjugacy $\psi : X_A \to X_{F_X}$), followed by $a$ rounds of out-splitting (resulting in a left-resolving conjugacy $\psi_1 : X_K \to X_A$). The encoder for $\phi$ is the tall commuting triangle on the right side of the commutative diagram in Fig. 27.

$$\begin{tikzcd}
X_K \arrow[r, "\sigma^{-(m+d)}"] \arrow[d, "\psi_1'"'] & X_F \arrow[r, "\eta'"] \arrow[d, "\psi_1'"'] & X_K \arrow[d, "\psi_1"'] \arrow[dddr, "\mathcal{I}_\infty"] & \\
X_{F_X}^{[m+d+1]} \arrow[r, "\sigma^{-(m+d)}"'] \arrow[drr, "\omega"'] & X_{F_X}^{[m+d+1]} \arrow[r, "\eta"'] & X_A \arrow[d, "\psi"] & \\
& & X_{F_X} \arrow[d, "(L_X)_\infty"] & \\
& & X \arrow[r, "\phi"] \arrow[dr, "\alpha"'] & Y \arrow[d, "\sigma^{(m+d)}"] \\
& & & Y
\end{tikzcd}$$

FIG. 27.

The $m+d$ rounds of in-splitting leading from $F_X$ to $A$ can be completed by further in-splitting to reach $F_X^{[m+d+1]}$. Write $\eta : X_{F_X}^{[m+d+1]} \to X_A$ for the right-resolving conjugacy corresponding to this further in-splitting. The composition $\psi \circ \eta$ is the right-resolving conjugacy induced by the edge mapping that maps the edge $e_0 \dots e_{m+d}$ in $F_X^{[m+d+1]}$ to the edge $e_{m+d}$ in $F_X$. On the other hand the graph $F_X^{[m+d+1]}$ can also be obtained by $m + d$ rounds of complete out-splitting, resulting in the left-resolving conjugacy $\omega : X_{F_X}^{[m+d+1]} \to X_{F_X}$ induced by the edge mapping $e_0 \dots e_{m+d} \mapsto e_0$. Using these explicit descriptions, it is clear that $\omega = \sigma^{-(m+d)} \circ \psi \circ \eta$. Now let $(X_F, \psi_1', \eta')$ be the fiber product of $\eta, \psi'$. It is easy to verify by chasing the commutative diagram that $\mathcal{I}_\infty \circ \eta' : X_F \to Y$ gives an input labeling on $F$ and $(L_X)_\infty \circ \omega \circ \psi_1' : X_F \to X$ gives an output labeling on $F$ that together define an encoder $(F, \mathcal{I}', \mathcal{O}')$ for $\alpha$. Now notice that $\omega \circ \psi_1'$ is a left-resolving conjuagcy obtained by at most $m+d+a$ rounds of out-splitting. So, $(F, \mathcal{O}')$ is obtained from $(F_X, L_X)$ by a sequence of at most $m+d+a$ out-splittings. This completes the proof of part (1).

For part (2), first observe that for $z \in X_F$,

$$(12) \qquad (\mathcal{I}')_\infty(z) = \mathcal{I}_\infty(\eta'(z)) \quad \text{and} \quad (\mathcal{O}')_\infty(z) = \sigma^{-(m+d)} \circ \mathcal{O}_\infty(\eta'(z)).$$

Let $g : S_\alpha \to S_\phi$ be the map defined by $g(x, y) = (\sigma^{(m+d)}(x), y)$.

Now, by Theorem 6.2 there are right-resolving graph homomorphisms $M : K \to F_\phi$ and $M' : F \to F_\alpha$, such that $(\mathcal{O}, \mathcal{I}) = L_\phi \circ M$ and $(\mathcal{O}', \mathcal{I}') = L_\alpha \circ M'$.

From (12), we have

$$g \circ (L_\alpha)_\infty \circ M'_\infty = (L_\phi)_\infty \circ M_\infty \circ \eta'.$$

From this it follows that the lift (see Theorem 4.1) $c_g : X_{F_\alpha} \to X_{F_\phi}$ of $g$ satisfies

$$c_g \circ M'_\infty = M_\infty \circ \eta'$$

(see the argument at the end of the proof of Lemma 8.3). By Theorem 8.1, we may assume that $M_\infty$ is a conjugacy. But $\eta'$ is also a conjugacy. So, $M'_\infty$ is a conjugacy as well. Thus, $(F_\alpha, \bar{\mathcal{O}}_\alpha)$ is obtained from $(F, \mathcal{O}')$ by a sequence of in-amalgamations. So, $(F_\alpha, \bar{\mathcal{O}}_\alpha)$ is obtained from $(F_X, L_X)$ by a sequence of at most $m+d+a$ out-splittings followed by a sequence of in-amalgamations. □

We remark that the estimate, $m+d+a$, of out-splittings given here is often too high. A look at the proof shows that the number of out-splittings needed to produce $F_\alpha$ is equal to the total number of splittings (out and in) needed to produce $F_\phi$.

**9. The state-splitting algorithm.** In this section, we give a brief summary of the state-splitting algorithm; see [1] and [28] for more detail.

Suppose that we are given a sofic shift $W$ and a positive integer $k$, and we wish to encode arbitrary $k$-ary sequences to sequences that satisfy the constraints of $W$ via a finite-state code $(G, \mathcal{I}, \mathcal{O})$. It is important that $\mathcal{I}_\infty(X_G)$ be the entire full $k$-shift $X_{[k]}$, so that all (right semiinfinite) $k$-ary sequences can be encoded, and that $\mathcal{O}_\infty(X_G) \subseteq W$, so that the encoded (right semiinfinite) sequences obey the constraints dictated by $W$. If the finite-state code has a sliding block decoder, then it should be defined on $X \equiv \mathcal{O}_\infty(X_G)$ but not necessarily on all of $W$.

We introduce the following definitions in order to take account of this perspective.

Let $W$ and $Y$ be shift spaces. A *finite-state* $(W, Y)$*-code* is a finite-state code $(G, \mathcal{I}, \mathcal{O})$ such that

$$\mathcal{O}_\infty(X_G) \subseteq W \quad \text{and} \quad Y = \mathcal{I}_\infty(X_G).$$

A finite-state $(W, X_{[k]})$-code is sometimes called a finite-state $(W, k)$-code.

A *sliding block* $(W, Y)$*-decoder* is a sliding block decoder that maps a subshift of $W$ onto $Y$. A sliding block $(W, X_{[k]})$-decoder is sometimes called a sliding block $(W, k)$-decoder.

Given an irreducible sofic shift $W$ and a positive integer $k$ with $h(W) \geq \log(k)$, the state-splitting algorithm constructs a finite-state $(W, k)$-code $(H, \mathcal{I}, \mathcal{O})$. The algorithm begins with a right-resolving presentation $(G, L)$ of $W$ and a nonnegative integral vector satisfying $A_G r \geq kr$ (where $A_G$ is the adjacency matrix of $G$); such a vector is called an *approximate eigenvector* (for $G$ and $k$) and $k$ is called an *approximate eigenvalue*. The entries of $r$ are called *weights*. The algorithm proceeds by a sequence of out-splittings beginning with the graph $G$; the goal is to arrive at a graph $\bar{H}$ which has a subgraph $H$ with *uniform out-degree* $k$, i.e., each state of $H$ has out-degree exactly $k$. Now, $H$ inherits an output labeling $\mathcal{O}$ from $(G, L)$ via the sequence of out-splittings, and an input labeling $\mathcal{I}$ is defined on $H$ by assigning, at each state, each $k$-ary digit to a unique outgoing edge. This defines a finite-state code $(H, \mathcal{I}, \mathcal{O})$. In general, such a code need not have a sliding block decoder. However, if $L_\infty$ is a conjugacy (e.g., if $W$ is an SFT and $(G, L)$ is the minimal right-resolving presentation), then $\mathcal{O}_\infty$ is invertible and a sliding block $(W, k)$-decoder is defined on $\mathcal{O}_\infty(X_H) \subseteq W$ by $\phi = \mathcal{I}_\infty \circ (\mathcal{O}_\infty)^{-1}$. So, if $W$ is an SFT and $h(W) \geq \log(k)$, then there is a sliding block $(W, k)$-decoder (this is the main result of [1])—results for strictly sofic shifts are contained in [20].

The key to the algorithm is the way that the out-splittings are selected. An out-splitting that constructs a graph $\bar{G}$ from a graph $G$ is said to be *legal* with respect to an approximate eigenvector $r$ (for $G$ and $k$) if there is an approximate eigenvector $\bar{r}$ (for $\bar{G}$ and $k$) such that for each state $I$ of $G$, the $\bar{r}$-components of the descendants of $I$ in $\bar{G}$ sum to the $r$-component of $I$. What this means is that it is possible to assign weights $\bar{r}_{I^i}$ to each atom $P_I^i$ of the partition which defines the splitting so that (1) the sum of the $r$-weights of the terminal states of the outgoing edges in each partition element $P_I^i$ is at least $k \cdot \bar{r}_{I^i}$ and (2) $\sum_i \bar{r}_{I^i} = r_I$. The algorithm guarantees that, for any choice of approximate eigenvector, it is always possible to find a sequence of legal out-splittings which eventually arrives at a graph with an approximate eigenvector consisting only of 0's and 1's—this is the graph $\bar{H}$ above.

Let $r$ be an approximate eigenvector for $G$ and $k$. We claim that any out-splitting of $G$ is legal with respect to $kr$. To see this, let $q_{I^i}$ be the sum of the $r$-weights of the terminal states of the edges in $P_I^i$; then assign weights $\bar{r}_{I^i}$ such that (1) each $\bar{r}_{I^i} \leq q_{I^i}$ and (2) $\sum_i \bar{r}_{I^i} = kr_I$. It follows that, in fact, any sequence of $n$ rounds of out-splitting is legal with respect to $k^n r$.

## 10. An example.

In this section, we consider, for some fixed $G$ and $k$, sliding block $(X_G, k)$-decoders where the domain of the decoder is all of $X_G$.

$G$ will be an irreducible graph with $h(X_G) = \log(5)$. In such a case, an approximate eigenvector (for $G$ and 5) must be a true eigenvector (for eigenvalue 5), and so it is unique up to scale. Thus, there is a unique smallest approximate eigenvector (for $G$ and 5), and we call it the *primitive eigenvector.* We describe three sliding block $(X_G, 5)$-decoders.

The first decoder is obtained by applying the state-splitting algorithm (in particular, only out-splittings) to the presentation $(G, \mathrm{id})$ and the primitive eigenvector $r$. This decoder has the smallest (89 states) canonical encoder of any $(X_G, 5)$-decoder. It has decoding window length $= 4$, and we will show that this minimizes the decoder window length of any decoder constructed in this way.

The second decoder is obtained by first in-splitting $G$ to obtain a graph $\hat{G}$ and then applying the state-splitting algorithm to $(\hat{G}, \Psi)$ (where $\Psi : \hat{G} \to G$ is the corresponding graph homomorphism) and the primitive eigenvector on $\hat{G}$. This decoder illustrates the use of all three graph operations in Theorem 8.1. Compared to the first decoder, its canonical enocoder is only slightly larger (91 states), but it has the significant advantage of a smaller decoding window length (3).

The third decoder is obtained by applying Proposition 8.11 to the second decoder. This is equivalent to applying the state-splitting algorithm to $G$ and the vector $5r$. So, only out-splittings are used. This yields a decoder with the same window length (3) as the second decoder, but many more states ($5 \cdot 89 = 445$) in its canonical encoder.

*Example* 10.1. In this example, $X = X_G$ where $G$ is the graph in Fig. 28 and $Y = X_{[5]}$, the full 5-shift. The primitive eigenvector components ($r$-weights) are indicated by the numbers inside the circles.

For our first decoder, we exhibit 3 rounds of legal out-splitting applied to the vector $r$.

In round 1, we split each state $I$ in row 4 into $r_I$ states, each of weight 1, by partitioning the outgoing edges into groups of 5 edges each. We can do this since these edges terminate at the top state, which has weight 1. The resulting graph looks just like $G$ in rows 1, 2, and 3, but has 70 states of weight 1 on the bottom row, with

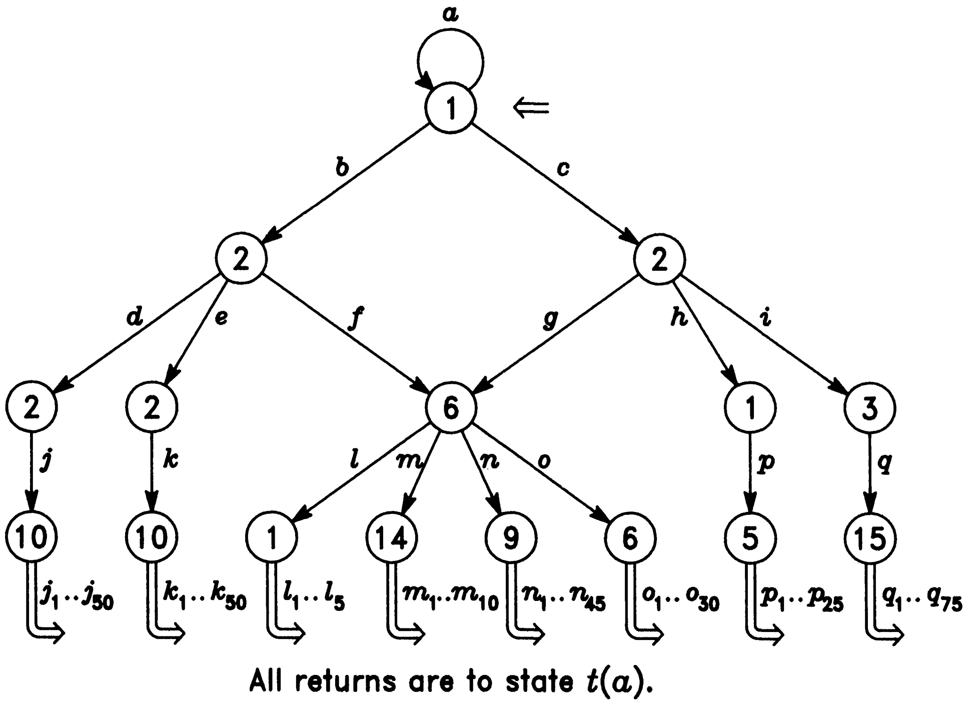


FIG. 28.

a total of 350 outgoing edges, all terminating at the top state.

At this point, the terminal states of the outgoing edges from states of row 3 have weight 1. So, in round 2, we can split the states of row 3 into states of weight 1. Finally, in round 3, we split each of the states in row 2 into 2 states each of weight 1. At this point, we have constructed a graph $H$ with uniform out-degree 5. This graph can be endowed with input and output labelings $\mathcal{I}$, $\mathcal{O}$ as described in §9.

We claim that this finite-state code has a sliding block decoder with memory 0 and anticipation 3. For this, observe that since we applied 3 rounds of out-splitting, each path $x_0x_1x_2x_3$ of length 4 in $G$ determines an edge $e$ in $H$ in the sense that all paths in $H$ which are $\mathcal{O}$-labeled $x_0x_1x_2x_3$ begin with edge $e$ (see Proposition 8.2); now, we can define $\Phi(x_0x_1x_2x_3) = \mathcal{I}(e)$.

So, this gives a sliding block decoder with window length 4. At the end of this section, we will show that any sliding block decoder obtained by applying legal out-splittings to the primitive eigenvector $r$ must have window length 4. The encoder that we have constructed has 89 states (the sum of the entries of $r$), and it is easy to see that no states can be merged. In fact, it follows from [27] that no encoder for a sliding block $(X_G, 5)$-decoder can have fewer than 89 states.

Our second decoder is obtained by 1 round of in-splitting, 2 rounds of out-splitting, and 1 round of in-amalgamation.

We first in-split the state $t(f) = t(g)$ into 2 states, resulting in the graph $\hat{G}$ in Fig. 29, with the indicated primitive eigenvector $\hat{r}$ (inherited from $G$).

We then apply the state-splitting algorithm to $\hat{G}$ and $\hat{r}$. In our first round of out-splitting, we split the two descendants, with weight 6, in $\hat{G}$ as shown in Fig. 30 below, and simultaneously we split each of the states $t(j), t(k), \ldots, t(q)$ entirely into states of weight 1 by partitioning their outgoing edges into atoms of 5 edges each (we

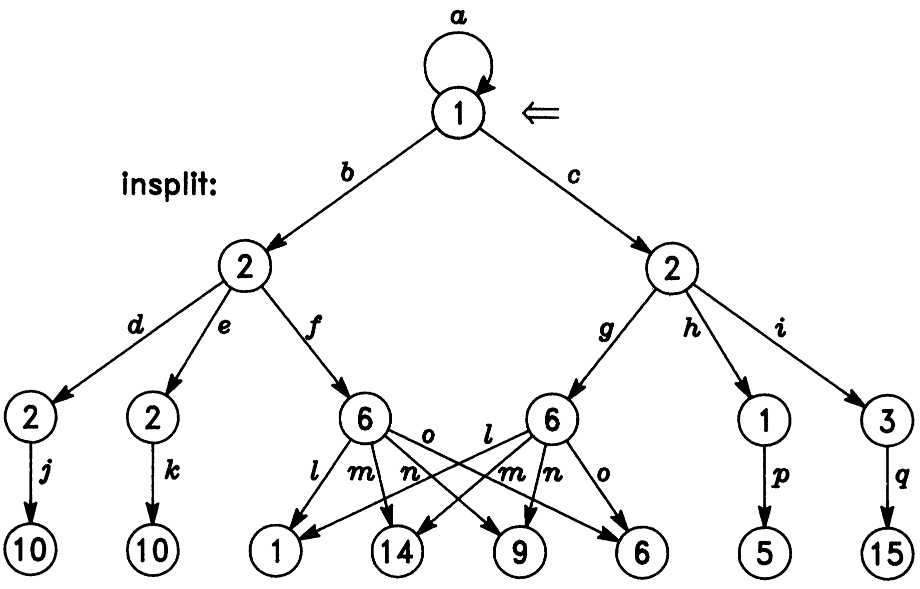


FIG. 29.

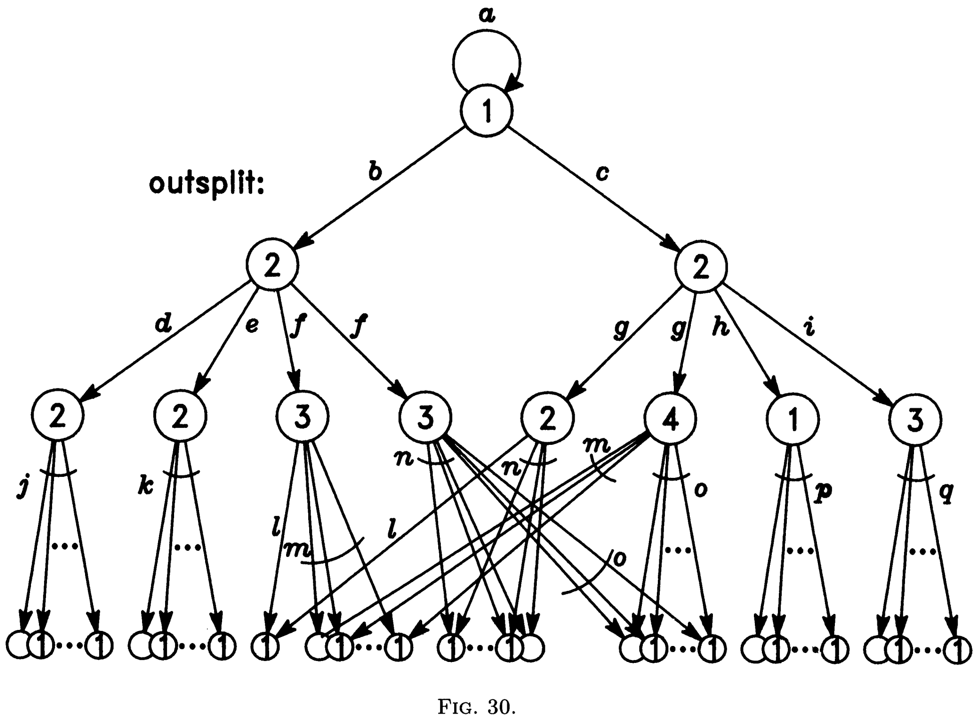
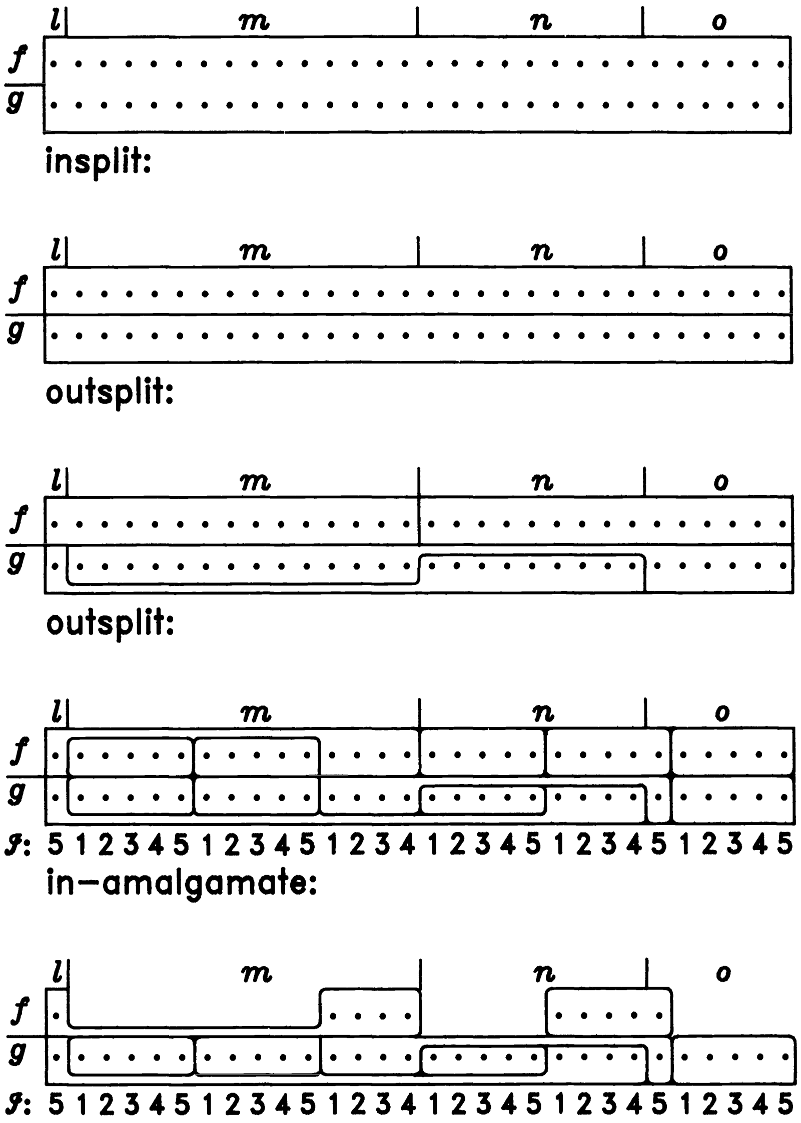


FIG. 30.

do not show this part of the graph in the figure). Call the resulting graph $G_1$.

At this point, one more round of out-splitting will yield a graph $G_2$ having uniform out-degree 5. The history of the incoming and outgoing edges to state $t(f)$ and their descendants is shown schematically in Fig. 31. The input labeling on the edges outgoing from these descendants is also shown. The input labeling on the remainder



l
m
n
o
f
g
insplit:
outsplit:
outsplit:
$\mathcal{I}$: 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
in–amalgamate:
$\mathcal{I}$: 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

FIG. 31.

of the graph is unimportant.

These in/out-splittings define a 1-block conjugacy $\mathcal{O}_\infty : X_{G_2} \to X_G$. With the indicated input labeling $\mathcal{I}$, we get a finite-state code $(G_2, \mathcal{I}, \mathcal{O})$. Now, while $(\mathcal{O}_\infty)^{-1}$ has memory 1 and anticipation 2 (because there was one round of in-splitting and two rounds of out-splitting), $\mathcal{I}$ is chosen so that the sliding block decoder $\phi \equiv \mathcal{I}_\infty \circ (\mathcal{O}_\infty)^{-1} : X_G \to X_{[5]}$ has memory 0 and anticipation 2, and thus decoding window length 3. To get the canonical encoder requires 1 round of in-amalgamation on $(G_2, \mathcal{I}, \mathcal{O})$. The number of states of this canonical encoder is the sum of the eigenvector components (89) plus 6 (because of in-splitting) minus 4 (because of in-amalgamation), which is $89 + 6 - 4 = 91$.

Now, we apply Proposition 8.11 (really the proof of Proposition 8.11) to the second decoder. We exchange the one round of in-splitting for one round of out-splitting. This gives a decoder with memory $-1$ and anticipation 3—so, it too has decoding window length 3. This decoder is obtained by 3 rounds of out-splitting; these splittings are not legal with respect to $r$, but it is not hard to see that they are legal with respect to $5r$. The number of states in the resulting encoder is the sum of the entries of this vector: $5 \cdot 89 = 445$. We leave it to the reader to verify that any encoder for a sliding block decoder on a graph shift using only out-splittings cannot be collapsed onto a smaller encoder. So, the canonical encoder for this decoder has 445 states.

Finally, we show that no sequence of out-splittings, legal with respect to the primitive eigenvector $r$, can yield a sliding block decoding window length of less than 4. We do this by showing that the anticipation of any sliding block decoder must be at least 3 and the memory cannot be negative.

We begin by showing that, if only out-splitting is used in the construction of a finite-state $(X_G, 5)$-code $(H, \mathcal{I}, \mathcal{O})$, then at least 3 rounds of out-splitting are required. For this, first observe that nontrivial legal splittings of the two states in row 2 cannot be carried out until the state $I$, in row 3, with weight 6 is split. But, as the reader can check, there is no way to split $I$ that works simultaneously for both states in row 2. Thus, we must split states in row 4 (terminal states of outgoing edges from $I$), then split state $I$, and finally split the states in row 2. So, there must be at least three rounds of out-splitting (and we saw earlier that three rounds of splitting will suffice). In fact, in Appendix II, we show that there must be at least three rounds of out-splitting, legal with respect to any approximate eigenvector.

Now, since at least 3 rounds of out-splitting are required, there are paths $e_0e_1e_2$ and $f_0f_1f_2$ in $H$ with the same initial state $I_0$, the same output labeling, but distinct initial edges. So, for any path $\eta$ which terminates at $I_0$, $\mathcal{O}(\eta e_0e_1e_2) = \mathcal{O}(\eta f_0f_1f_2)$, but $\mathcal{I}(e_0) \neq \mathcal{I}(f_0)$. From this, it follows that the anticipation must be at least 3.

If the memory were negative, then all edges in $H$ which terminate at the same state would have the same input label. Now, the unique descendant $I^*$ in $H$ of the state of $G$ in row 1 is the terminal state of all of the outgoing edges from all of the descendants of states in row 4. Thus, in fact, we see all 5 input labels on edges incoming to $I^*$. So, the memory cannot be negative. This completes the proof of the fact that no sequence of out-splittings legal with respect to the primitive eigenvector $r$ can yield a decoding window length of less than 4. So, the state-splitting algorithm, in its narrowest and most algorithmic form, can fail to find the sliding block decoders with smallest decoding window length.

## 11. Improper encoders.

In this section, we focus on finite-state $(W, Y)$-codes and sliding block $(W, Y)$-decoders where the domain of the decoder is a proper subshift of $W$. We call an encoder for such a decoder an *improper encoder.*

This is really the typical case. To see why, first observe that if $\phi : X \to Y$ is a sliding block $(W, Y)$-decoder, then since right-resolving and right-closing factor maps preserve entropy,

$$h(X) = h(\mathcal{O}_\infty(X_G)) = h(X_G) = h(\mathcal{I}_\infty(X_G)) = h(Y),$$

and thus $h(W) \geq h(X) = h(Y)$ is a necessary condition for the existence of a sliding block $(W, Y)$-decoder. If $h(W) > h(Y)$, then the domain $X$ must be a proper subshift of $W$ and any encoder must be improper.

We assume that $W$ is irreducible. While this does technically entail a loss in generality, we could always replace $W$ by an irreducible sofic subshift of maximal entropy; since entropy is not reduced, encoding into this subshift is no more difficult than encoding into $W$.

We write $X = \mathcal{O}_\infty(X_G)$, the domain of the sliding block decoder $\phi$. We assume that $X_G$, and therefore $X$, is irreducible. This assumption can be met by restricting $\mathcal{I}, \mathcal{O}$ to an irreducible component of $G$ with maximal entropy; this does not change the encoding or decoding in any essential way.

Since right-resolving factor maps preserve entropy, if $h(W) > h(Y)$, then $X_{F_X}$ will have strictly smaller entropy than $X_{F_W}$. Since $X_{F_\phi}$ is conjugate to $X_{F_X}$, it too will have strictly smaller entropy than $X_{F_W}$. Thus, since conjugacies preserve entropy, it is impossible to construct $(F_\phi, \bar{\mathcal{O}}_\phi)$ from $(F_W, L_W)$ via basic graph operations. It is natural to wonder if $(F_\phi, \bar{\mathcal{O}}_\phi)$ can be obtained from $(F_W, L_W)$ by a sequence of basic graph operations together with *pruning*, i.e., deletion of some edges. This turns out to be false. For instance, if $W$ were an SFT, then $(L_W)_\infty$ would be a conjugacy, and so, if the above were true, then $(\bar{\mathcal{O}}_\phi)_\infty$ would be a conjugacy of a graph shift onto $X$. This would force $X$ to be an SFT. But $X$ can be strictly sofic—just take $X$ to be any strictly sofic subshift of the full 2-shift; there are lots of them!

Nevertheless, there is something that can be said along these lines. In order to say this, we need the following result, which we prove below.

PROPOSITION 11.1. *Let $X \subseteq W$ be irreducible sofic subshifts. Then there are right-resolving labeled graphs $(F, L)$ and $(G, M)$ such that $(F, L)$ presents $X_{F_X}$, $(G, M)$ presents $X_{F_W}$, and $F$ is a subgraph of $G$, with the obvious imbedding $\iota : X_F \to X_G$, such that the diagram in Fig. 32 commutes.*

$$\begin{tikzcd}
X_F \arrow[rr, "\iota"] \arrow[d, "L_\infty"'] & & X_G \arrow[d, "M_\infty"] \\
X_{F_X} \arrow[dr, "(L_X)_\infty"'] & & X_{F_W} \arrow[dl, "(L_W)_\infty"] \\
& X \subseteq W &
\end{tikzcd}$$

FIG. 32.

*Moreover, if $W$ is an SFT, then $L_\infty$ can be taken to be a right-resolving conjugacy, and if $X$ is an SFT, then $M_\infty$ can be taken to be a right-resolving conjugacy.*

The preceding result asserts that $(F_X, L_X)$ can be obtained from $(F_W, L_W)$ by *expansion* (i.e., a right-resolving extension), pruning, and *merging* (i.e, a right-resolving factor). In the special case that $W$ is an SFT, the merging can, by Proposition 8.2, be taken to be a sequence of in-amalgamations. So, for any sliding block $(W, Y)$-decoder $\phi$, we can construct $(F_\phi, \bar{\mathcal{O}}_\phi)$ from $(F_W, L_W)$ by expansion, pruning, merging, and basic graph operations. So, there is a collection of operations that can be used to construct a canonical (improper) encoder.

The proof of Proposition 11.1 relies on Theorem 8.1 and the fiber product construction and its properties (see Proposition 8.6).

*Proof of Proposition* 11.1. Let $(Z, \psi_1, \psi_2)$ be the fiber product of $(L_X)_\infty$ and $(L_W)_\infty$, viewing $(L_X)_\infty$ as a map into $W$. By Proposition 8.6 (part (1)), we may write $Z = X_F$. We have the commutative diagram in Fig. 33.

$$\begin{tikzcd}
& & X_F \arrow[dll, "\psi_1"'] \arrow[drr, "\psi_2"] & & \\
X_{F_X} \arrow[dr, "(L_X)_\infty"'] & & & & X_{F_W} \arrow[dl, "(L_W)_\infty"] \\
& X & \subseteq & W &
\end{tikzcd}$$

FIG. 33.

Write

$$\psi_1 = L'_\infty, \qquad \psi_2 = M'_\infty.$$

By Proposition 8.6 (part (2)), since $(L_X)_\infty$ is right resolving, so is $\psi_2$; thus, $\psi_2$ is a right-resolving code into $X_{F_W}$. So, $(F, M')$ is a right-resolving presentation of a subshift of $X_{F_W}$. By [7, Thm. 5], there is a right-resolving presentation $(G, M)$ of $X_{F_W}$ such that $F$ imbeds as a subgraph of $G$ and $M$ extends the labeling $M'$. Take $L = L'$. This gives us the desired commutative diagram in Fig. 32.

If $W$ is an SFT, then $(L_W)_\infty$ is a right-resolving conjugacy. So, by Proposition 8.6 (part (2)), $\psi_1 = L_\infty$ is also a right-resolving conjugacy.

If $X$ is an SFT, then $(L_X)_\infty$ is 1-1. So, by Proposition 8.6 (part (2)), $\psi_2$ is also 1-1. Thus, the image of $\psi_2$ is an SFT in $X_{F_W}$.

For a graph $H$ and positive integer $m$, let $\theta_{H,m} = (\Theta_{H,m})_\infty : X_H^{[m]} \to X_H$ be the right-resolving conjugacy which simply reads off the last symbol in an $m$-block.

Now, for some $m$, the higher block shift $\psi_2(X_F^{[m]})$ is a graph shift $X_K$ in $X_{F_W}^{[m]}$. In particular, $K$ is a subgraph of $F_W^{[m]}$. Thus,

$$\beta \equiv \theta_{F_W,m}^{-1} \circ \psi_2 \circ \theta_{F,m} : X_F^{[m]} \to X_{F_W}^{[m]}$$

is a right-resolving imbedding whose image is $X_K$. So, $\beta : X_F^{[m]} \to X_K$ can be decomposed into a sequence of in-splittings. From this, it is not hard to see that $F^{[m]}$ imbeds in a graph $G$ such that $\beta$ extends to a right-resolving conjugacy $(\bar{M})_\infty : X_G \to X_{F_W}^{[m]}$.

Now, take $L = L' \circ \Theta_{F,m}$ and $M = \Theta_{F_W,m} \circ \bar{M}$. □

Assume that $h(W) > h(Y)$. So, $h(W) > h(X)$. In our passage from $(F_W, L_W)$ to $(F_\phi, \bar{O}_\phi)$ via expansion, pruning, merging, and basic graph operations, we cannot eliminate pruning, since right-resolving factor maps, as well as conjugacies, preserve entropy. In the following, we give two (very simple) examples to show that expansion and merging are necessary as well—even if we interpret pruning to mean an imbedding of a graph shift (rather than the trivial kind of imbedding obtained by deleting edges).

We will consider the labeled graphs in Fig. 34. Figure 34 (a) presents the full 2-shift, (b) presents the *even shift*, i.e., the sofic shift consisting of binary sequences where run-lengths of zeros are constrained to be even, and (c) presents the trivial shift.

In the first example, we show why expansion is necessary.

*Example* 11.2. Let $W$ be the full 2-shift and $X$ be the even shift. Let $\phi$ be the identity map. Then, we may regard (technically, only to graph isomorphism) $(F_\phi, \bar{O}_\phi) = (F_X, L_X)$. Now, suppose that we could pass from $(F_W, L_W)$ to $(F_\phi, \bar{O}_\phi) =$

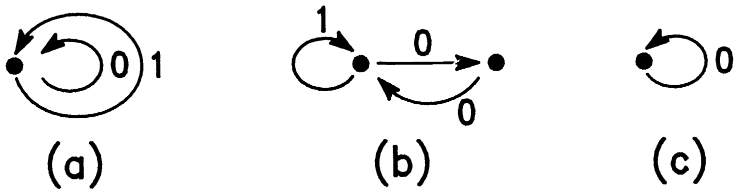


FIG. 34.

$(F_X, L_X)$ via pruning, merging, and basic graph operations only. Then there would be an imbedding $\iota : Z \to X_{[2]}$ and a factor map $\eta : Z \to X_{F_X}$ (the composition of a right-resolving factor map and conjugacies) such that the diagram in Fig. 35 commutes. Let $p = \ldots 000 \ldots \in X \subseteq W$. Let $e$ be the self-loop labeled 0 in Fig. 34 (a), and let

$$\begin{tikzcd}
Z \arrow[r, "\iota"] \arrow[d, "\eta"'] & X_{F_W} \arrow[ddl, "(L_W)_\infty"] \\
X_{F_X} \arrow[d, "(L_X)_\infty"'] & \\
X \subseteq W &
\end{tikzcd}$$

FIG. 35.

$z = \ldots eee \ldots$. Then $((L_W)_\infty)^{-1}(p) = z$. Since $\iota$ is an imbedding, $((L_W)_\infty \circ \iota)^{-1}(p)$ consists of at most one point. But $((L_X)_\infty \circ \eta)^{-1}(p) = ((L_W)_\infty \circ \iota)^{-1}(p)$, and so it too must consist of at most one point. Since $\eta$ is onto, this contradicts the fact that $p$ is presented, via $(L_X)_\infty$, by two distinct elements of $X_{F_X}$: the two phases of the periodic 2-cycle in Fig. 34 (b) above.

In our next example, we show why merging is necessary.

*Example* 11.3. Let $W$ be the even shift and $X$ be the shift space consisting of only the single point $p = \ldots 000 \ldots$. Let $\phi$ be the trivial code $\phi(p) = p$. Then we may regard (technically, only up to graph isomorphism) $(F_\phi, \bar{\mathcal{O}}_\phi)$ as the trivial labeled graph in Fig. 34 (c). Now, suppose that we could pass from $(F_W, L_W)$ to this trivial graph via expansion, pruning, and basic graph operations only. Then there would be a factor map $\gamma : X_G \to X_{F_W}$ and a point $z \in X_G$ such that $((L_W)_\infty \circ \gamma)^{-1}(p) = z$. But this contradicts the fact that $p$ has two preimages in $X_{F_W}$ via $(L_W)_\infty$.

In our final example, we exhibit an SFT $W$ such that the minimal number of states in any finite-state $(W, 2)$-code is achieved uniquely (let $(G, \mathcal{I}, \mathcal{O})$ denote this unique finite-state code). Moreover, $(G, \mathcal{I}, \mathcal{O})$ has a sliding block decoder which is a 1-block code. So, from the point of view of (1) number of encoder states and (2) sizes of decoder window length, this finite-state code is the unique "best" finite-state $(W, 2)$-code. We will see that $\mathcal{O}_\infty(X_G)$ is strictly sofic (i.e., sofic, but not SFT). So, according to the discussion earlier in this section, $(G, \mathcal{I}, \mathcal{O})$ cannot be obtained via only pruning and basic graph operations—in particular, the state-splitting algorithm, applied to $(F_W, L_W)$ and any approximate eigenvector (for $F_W$ and $k = 2$), will fail to find this code.

*Example* 11.4. Let $W$ be the SFT with alphabet $\{a, b, c\}$ defined by forbidding the blocks $\{bb, cc, bab, cac\}$. Let $(G, \mathcal{I}, \mathcal{O})$ be the transducer in Fig. 36. Here, the

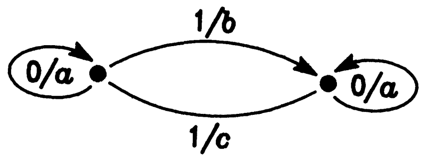


FIG. 36.

labels in front of the "/" are the $\mathcal{I}$-labels, and the labels in back of the "/" are the $\mathcal{O}$-labels.

Now, from the presentation $(G, \mathcal{O})$, we see that $X \equiv \mathcal{O}_\infty(X_G)$ is the sofic shift defined by forbidding $\{ba^n b : n \geq 0\} \cup \{ca^n c : n \geq 0\}$. It is clear that $X$ is strictly sofic and obeys the constraints dictated by $W$.

This is a finite-state $(W, 2)$-code since $(G, \mathcal{I})$ is a right-resolving presentation of the full 2-shift, $\mathcal{O}$ is right closing, and $X \subset W$.

The 1-block code $\phi = \Phi_\infty : X \to X_{[2]}$ is defined by

$$\Phi(a) = 0, \ \Phi(b) = 1, \ \Phi(c) = 1. \tag{13}$$

Now, $\phi$ is a sliding block decoder for $(G, \mathcal{I}, \mathcal{O})$ because for all edges $e$

$$\Phi \circ \mathcal{O}(e) = \mathcal{I}(e).$$

Now, by exhaustively examining all possibilities (there aren't too many) one can show that $(G, \mathcal{I}, \mathcal{O})$ is the unique finite-state $(W, 2)$-code with at most two states. Just keep in mind that $G$ must have two outgoing edges from each state (since $\mathcal{I}$ is right resolving), $(G, \mathcal{O})$ must present a subshift of $W$, and $\mathcal{O}$ must be right closing.

Let $W$ be the sofic shift and $\phi$ be the 1-block decoder in Example 11.4. Fig. 37 depicts the Fischer cover of $W$. Since $W$ is an SFT and the sliding block decoder $\phi$

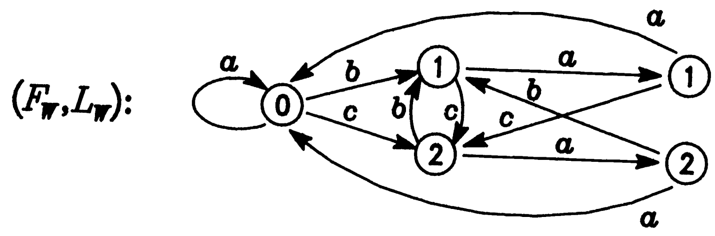


FIG. 37.

is a 1-block code, it follows from Proposition 11.1 and Theorem 8.1 that $(F_\phi, \bar{\mathcal{O}}_\phi)$ is obtained from $(F_W, L_W)$ by expansion, pruning and in-amalgamations. This sequence of steps is shown in Fig. 38.

It is interesting to note, however, that there is a different 1-block sliding block decoder obtained merely by pruning the Fischer cover of $W$: in Fig. 37 simply delete one of the edges outgoing from state 0 that is labeled $b$ or $c$, and assign an input labeling $\mathcal{I}$ that defines the same map $\Phi$ as in (13) (i.e., if $L_W(e) = a$, then $\mathcal{I}(e) = 0$,

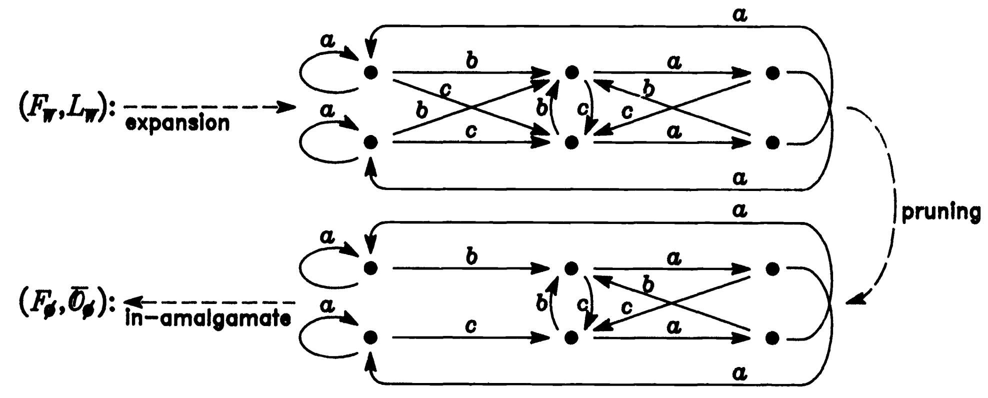


FIG. 38.

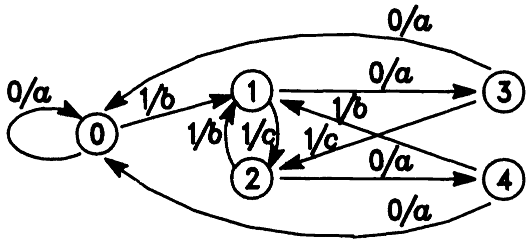


FIG. 39.

and if $L_W(e) = b$ or $c$, then $\mathcal{I}(e) = 1$); one sees that $\mathcal{I}$ is indeed right resolving, and so this gives the finite-state code with the 1-block decoder shown in Fig. 39. This is the same decoder as before except that the domain has changed; note that the new domain is an SFT.

So, here we found a 1-block decoder and then found that there was actually another 1-block decoder given by pruning the Fischer cover. This is no accident.

PROPOSITION 11.5. *Let $W$ be an irreducible sofic shift and $k$ be a positive integer. Suppose there is a finite-state $(W, k)$-code with a 1-block (i.e., $m = a = 0$) decoder. Then there is another finite-state $(W, k)$-code $(G', \mathcal{I}', \mathcal{O}')$ with a 1-block decoder where $G'$ is a subgraph of $F_W$ and $\mathcal{O}'$ restricts the labeling $L_W$.*

*Proof.* Let $(G, \mathcal{I}, \mathcal{O})$ be a finite-state $(W, k)$-code with a 1-block decoder $\phi = \Phi_\infty$. Let $(X_F, \psi_1, \psi_2)$ be the fiber product of $\mathcal{O}_\infty, L_W$. Write $\psi_1 = (\Psi_1)_\infty$ and $\psi_2 = (\Psi_2)_\infty$.

$$\begin{tikzcd}
 & & X_F \arrow[dl, "\psi_1"'] \arrow[dr, "\psi_2"] & \\
 & X_G \arrow[dl, "\mathcal{I}_\infty"'] \arrow[dr, "\mathcal{O}_\infty"] & & X_{F_W} \arrow[dl, "(L_W)_\infty"] \\
X_{[K]} & & X \subseteq W \arrow[ll, "\phi"] &
\end{tikzcd}$$

FIG. 40.

We have the commutative diagram in Fig. 40.

Since $m = a = 0$ and $\mathcal{I}$ is right resolving, it follows that $\mathcal{O}$ is right resolving. So, $\Psi_2$ is also right resolving.

For each state $J \in \mathcal{V}(F_W)$ that is in the image of $\Psi_2^*$, choose (arbitrarily) a state $I = I_J \in \mathcal{V}(F)$ such that $\Psi_2^*(I_J) = J$. Let $\mathcal{E}_I$ denote the set of edges outgoing from $I$.

Let $G'$ be the subgraph of $F_W$ which consists of the edges

$$\mathcal{E}' \equiv \cup_{J \in \{\text{image of } \Psi_2^*\}} \Psi_2(\mathcal{E}_{I_J}).$$

For $e \in \mathcal{E}'$, choose $d = d_e \in \mathcal{E}_{I_{s(e)}}$ such that $\Psi_2(d) = e$. Define

$$\mathcal{O}'(e) = L_W(e), \quad \mathcal{I}'(e) = \mathcal{I} \circ \Psi_1(d).$$

Now, we show that $(G', \mathcal{I}', \mathcal{O}')$ is a finite-state $(W, k)$-code.

First, we show that $\mathcal{I}'$ is right resolving. Let $e, f \in \mathcal{E}'$ with the same initial state. Then $d_e, d_f$ have the same initial state in $F$. Now, $\mathcal{I}$ is right resolving. Since $L_W$ is right resolving, so is $\Psi_1$. Thus, $\mathcal{I} \circ \Psi_1$ is right resolving. So, if $\mathcal{I}'(e) = \mathcal{I}'(f)$, then $\mathcal{I} \circ \Psi_1(d_e) = \mathcal{I} \circ \Psi_1(d_f)$ and so $d_e = d_f$. Thus, $e = f$. So, $\mathcal{I}'$ is right resolving.

Now, $G$ has uniform out-degree $k$; since $\psi_1$ and $\psi_2$ are right resolving, $F$ and therefore also $G'$ have uniform out-degree $k$. Thus, the image of $\mathcal{I}'_\infty$ is the entire full $k$-shift. And, by definition, the image of $\mathcal{O}'_\infty$ is contained in $W$. Since $\mathcal{I}'$ and $\mathcal{O}'$ are right resolving, $(G', \mathcal{I}', \mathcal{O}')$ is a finite-state $(W, k)$-code.

Define $\phi' : (L_W)_\infty(X_{G'}) \to X_{[k]}$ by $\phi' = \Phi_\infty$, a 1-block code. We claim that $\phi'$ is a decoder for $(G', \mathcal{I}', \mathcal{O}')$. For this, we must show that $\Phi \circ \mathcal{O}' = \mathcal{I}'$ or, equivalently, for all edges $e \in \mathcal{E}'$ $\Phi \circ L_W \circ \Psi_2(d_e) = \mathcal{I} \circ \Psi_1(d_e)$. But this follows from the definition of the fiber product and the fact that $\phi$ is a decoder for $(G, \mathcal{I}, \mathcal{O})$. □

The main result of §12 is a generalization of the preceding result for arbitrary memory and anticipation.

Of course, we know from the discussion above that the decoder, defined by the finite-state code in Fig. 39, cannot have an encoder with less than three states. In fact, by merging states in Fig. 39, we get the canonical (3-state) encoder shown in Fig. 41.

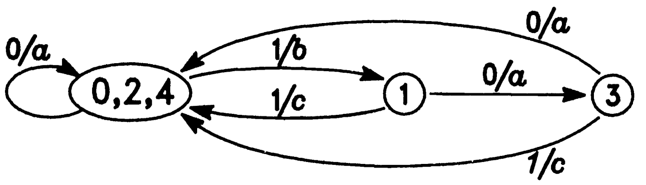


FIG. 41.

The following result organizes the operations used to pass from $(F_W, L_W)$ to $(F_\phi, L_\phi)$ in a different way than explained in the discussion following the statement of Proposition 11.1.

THEOREM 11.6. *Let $X \subseteq W$ be irreducible sofic subshifts. Let $\phi : X \to Y$ be a sliding block $(W, Y)$-decoder. Then $(F_\phi, \bar{\mathcal{O}}_\phi)$ is obtained from $(F_W, L_W)$ by a right-resolving extension, a left-resolving conjugacy, pruning of edges and a right-resolving factor, i.e., there are graphs $G_1, G_2$, and $G_3$ such that $G_2 \supseteq G_3$, a right-resolving factor map $\alpha_1 : X_{G_1} \to X_{F_W}$, a left-resolving conjugacy $\alpha_2 : X_{G_2} \to X_{G_1}$, and a right-resolving factor map $\alpha_3 : X_{G_3} \to X_{F_\phi}$ such that the diagram in Fig. 42 commutes. Moreover, if $W$ is an SFT, then $\alpha_3$ can be taken to be a right-resolving conjugacy.*

$$\begin{tikzcd}
X_{G_3} \subseteq X_{G_3} \arrow[rr, "\alpha_2"] \arrow[ddd, "\alpha_3"'] & & X_{G_1} \arrow[dr, "\alpha_1"] & \\
& & & X_{F_W} \arrow[dd, "(L_W)_\infty"] \\
& & & \\
X_F \arrow[rrr, "(\mathcal{O}_\phi)_\infty"] & & & X \subseteq W
\end{tikzcd}$$

FIG. 42.

The proof of this result is based on Proposition 11.1, Theorem 8.1, fiber products, and the fact that for any right- (resp., left-) resolving conjugacy $\phi : X_K \to X_F$ and any graph $G$ that contains $F$, there is a graph $K'$ containing $K$ such that $\phi$ can be extended to a right- (resp., left-) resolving conjugacy $X_{K'} \to X_F$. We leave the proof as an exercise for the reader.

COROLLARY 11.7. *Let $X \subseteq W$ be irreducible sofic subshifts. Let $\phi : X \to Y$ be a sliding block $(W, Y)$-decoder. Then $(F_\phi, \bar{\mathcal{O}}_\phi)$ is obtained from $(F_W, L_W)$ by expansion, out-splitting, pruning, and then merging.*

We remark that the steps of expansion, out-splitting, and pruning already give an encoder. Merging simplifies this encoder.

From Theorem 4.1, we know that the right-resolving presentations of $W$ are the same as the expansions of $(F_W, L_W)$. Recall from §9 that every sequence of out-splittings is legal with respect to some approximate eigenvector. Thus, Corollary 11.7 says that if we are allowed to start with any right-resolving presentation $(G, L)$ of $W$ and any approximate eigenvector for $G$ and $k$, then the state-splitting algorithm will find every sliding block $(W, k)$-decoder.

**12. State splitting without expansion.** Proposition 11.5 shows that if $W$ is an irreducible sofic shift having a finite-state $(W, k)$-code $(G, \mathcal{I}, \mathcal{O})$ with a 1-block decoder $\phi = \Phi_\infty$, then in fact we can construct a finite-state $(W, k)$-code $(G', \mathcal{I}', \mathcal{O}')$ with a 1-block decoder $\phi' = \Phi'_\infty$ starting from $(F_W, L_W)$ by pruning alone. No expansion is needed! Moreover, $\Phi$ and $\Phi'$ are identical as functions of the 1-blocks of $W$. The sliding block decoders $\phi$ and $\phi'$ differ only in the strict mathematical sense that their domains $\mathcal{O}_\infty(X_G)$ and $\mathcal{O}'_\infty(X_{G'})$ might be different subshifts of the sofic subshift $W$. This is a distinction without any operational significance to the engineer who will design a circuit to implement the decoder (of course, there may be a vast difference in the encoders).

We generalize this result as follows.

PROPOSITION 12.1. *Let $W$ be an irreducible sofic shift and let $k$ be a positive integer. Suppose there is a finite-state $(W, k)$-code $(G, \mathcal{I}, \mathcal{O})$ with a sliding block $(W, k)$-decoder $\phi = \Phi_\infty^{m,a} : X \to X_{[k]}$ having memory $m$, anticipation $a$, and delay $d$. Then there is another finite-state $(W, k)$-code $(G', \mathcal{I}', \mathcal{O}')$ with a sliding block $(W, k)$-decoder $\phi' : X' \to X_{[k]}$, where $\phi' = \Phi_\infty^{m,a}$ and $(G', \mathcal{O}')$ is obtained from $(F_W, L_W)$ by at most $m + d$ rounds of in-splitting, followed by at most $a$ rounds of out-splitting, followed by pruning. Moreover, $\phi'$ has memory $m$, anticipation $a$, and delay $d$.*

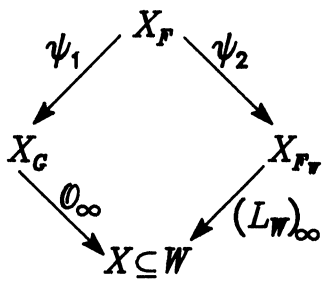
Thus the operation of expansion is not needed to obtain *some* encoder having a sliding block decoder $\phi$ specified (except for its domain) by the finite-block function $\Phi$. What is more, if one is willing to compose the given decoder $\phi$ with a power of the shift map $\sigma$ then by Proposition 8.11, in-splitting can be avoided as well.

COROLLARY 12.2. *Let $W$ be an irreducible sofic shift and let $k$ be a positive integer. Suppose there is a finite-state $(W,k)$-code $(G,\mathcal{I},\mathcal{O})$ with a sliding block $(W,k)$-decoder $\phi = \Phi_\infty^{m,a} : X \to X_{[k]}$ having memory $m$, anticipation $a$, and delay $d$. Then there is another finite-state $(W,k)$-code $(G'',\mathcal{I}'',\mathcal{O}'')$ with a sliding block decoder $\phi'' : X'' \to X_{[k]}$, where $\phi'' = \Phi_\infty^{-d,a+m+d}$ and $(G'',\mathcal{O}'')$ is obtained from $(F_W, L_W)$ by $m+a+d$ rounds of out-splitting followed by pruning.*

We conclude that out-splitting and pruning are, taken together, powerful enough tools to construct $(W,k)$-codes from $(F_W, L_W)$ having the shortest possible decoder window length. However, as Examples 10.1 and 11.4 show, we avoid in-splitting and expansion at the peril of greater encoder complexity.

Recall from §9 that any sequence of $n$ rounds of out-splitting is legal with respect to the scaled-up approximate eigenvector $k^n\mathbf{r}$ (where $k$ is the approximate eigenvalue and $\mathbf{r}$ is any smallest approximate eigenvector (in terms of maximal component)). Corollary 12.2 gives an upper bound on this scaling factor in terms of the parameters $d$, $m$, and $a$ of the sliding block $(W,k)$-decoder $\phi$. The bound is $k^{m+a+d}$. Now if $s$ is the number of states in the constraint graph $F_W$, then [6] gives an upper bound of $k^{2s}$ on the largest entry of a smallest approximate eigenvector $\mathbf{r}$. The state-splitting algorithm will operate by a series of at most $m+a+d$ rounds of legal splitting on $F_W$ (with an initial approximate eigenvector whose components are dominated by $k^{2s+m+a+d}$) to produce a finite-state $(W,k)$-code having sliding block decoder $\phi = \Phi_\infty^{-d,a+m+d}$ and having no more than $sk^{2s+a+m+d}$ states.

*Proof of Proposition* 12.1. Let $(G,\mathcal{I},\mathcal{O})$ be a finite-state $(W,k)$-code having a sliding block decoder $\phi = \Phi_\infty^{m,a} : X \to X_{[k]}$. Let $(X_F, \psi_1, \psi_2)$ be the fiber product of $\mathcal{O}_\infty, (L_W)_\infty$. We have the commutative diagram in Fig. 43.



$$\begin{tikzcd} & X_F \arrow[dl, "\psi_1"'] \arrow[dr, "\psi_2"] & \\ X_G \arrow[dr, "\mathcal{O}_\infty"'] & & X_{F_W} \arrow[dl, "(L_W)_\infty"] \\ & X \subseteq W & \end{tikzcd}$$

FIG. 43.

Write $\psi_1 = (\Psi_1)_\infty$ and $\psi_2 = (\Psi_2)_\infty$. Note that $(F, \mathcal{I}\circ\Psi_1, L_W\circ\Psi_2)$ is a $(W,k)$-code having sliding block decoder $\phi$. Define $\psi_2^{[m+a+1]} : X_{F^{[m+a+1]}} \to X_{F_W^{[m+a+1]}}$ by $\psi_2^{[m+a+1]} = (\Psi_2^{[m+a+1]})_\infty$, where

$$\Psi_2^{[m+a+1]}(x_{-m-d} \cdots x_{a-d}) = \Psi_2(x_{-m-d}) \dots \Psi_2(x_{a-d}).$$

Apply $m+d$ rounds of complete in-splitting followed by $a-d$ rounds of complete out-splitting to $(F_W, L_W)$ to obtain $(F_W^{[m+a+1]}, L)$, where $L(y_{-m-d} \cdots y_{a-d}) = L_W(y_0)$.

Let $H \subseteq F_W^{[m+a+1]}$ be the subgraph remaining after pruning away all of the edges of $F_W^{[m+a+1]}$ that are not in the image of $\Psi_2^{[m+a+1]}$ and then pruning away any stranded states. Notice that the image of $\psi_2^{[m+a+1]}$ is actually contained in $X_H$.

We will define a left Markov equivalence relation $\approx$ on the $d$-blocks of $X_H$ in two stages. First we define an equivalence relation $\sim$ on the edges of $H$. Let $y_{-m-d} \dots y_{a-d}$ and $y'_{-m-d} \dots y'_{a-d}$ be two $(m+a+1)$-blocks of $F_W$ that are in the image of $\Psi_2^{[m+a+1]}$. (These are edges of $H$.) We declare

$$y_{-m-d} \dots y_{a-d} \sim y'_{-m-d} \dots y'_{a-d}$$

if there are $(m+a+1)$-blocks $x_{-m-d} \dots x_{a-d}$ and $x'_{-m-d} \dots x'_{a-d}$ of $F$ such that

$$x_{-m-d} \dots x_{-d} = x'_{-m-d} \dots x'_{-d},$$

$$\Psi_2^{[m+a+1]}(x_{-m-d} \dots x_{a-d}) = y_{-m-d} \dots y_{a-d},$$

and

$$\Psi_2^{[m+a+1]}(x'_{-m-d} \dots x'_{a-d}) = y'_{-m-d} \dots y'_{a-d}.$$

Now we extend the relation $\sim$ to an equivalence relation by taking its transitive closure.

If

$$y_{-m-d} \dots y_{a-d} = \Psi_2^{[m+a+1]}(x_{-m-d} \dots x_{-d} x_{-d+1} \dots x_{-d+a})$$

and

$$y'_{-m-d} \dots y'_{a-d} = \Psi_2^{[m+a+1]}(x_{-m-d} \dots x_{-d} x'_{-d+1} \dots x'_{-d+a}),$$

then clearly

$$y_{-m-d} \dots y_{-d} = y'_{-m-d} \dots y'_{-d}.$$

But more is true. By Proposition 7.4, the edge $x_{-d}$ of the encoder graph $F$ determines $L_W \circ \Psi_2(x_{-d+1} \dots x_{-2d+a})$. But $L_W$ is right resolving. So the edge $x_{-d}$ determines $\Psi_2(x_{-d+1} \dots x_{-2d+a})$. So in fact

$$y_{-m-d} \dots y_{-2d+a} = y'_{-m-d} \dots y'_{-2d+a}.$$

Thus if

$$y_{-m-d} \dots y_{a-d} \sim y'_{-m-d} \dots y'_{a-d},$$

then

$$y_{-m-d} \dots y_{-2d+a} = y'_{-m-d} \dots y'_{-2d+a}.$$

Now let $u_1 \dots u_d$ and $u'_1 \dots u'_d$ be two $d$-blocks of $X_H$. We define $u_1 \dots u_d \approx u'_1 \dots u'_d$ if $u_i \sim u'_i$ for $1 \le i \le d$. We show that $\approx$ satisfies the left Markov property. We can regard $u_1 \dots u_d$ as an $(m+a+d)$-block $y_{1-m-d} \dots y_a$ of $F_W$, each $(m+a+1)$-subblock of which is in the image of $\Psi_2^{[m+a+1]}$. Similarly, we can regard $u'_1 \dots u'_d$ as an $(m+a+d)$-block $y'_{1-m-d} \dots y'_a$ of $F_W$. Since $u_i \sim u'_i$ we have $y_{i-m-d} \dots y_{i-2d+a} = y'_{i-m-d} \dots y'_{i-2d+a}$ for $1 \le i \le d$. So $y_{1-m-d} \dots y_{a-d} = y'_{1-m-d} \dots y'_{a-d}$. Now, if $u_0$ is any edge preceding the edge $u_1$ in $H$, then $u_0 = y_{-m-d} \dots y_{a-d}$ for some edge $y_{-m-d}$

of $F_W$ such that $y_{-m-d}\dots y_{a-d}$ is in the image of $\Psi_2^{[m+a+1]}$. So $u_0$ also precedes the edge $u'_1$. Hence $u_0u_1\dots u_{d-1}\approx u_0u'_1\dots u'_{d-1}$.

Let $\pi : X_{H'}\to X_H$ be the left-resolving conjugacy corresponding to the left Markov equivalence relation $\approx$, according to Proposition 8.2. Each edge of $H'$ can be regarded as an $\approx$-equivalence class, where $\approx$ is regarded as an equivalence relation on that set of $(m+d+a+1)$-blocks of $F_W$ all of whose $(m+a+1)$-subblocks are in the image of $\Psi_2^{[m+a+1]}$. Here, $y_{-m-d}\dots y_a\approx y'_{-m-d}\dots y'_a$ if and only if $y_{i-m-d}\dots y_{i-d+a}\sim y'_{i-m-d}\dots y'_{i-d+a}$ for all $0\le i\le d$. We now use this view of the edges of $H'$ to define two labelings $\mathcal{I}'$ and $\mathcal{O}'$ on the edges. These will be the input and output labelings when they are restricted to a subgraph $G'$ of $H'$.

First we define the labeling $\mathcal{I}'$. For the edge $z_0$ of $H'$ choose any representative $(m+d+a+1)$-block $y_{-m-d}\dots y_a$ in $F_W$ and set $\mathcal{I}'(z_0)=\Phi\circ L_W(y_{-m}\dots y_a)$. We must verify that $\mathcal{I}'$ is well defined. Suppose that $y'_{-m-d}\dots y'_a$ is another representative $(m+d+a+1)$-block of the edge $z_0$. We must show that $\Phi\circ L_W(y_{-m}\dots y_a)=\Phi\circ L_W(y'_{-m}\dots y'_a)$. We can assume that there are $(m+a+1)$-blocks $x_{-m}\dots x_a$ and $x'_{-m}\dots x'_a$ in $X_F$ such that $x_{-m}\dots x_0=x'_{-m}\dots x'_0$, $\Psi_2^{[m+a+1]}(x_{-m}\dots x_a)=y_{-m}\dots y_a$, and $\Psi_2^{[m+a+1]}(x'_{-m}\dots x'_a)=y'_{-m}\dots y'_a$, because the equivalence relation $\sim$ is the transitive closure of pairs of $(m+a+1)$-blocks of $X_{F_W}$ related in this way. We have

$$\Phi\circ L_W(y_{-m}\dots y_a)=\mathcal{I}\circ\Psi_1(x_0)=\mathcal{I}\circ\Psi_1(x'_0)=\Phi\circ L_W(y'_{-m}\dots y'_a),$$

showing that $\mathcal{I}'$ is well defined.

Now define a labeling $\mathcal{O}'$ on $H'$ as follows. We regard an edge $z_0$ of $H'$ as an $\approx$-equivalence class of $(m+d+a+1)$-blocks in $F_W$. All of the representative $(m+d+a+1)$-blocks of $z_0$ have identical initial $(m+a+1)$-blocks, say $y_{-m-d}\dots y_{a-d}$. Define $\mathcal{O}'(z_0)=L_W(y_0)$. This is the labeling inherited from $(F_W^{[m+a+1]},L)$ after the edge deletions and the $d$ rounds of out-splitting leading from $H$ to $H'$. Since $\phi=\Phi_\infty^{m,a}$, $\phi$ naturally extends (as an $(m+a+1)$-block map) to all of $\mathcal{O}'_\infty(X_{H'})$. We now show that $\mathcal{I}'_\infty=\phi\circ\mathcal{O}'_\infty$ as functions from $X_{H'}$ to $X_{[k]}$. Say $z\in X_{H'}$. Then $\mathcal{O}'(z)_{[-m,a]}=L_W(y_{-m}\dots y_a)$, where for $-m\le i\le a$, $y_{i-m-d}\dots y_{i+a}$ is a representative $(m+d+a+1)$-block of the edge $z_i$. So

$$\phi\circ\mathcal{O}'_\infty(z)_0=\Phi\circ L_W(y_{-m}\dots y_a)=\mathcal{I}'(z_0).$$

Finally, we show that there is an irreducible subgraph $G'\subseteq H'$ having at each state $k$ out-going edges that are distinctly $\mathcal{I}'$-labelled. Thus $(G',\mathcal{I}',\mathcal{O}')$ is a finite-state $(W,k)$-code having a sliding block decoder $\phi'$ defined by the same function $\Phi$ on $(m+a+1)$-blocks that defines $\phi : X\to X_{[k]}$.

Fix some $(d+m+1)$-block $x_{-m-d}\dots x_0$ in $X_F$. Let $x_1^{(1)},\dots,x_1^{(k)}$ be the edges in $F$ that follow $x_0$. Say they are input-labeled $\mathcal{I}\circ\Psi_1(x_1^{(j)})=j$, for $1\le j\le k$. For each $1\le j\le k$, fix an $a$-block $x_2^{(j)}\dots x_{a+1}^{(j)}$ in $X_F$ with $t(x_1^{(j)})=s(x_2^{(j)})$. Write $y_{-m-d}^{(j)}\dots y_{a+1}^{(j)}=\Psi_2(x_{-m-d}\dots x_0x_1^{(j)}\dots x_{a+1}^{(j)})$. Notice that, having fixed $x_{-m-d}\dots x_0$, there is a unique edge $z_0$ of $H'$ such that if $x'_1\dots x'_a$ is *any* $a$-block following $x_{-m-d}\dots x_0$ in $F$, then $y'_{-m-d}\dots y'_a=\Psi_2(x_{-m-d}\dots x_0x'_1\dots x'_a)$ is a representative of the edge $z_0$, where $z_0$ is regarded as an $\approx$-equivalence class. In particular, $y_{-m-d}^{(j)}\dots y_a^{(j)}$ for $1\le j\le k$ are all representatives of $z_0$. It follows from this that the edge $z_1^{(j)}$ of $H'$ that, as an $\approx$-equivalence class, contains $y_{-m-d+1}^{(j)}\dots y_{a+1}^{(j)}$, follows the edge $z_0$. Now

$$\mathcal{I}'(z_1^{(j)}) = \mathcal{I} \circ \Psi_1(x_1^{(j)}) = j.$$

In summary, for any $(m+d+1)$-block $x_{-m-d} \dots x_0$ in $X_F$, there is a unique edge $z_0$ in $H'$ that, as a $\approx$-equivalence class, contains the set

$$\{\Psi_2(x_{-m-d} \dots x_0 x_1' \dots x_a') : x_1' \dots x_a' \text{ follows } x_0 \text{ in } F\}.$$

Moreover, $z_0$ is followed in $H'$ by $k$ distinctly $\mathcal{I}'$-labelled edges $z_1^{(1)}, \dots, z_1^{(k)}$. The same argument applies to each of these $k$ edges to show again that each is followed in $H'$ by $k$ edges that are distinctly $\mathcal{I}'$-labelled. In this way, we can generate a subgraph of $H'$ having at each state exactly $k$ out-going edges, and these are distinctly $\mathcal{I}'$-labeled. Finally we take $G'$ to be an irreducible component of this subgraph keeping full out-degree $k$.

To summarize the construction of $G'$, we started with $F_W$, took the higher block presentation $F_W^{[m+a+1]}$, pruned to get $H$, out-split to get $H'$, and finally pruned to get $G'$. The higher block presentation $F_W^{[m+a+1]}$ was obtained from $F_W$ by $m+d$ rounds of complete in-splitting followed by $a-d$ rounds of complete out-splitting. The pruning done on $F_W^{[m+a+1]}$ to leave $H$ can be postponed until after the $d$ rounds of out-splitting leading from $H$ to $H'$. In starting the $d$ rounds of out-splitting on $F_W^{[m+a+1]}$ rather than its subgraph $H$, one has only to make sure that at each round of out-splitting, the descendant edges are partitioned in a way that restricts to a corresponding partition of the descendants of the edges of the subgraph $H$ when $H$ is out-split in isolation. How edges that are not descendants of edges of the subgraph $H$ are distributed in each successive partition is unimportant, for their descendants will ultimately be pruned away to leave $H'$. □

As a corollary of Proposition 12.1, we give a necessary condition for a sliding block $(W, k)$-decoder to have a given delay.

In order to state the result, we need to introduce the notion of the higher-power graph $G^n$, defined as the graph with the same set of states as $G$ and an edge from $I$ to $J$ for each path of length $n$ in $G$ from $I$ to $J$.

COROLLARY 12.3. *Let $W$ be an irreducible sofic shift and $k$ be a positive integer. If there is a finite-state $(W, k)$-code with an $(m, a)$-block decoder having delay $d$, then there is a graph $E$ such that*

(1) *$E$ has uniform out-degree $k^d$,*

*and*

(2) *$E$ is obtained from $(F_W)^d$ by pruning and one (round of) out-splitting.*

In [8], it is shown that in (1) and (2) above, $d$ can be replaced by the delay of the output-labeling $\mathcal{O}$ of the finite-state $(W, k)$-code. Using Proposition 7.4, it is not hard to see that $a - d \le d(\mathcal{O}) \le a$. However, neither $d$ nor $d(\mathcal{O})$ necessarily dominates the other. There are more necessary conditions in [27].

*Proof of Corollary* 12.3. We continue using the notation used in the proof of Proposition 12.1. The graph $G'$ in that proof can be constructed from $F_W$ by completely in-splitting $(m+d)$ rounds, completely out-splitting $(a-d)$ rounds, out-splitting $d$ more rounds, and finally pruning. Let $G''$ be the graph containing $G'$ before the pruning is done. Then $G''$ contains an irreducible subgraph $(G')$ having out-degree $k$.

We regard the states of $G''$ as the equivalence classes of $(m+d+a)$-blocks of $F_W$ defined by the splittings leading from $F_W$ to $G''$. If two $(m+d+a)$-blocks $y_{-m-d} \cdots y_{a-1}$ and $y'_{-m-d} \cdots y'_{a-1}$ of $F_W$ are equivalent, then $y_{-m-d} \cdots y_{a-d-1} =$

$y'_{-m-d}\cdots y'_{a-d-1}$. Thus, each $(a+m)$-block $y_{-m-d}\dots y_{a-d-1}$ defines a partition $\mathcal{P}_{y_{-m-d}\cdots y_{a-d-1}}$ of the $d$-blocks $y_{a-d}\cdots y_{a-1}$ that follow it (according to the equivalence class of $y_{-m-d}\cdots y_{a-1}$). Now define a vector $\mathbf{r}$ indexed by the states of $F_W$ as

$$r_J = \max\left\{|\mathcal{P}_{y_{-m-d}\cdots y_{a-d-1}}| : \begin{array}{c} y_{-m-d}\cdots y_{a-d-1} \text{ is in the image of } \Psi_2 \\ \text{and } t(y_{a-d-1}) = J \end{array}\right\}$$

if $J$ is in the image of $\Psi_2^*$, and $r_J = 0$ otherwise. If $J$ is in the image of $\Psi_2^*$, set $\mathcal{P}_J = \mathcal{P}_{y_{-m-d}\cdots y_{a-d-1}}$, where $y_{-m-d}\cdots y_{a-d-1}$ is in the image of $\Psi_2$, $t(y_{a-d-1}) = J$, and $|\mathcal{P}_J| = r_J$. If a state $J$ of $F_W$ is not in the image of $\Psi_2^*$, then define $\mathcal{P}_J$ arbitrarily. Now let $A \in \mathcal{P}_J = \mathcal{P}_{y_{-m-d}\cdots y_{a-d-1}}$, where the set of paths

$$y_{-m-d}\cdots, y_{a-d-1}A = \{y_{-m-d}\cdots y_{a-1} : y_{a-d}\cdots y_{a-1} \in A\}$$

intersects the image of $\Psi_2$. The set of paths $y_{-m-d}\dots y_{a-d-1}A$ can be identified with a state $I$ of $G''$ that is in the subgraph $G'$. As such, there are at least $k^d$ paths of length $d$ in $G''$ emanating from $I$ , each path terminating at a state $y_{-m}\cdots y_{a-1}A'$ where $y_{-m-d}\cdots y_{a-1} \in y_{-m-d}\cdots y_{a-d-1}A$ and $A' \in \mathcal{P}_{y_{-m}\cdots y_{a-1}}$. Thus

$$\begin{aligned}&\Sigma[r_{t(y_{a-d}\cdots y_{a-1})} : y_{a-d}\cdots y_{a-1} \in A]\\ &\geq \Sigma[|\mathcal{P}_{y_{-m}\cdots y_{a-1}}| : y_{-m}\cdots y_{a-1} \text{ is in the image of } \Psi_2 \text{ and } y_{a-d}\cdots y_{a-1} \in A]\\ &\geq k^d.\end{aligned}$$

It follows that

$$\Sigma[r_{t(y_{a-d}\cdots y_{a-1})} : s(y_{a-d}\cdots y_{a-1}) = J] \geq r_J k^d$$

for each state $J$ of $F_W$ in the image of $\Psi_2^*$. It follows that if we out-split $(F_W)^d$ according to the partitions $\mathcal{P}_J$, we obtain a graph containing an irreducible subgraph $E$ having out-degree at least $k^d$. □

## 13. Appendix I. *Proof of Proposition* 4.4:

$d(f_2 \circ f_1) \leq d(f_2) + d(f_1)$: Write $f_1 = (F_1)_\infty$ and $f_2 = (F_2)_\infty$. We can assume that $D = d(f_2) + d(f_1)$ is finite. Let $w = w_0 \dots w_D$ and $w' = w'_0 \dots w'_D$ be $(D+1)$-blocks in $X_G$ with $s(w) = s(w')$ and $F_2 \circ F_1(w) = F_2 \circ F_1(w')$. Let $F_1(w)_{[0,k]}$ be the longest common prefix of $F_1(w)$ and $F_1(w')$. In case $k < D$, $F_1(w)_{[k+1,D]}$ and $F_1(w')_{[k+1,D]}$ have the same initial state in $H$, distinct first edges, and $F_2 \circ F_1(w)_{[k+1,D]} = F_2 \circ F_1(w')_{[k+1,D]}$. So $D - k \leq d(f_2)$, giving $d(f_1) \leq k$. In case $k = D$, again $d(f_1) \leq k$. Recall $F_1(w)_{[0,k]} = F_1(w')_{[0,k]}$, so $w_0 = w'_0$. We can conclude $d(f_2 \circ f_1) \leq D = d(f_2) + d(f_1)$.

$d(f_1) \leq d(f_2 \circ f_1)$: Set $D = d(f_1)$, and let $w = w_0 \dots w_{D-1}$ and $w' = w'_0 \dots w'_{D-1}$ be $D$-blocks in $X_G$ having the same initial state, distinct first edges, and $F_1(w) = F_1(w')$. Then $F_2 \circ F_1(w) = F_2 \circ F_1(w')$ also, showing $D \leq d(f_2 \circ f_1)$.

$d(f_2) \leq d(f_2 \circ f_1)$: Set $D = d(f_2)$. We may assume that $d = d(f_1)$ is finite (for, as we have shown, $d(f_1) \leq d(f_2 \circ f_1)$). Let $v = v_0 \dots v_{D-1}$ and $v' = v'_0 \dots v'_{D-1}$ be $D$-blocks in $X_H$ with $s(v) = s(v')$, distinct first edges, and $F_2(v) = F_2(v')$. Let $v^*$ be any $d$-block in $X_H$ with $t(v^*) = s(v)$. Pick a $d$-block $u^*$ in $X_G$ with $F_1(u^*) = v^*$. Using that $X_G$ and $X_H$ are irreducible, and that $f_1$ is right closing and onto, [7, Lem. 5] gives two $(d+D)$-blocks $u = u_0 \dots u_{d+D-1}$ and $u' = u'_0 \dots u'_{d+D-1}$ in $X_G$ such that $s(u) = s(u') = s(u^*)$, $F_1(u) = v^*v$, and $F_1(u') = v^*v'$. Now

$$F_2 \circ F_1(u) = F_2(v^*)F_2(v) = F_2(v^*)F_2(v') = F_2 \circ F_1(u').$$

Since $v_0 \neq v_0'$, we have $u_d \neq u_d'$. If $0 \leq k \leq d$ is the least index with $u_k \neq u_k'$, then $s(u_k) = s(u_k')$. So

$$d(f_2 \circ f_1) \geq D + d - k \geq D = d(f_2). \qquad \square$$

**14. Appendix II.** We show that, for Example 10.1, out-splitting alone requires at least 3 rounds. The eigenvector $\mathbf{r}$ of the adjacency matrix of $G$, corresponding to eigenvalue 5, gives an explanation.. Each state $s$ of Fig. 28 is labeled by its corresponding eigenvector component $r_s$, called its *weight.* If $G$ could be out-split in 2 rounds alone to achieve the underlying graph $\bar{G}$ of a finite-state $(X_G, 5)$-code, then $\bar{G}$ would have uniform out-degree 5; it follows that there would be a weight $w_0$ such that for each state $s$ of $G$, the set of 2-blocks starting at state $s$ would be partitioned into atoms in such a way that within each atom the sum of weights of terminal states of its member 2-blocks is $w_0$. It is not hard to see that the only possibility here is $w_0 = 25$.

Suppose that there were such a partition and call it $\mathcal{P}$. For each state $s$, let $\mathcal{P}_s$ denote the partition restricted to 2-blocks outgoing from $s$.

The 2-blocks with initial state $t(b)$ have terminal states $t(j), t(k), t(\ell), \ldots, t(o)$ weighing in at $10, 10, 1, 14, 9, 6$, respectively. There is a unique partition of this multiset of integers into 2 atoms, each atom summing to 25: $\{10, 1, 14\}$ and $\{10, 9, 6\}$. Thus the pair of paths $\{f\ell, fm\}$ is contained in one atom of $\mathcal{P}_{t(b)}$, and the pair $\{fn, fo\}$ is contained in the other.

Similarly, the 2-blocks beginning at state $t(c)$ have terminal states $t(\ell)$, $t(m), \ldots t(q)$ weighing $1, 14, 9, 6, 5, 15$, respectively. Again there is a unique partition of these weights into 2 atoms, each summing to 25: $\{1, 9, 15\}$ and $\{14, 6, 5\}$. Thus the pair $\{g\ell, gn\}$ is contained in one atom of $\mathcal{P}_{t(c)}$, and $\{gm, go\}$ is contained in the other.

Let $\mathcal{P}_{\{\ell,m|n,o\}}$ be the partition of the 2-blocks beginning at state $t(f) = t(g)$ given by:

$$\{\{\ell\ell_i : i = 1\ldots 5\} \cup \{mm_i : i = 1\ldots 70\}, \qquad \{nn_i : i = 1 \ldots 45\} \cup \{oo_i : i = 1 \ldots 30\}\}.$$

Using that the sets $\{f\ell, fm\}$ and $\{fn, fo\}$ are contained in distinct atoms of $\mathcal{P}_{t(b)}$, the left Markov property (6) tells us that the partition $\mathcal{P}_{t(f)} = \mathcal{P}_{t(g)}$ refines $\mathcal{P}_{\{\ell,m|n,o\}}$.

Similarly, let $\mathcal{P}_{\{\ell,n|m,o\}}$ be the partition of the 2-blocks beginning at state $t(f) = t(g)$ given by:

$$\{\{\ell\ell_i : i = 1\ldots 5\} \cup \{nn_i : i = 1\ldots 45\}, \qquad \{mm_i : i = 1 \ldots 70\} \cup \{oo_i : i = 1 \ldots 30\}\}.$$

Using that $\{g\ell, gn\}$ and $\{gm, go\}$ are distinct of $\mathcal{P}_{t(c)}$, we have that $\mathcal{P}_{t(f)} = \mathcal{P}_{t(g)}$ refines $\mathcal{P}_{\{\ell,n|m,o\}}$. Thus $\mathcal{P}_{t(f)} = \mathcal{P}_{t(g)}$ refines $\mathcal{P}_{\{\ell,m|n,o\}} \vee \mathcal{P}_{\{\ell,n|m,o\}}$, which has 4 atoms:

$$\{\{\ell\ell_i : i = 1\ldots 5\}, \quad \{mm_i : i = 1\ldots 70\}, \quad \{nn_i : i = 1 \ldots 45\}, \quad \{oo_i : i = 1 \ldots 30\}\}.$$

But none of these 4 atoms comprises a set of 2-blocks whose terminal states' weights sum to a multiple of 25, as each must if the purported partition $\mathcal{P}$ exists. This contradiction shows that at least 3 rounds of out-splitting are required. $\square$

**Acknowledgments.** We are extremely grateful to Masakasu Nasu for suggesting the use of an idea inherent in his presentation of textile relational systems ([31]) as a way to simplify the proof of our main result, Theorem 6.2. This suggestion also motivated our consideration of the intrinsic characterization of sliding block decoders (Theorem 7.1). We are also grateful to the Mathematical Sciences Research Institute for their hospitality during the Program on Symbolic Dynamics, Fall 1992, where some of this work was done. The second author would like to thank Joel Friedman for useful discussions on these matters several years ago and Kees Schouhamer Immink for thought-provoking remarks. Finally, we thank both anonymous referees for their very thorough reading and many useful corrections.

## REFERENCES

[1] R. Adler, D. Coppersmith, and M. Hassner, *Algorithms for sliding-block codes*, IEEE Trans. Inform. Theory, IT-29 (1983), pp. 5–22.

[2] R. Adler, L. Goodwyn, and B. Weiss, *Equivalence of topological Markov shifts*, Israel J. Math., 27 (1977), pp. 49–63.

[3] R. Adler and B. Marcus, *Topological entropy and equivalence of dynamical systems*, Mem. Amer. Math. Soc., 219 (1979).

[4] J. Ashley, *Marker automorphisms of the one-sided d-shift*, Ergodic Theory Dynamical Systems, 10 (1990), pp. 247–262.

[5] ———, *LR conjugacies of shifts of finite type are uniquely so*, Contemp. Math., 135 (1992), pp. 57–84.

[6] ———, *A linear bound for sliding block decoder window size*, IEEE Trans. Inform. Theory, 24 (1988), pp. 389–399.

[7] J. Ashley, B. Marcus, D. Perrin, and S. Tuncel, *Surjective extensions of sliding block codes*, SIAM J. Discrete Math., 6 (1993), pp. 582–611.

[8] J. Ashley, B. Marcus, and R. Roth, *Construction of encoders with small decoding look-ahead for input-constrained channels*, IEEE Trans. Inform. Theory, 41 (1995), pp. 55–76.

[9] M.-P. Beal, *The method of poles: A coding method for constrained channels*, IEEE Trans. Inform. Theory, 36 (1990) pp. 763–772.

[10] M. Boyle, B. Kitchens, and B. Marcus, *A note on minimal covers for sofic systems*, Proc. Amer. Math. Soc., 95 (1985), pp. 403–411.

[11] M. Boyle, B. Marcus, and P. Trow, *Resolving maps and the dimension group for shifts of finite type*, Mem. Amer. Math. Soc., 377 (1987).

[12] E. Coven and M. Paul, *Endomorphisms of irreducible shifts of finite type*, Math. Systems Theory, 8 (1974), pp. 167–175.

[13] R. Fischer, *Graphs and symbolic dynamics*, Colloq. Math. Soc. Janos Bolyai, 16 (1975), pp. 229–243.

[14] P. Franaszek, *Construction of bounded delay codes for discrete noiseless channels*, IBM J. Res. Develop., 26 (1982), pp. 506–514.

[15] P. Franaszek and J. Thomas, *On the optimization of constrained channel codes*, summary in Proc. IEEE International Symposium on Information Theory, San Antonio, Texas, 1993.

[16] H. Hollmann, *A block-decodable* $(d,k) = (1,8)$ *runlength-limited rate* $8/12$ *code*, IEEE Trans. Inform. Theory, 40 (1994), pp. 1292–1296.

[17] J. Hopcroft and J. Ullmann, *Introduction to Automata Theory, Languages, and Computation*, Addison–Welsey, Reading, MA, 1979.

[18] K. A. Schouhamer Immink, *Block-decodable runlength-limited codes via look-ahead techniques*, Philips J. Res., 46 (1992), pp. 293–310.

[19] H. Kamabe, *Minimum scope for sliding-block decoder mappings*, IEEE Trans. Inform. Theory, IT-35 (1989), pp. 1335–1340.

[20] R. Karabed and B. Marcus, *Sliding-block coding for input-restricted channels*, IEEE Trans. Inform. Theory, IT-34 (1988), pp. 2–26.

[21] A. Khayrallah and D. Neuhoff, *On sliding block decoders for input-restricted channels*, Proceedings of the International Symposium on Information Theory and its Applications, 1990, pp. 1-2.5–1-2.7.

[22] B. Kitchens, B. Marcus, and P. Trow, *Eventual factor maps and compositions of closing maps*, Ergodic Theory Dynamical Systems, 11 (1991), pp. 85–113.

[23] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw–Hill, New York, 1970.

[24] W. Krieger, *On sofic systems* I, Israel J. Math., 48 (1984), pp. 305–330.

[25] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding*, Cambridge University Press, Cambridge, 1995.

[26] B. Marcus, *Sofic systems and encoding data*, IEEE Trans. Inform. Theory, IT-31 (1985),

pp. 366–377.

[27] B. MARCUS AND R. ROTH, *Bounds on the number of states in encoder graphs for input-constrained channels*, IEEE Trans. Inform. Theory, IT-37 (1991), pp. 742–758.

[28] B. MARCUS, P. SIEGEL, AND J. WOLF, *A tutorial on finite-state modulation codes*, IEEE Journal on Selected Areas in Communication, 10 (1992), pp. 5–37.

[29] R. MCELIECE, *The theory of information and coding*, Encyclopedia of Mathematics and Its Applications, vol. 3, Addison–Wesley, Reading, MA, 1977.

[30] M. NASU, *Constant-to-one and onto global maps of homomorphisms between strongly connected graphs*, Ergodic Theory Dynamical Systems, 3 (1983), pp. 387–413.

[31] ———, *Textile systems for endomorphisms and automorphisms*, Mem. Amer. Math. Soc., 546 (1995).

[32] W. PARRY AND S. TUNCEL, *Classification problems in ergodic theory*, LMS Lecture Notes, vol. 67, Cambridge University Press, Cambridge, 1982.

[33] P. SIEGEL, *Recording codes for digital magnetic storage*, IEEE Transactions on Magnetics, MAG-21 (1985), pp. 1344–1349.

[34] J. WAGONER, *Triangle identities and symmetries of a subshift of finite type*, Pacific J. Math., 144 (1990), pp. 181–205.

[35] R. WILLIAMS, *Classification of shifts of finite type*, Ann. of Math., 98 (1973), pp. 120–153; errata, Ann. of Math., 99 (1974), pp. 380–381.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 606–616, November 1995

# TREEWIDTH AND PATHWIDTH OF PERMUTATION GRAPHS*

HANS L. BODLAENDER†, TON KLOKS‡, AND DIETER KRATSCH§

**Abstract.** In this paper, we show that the treewidth and pathwidth of a permutation graph can be computed in polynomial time. In fact we show that, for permutation graphs, the treewidth and pathwidth are equal. These results make permutation graphs one of the few nontrivial graph classes for which, at the moment, treewidth is known to be computable in polynomial time. Our algorithm, which decides whether the treewidth (pathwidth) is at most some given integer $k$, can be implemented to run in $O(nk)$ time when the matching diagram is given. We show that this algorithm can easily be adapted to compute the pathwidth of a permutation graph in $O(nk)$ time, where $k$ is the pathwidth.

**Key words.** permutation graphs, graph algorithms, treewidth

**AMS subject classifications.** 05C85, 68Q20, 68R10, 06A07

## 1. Introduction.

In many recent investigations in computer science, the notions of treewidth and pathwidth play an increasingly important role. One reason for this is that many problems, including many well-studied NP-complete graph problems, become solvable in polynomial and usually even linear time when restricted to the class of graphs with bounded treewidth or pathwidth [1], [3], [4], [7], [9], and [25]. Of crucial importance for these algorithms is that a tree decomposition or path decomposition of the graph is given in advance. Much research has been done in finding a tree decomposition with a reasonable small treewidth. Recent results show that a linear-time algorithm exists to find an optimal tree decomposition for a graph with bounded treewidth [8] (see also [22]). However, the constant hidden in the "big oh" is exponential in the treewidth, limiting the practicality of this algorithm.

For some *special classes* of graphs, it has been shown that the treewidth can be computed efficiently. In this paper we discuss the problem of finding tree and path decompositions for permutation graphs. We also show that for these graphs, the treewidth and the pathwidth are the same.

Permutation graphs are a nontrivial class of perfect graphs. They have many applications in scheduling problems. See, for example, [14], where permutation graphs are used to describe the memory requirements of a number of programs at a certain time (see also [17]). Permutation graphs also arise in a very natural way in the problem of sorting a permutation using queues in parallel. In [17] it is shown that this problem is closely related to the coloring problem of permutation graphs. Other applications occur for example in very-large-scale integration (VLSI) layout (see, e.g., [27]). There exist fast algorithms for many NP-complete problems like CLIQUE, INDEPENDENT SET, FEEDBACK VERTEX SET, and DOMINATING SET when restricted to permutation graphs

* Received by the editors November 12, 1992; accepted for publication (in revised form) September 23, 1994. A preliminary version of this paper appeared in the Proceedings of the 20th International Colloquium on Automata, Languages, and Programming, Lund (Sweden), July 1993.

† Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands.

‡ Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands (ton@win.tue.nl). The research of this author was supported by S.I.O.N., the foundation of the Netherlands Organization for Scientific Research (N.W.O.).

§ Fakultät für Mathematik und Informatik, Friedrich-Schiller-Universität, Universitätshochhaus, 07740 Jena, Germany.

[12], [13], [15], [17]. However, some problems remain NP-complete, like COCHROMATIC NUMBER [16], [30], and ACHROMATIC NUMBER [6].

We give an $O(nk)$-time algorithm that determines whether the pathwidth (or treewidth) of a permutation graph is at most $k$ when the matching diagram of the graph is given. The algorithm can easily be adapted such that it computes the pathwidth (or treewidth) of a permutation graph in $O(nk)$ time, where $k$ denotes the pathwidth of the input graph.

## 2. Preliminaries.

In this section we start with some definitions and easy lemmas. For more information on perfect graphs the reader is referred to [5], [11], [17].

DEFINITION 2.1. *A graph is* chordal *if it has no induced chordless cycle of length at least four.*

Chordal graphs are also called triangulated. There are basically two ways to define the treewidth of a graph. One way is to use the concept of a *tree decomposition*. For more information on tree decompositions, the reader is referred to [9]. In this paper, we introduce the treewidth of a graph by means of $k$-trees.

DEFINITION 2.2. *Let $k$ be an integer. A $k$-tree is a graph that is defined recursively as follows. A clique with $k+1$ vertices is a $k$-tree. Given a $k$-tree $T_n$ with $n$ vertices, a $k$-tree with $n+1$ vertices can be constructed by making a new vertex adjacent to the vertices of a $k$-clique in $T_n$. A graph is a* partial $k$-tree *if either it has at most $k$ vertices or it is a subgraph of a $k$-tree $T$ with the same vertex set as $T$.*

Every $k$-tree $G$ is chordal and $\omega(G) = k+1$ (where $\omega(G)$ is the maximum clique size). Notice that any graph is a partial $k$-tree for some $k$; take $k$ equal to the number of vertices minus one.

DEFINITION 2.3. *The* treewidth *of a graph $G$ is the minimum value $k$ for which $G$ is a partial $k$-tree.*

DEFINITION 2.4. *A* triangulation *of a graph $G$ is a graph $H$ with the same vertex set as $G$, such that $G$ is a subgraph of $H$ and $H$ is chordal.*

For a proof of the following lemma see for example [25].

LEMMA 2.5. *A graph $G$ has treewidth $\leq k$ if and only if there is a triangulation $H$ of $G$ with $\omega(H) \leq k+1$.*

It follows that the treewidth of a chordal graph is the maximum clique size minus one. The treewidth can be defined in terms of the minimum over all triangulations of the maximum clique size. We define the *pathwidth* of a graph using triangulations of a special kind.

DEFINITION 2.6. *An* interval graph *is a graph of which the vertices can be put into one-to-one correspondence with intervals on the real line such that two vertices are adjacent if and only if the corresponding intervals have a nonempty intersection.*

There are many ways to characterize interval graphs. We state only one of the first characterizations [26].

LEMMA 2.7. *An undirected graph is an interval graph if and only if the following two conditions are satisfied:*

(i) *$G$ is chordal*; *and*

(ii) *any three vertices of $G$ can be ordered in such a way that every path from the first to the third vertex passes through a neighbor of the second vertex.*

Three vertices that do not satisfy the second condition are called an *astroidal triple*. These are pairwise nonadjacent, and for any pair of them there is a path that avoids the neighborhood of the remaining vertex.

DEFINITION 2.8. *Let $k$ be an integer. A graph $G$ has* pathwidth $\leq k$ *if and only if there is a triangulation $H$ of $G$ such that $H$ is an interval graph with $\omega(H) \leq k+1$.*

Determining the treewidth or the pathwidth of a graph is NP-complete [2]. However, for *constant* $k$, graphs with treewidth $\leq k$ are recognizable in $O(n)$ time [8]. The large constants involved in these algorithms usually make them not very practical. It is therefore of importance to find fully polynomial algorithms for treewidth and pathwidth for special classes of graphs which are as large as possible.

One of the main reasons why there exist fast algorithms for many problems when restricted to graphs with bounded treewidth is the existence of vertex separators of bounded size.

DEFINITION 2.9. *A subset $S \subseteq V$ is an $a,b$-separator for nonadjacent vertices $a$ and $b$ if the removal of $S$ separates $a$ and $b$ in distinct connected components. If no proper subset of $S$ is an $a,b$-separator, then $S$ is a minimal $a,b$-separator. A* minimal separator *$S$ is a subset such that $S$ is a minimal $a,b$-separator for some nonadjacent vertices $a$ and $b$.*

The following lemma, which must have been rediscovered many times, appears for example as an exercise in [17].

LEMMA 2.10. *Let $S$ be a minimal $a,b$-separator and $C_a$ and $C_b$ be the connected components of $G[V-S]$ containing $a$ and $b$, respectively. Then every vertex of $S$ has a neighbor in $C_a$ and a neighbor in $C_b$.*

THEOREM 2.11. *Let $G$ be a partial $k$-tree. There exists a triangulation of $G$ into a chordal graph $H$ such that the following three statements hold:*

(i) $\omega(H) \leq k+1$.

(ii) *If $a$ and $b$ are nonadjacent vertices in $H$, then every minimal $a,b$-separator of $H$ is also a minimal $a,b$-separator in $G$.*

(iii) *If $S$ is a minimal separator in $H$ and $C$ is the vertex set of a connected component of $H[V-S]$, then $C$ induces also a connected component in $G[V-S]$.*

*Proof.* Take a triangulation $H$, with treewidth at most $k$ and with a minimum number of edges (this exists by Lemma 2.5). Suppose $H$ has a minimal vertex separator $C$ for nonadjacent vertices $a$ and $b$ such that either $C$ induces no minimal $a,b$-separator in $G$ or the vertex sets of the connected components of $H[V-C]$ are different from those of $G[V-C]$. Let $S \subseteq C$ be a minimal $a,b$-separator in $G$. Let $C_1,\ldots,C_t$ be the connected components of $G[V-S]$. Make a chordal graph $H'$ as follows. For each $C_i \cup S$, take the chordal subgraph of $H$ induced by these vertices. Since $S$ is a clique in $H$, this gives a chordal subgraph $H'$ of $H$. Notice that the vertex sets of the connected components of $H'[V-S]$ are the same as those of $G[V-S]$. We claim that the number of edges of $H'$ is less than the number of edges of $H$, which is a contradiction. Clearly, the number of edges of $H'$ does not exceed the number of edges of $H$. First assume that $S \neq C$ and let $x \in C \backslash S$. By Lemma 2.10, in $H$, $x$ has a neighbor in the component containing $a$ and a neighbor in the component containing $b$. Not both of these edges can be present in $H'$. Thus, we may assume $S = C$. Then the vertex sets of the connected components of $H[V-C]$ are different from those of $H'[V-C]$. Since $H'$ is a subgraph of $H$, every connected component $H'[V-C]$ is contained in some connected component of $H[V-C]$. It follows that there must be a connected component in $H[V-C]$ containing two different connected components of $H'[V-C]$. This can only be the case if there is some edge between these components in $H[V-C]$ (which is not there in $H'[V-C]$). This proves the theorem. □

DEFINITION 2.12. *We call a triangulation whose existence is guaranteed by Theorem* 2.11 *a* minimal triangulation.

Let $G=(V,E)$ be a graph and let $C$ be a minimal vertex separator. Let $C_1,\ldots,C_t$ be the connected components of $G[V-C]$. We denote by $\overline{C_i}$ $(i=1,\ldots,t)$ the graph obtained as follows. Take the induced subgraph $G[C \cup C_i]$ and add edges such that the subgraph induced by $C$ is complete. The following lemma easily follows from Theorem 2.11 (a similar result appears in [2]).

LEMMA 2.13. *A graph $G$ with at least $k+2$ vertices is a partial $k$-tree if and only if there exists a minimal vertex separator $C$ such that the graphs $\overline{C}_i$ are partial $k$-trees $(i = 1, \ldots, t)$.*

In this paper, we show that the treewidth and pathwidth of a permutation graph can be computed in polynomial time. We think of a permutation $\pi$ of the numbers $1, \ldots, n$ as the sequence $\pi = [\pi_1, \ldots, \pi_n]$.

DEFINITION 2.14. *If $\pi$ is a permutation of the numbers $1, \ldots, n$, we can construct a graph $G[\pi] = (V, E)$ with vertex set $V = \{1, \ldots, n\}$ and edge set $E$:*

$$(i,j) \in E \Leftrightarrow (i-j)(\pi_i^{-1} - \pi_j^{-1}) < 0.$$

*An undirected graph is a* permutation graph *if there is a permutation $\pi$ such that $G \cong G[\pi]$.*

The graph $G[\pi]$ is sometimes called the inversion graph of $\pi$. If the permutation is not given, it can be computed in $O(n^2)$ time [17], [28]. In this paper, we assume that the permutation is given and we identify the permutation graph with the inversion graph. A permutation graph is an intersection graph, which is illustrated by the matching diagram.

DEFINITION 2.15. *Let $\pi$ be a permutation of $1, \ldots, n$. The matching diagram can be obtained as follows. Write the numbers $1, \ldots, n$ horizontally from left to right. Underneath, write the numbers $\pi_1, \ldots, \pi_n$, also horizontally from left to right. Draw straight line segments joining the two 1's, the two 2's, etc.*

Notice that two vertices $i$ and $j$ of $G[\pi]$ are adjacent if and only if the corresponding line segments intersect. In Fig. 1 we give an example.

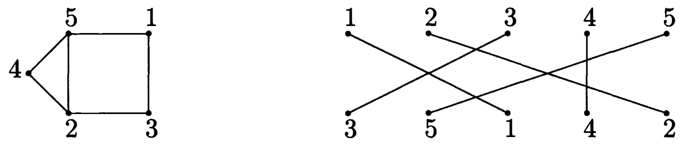


FIG. 1. *Permutation graph and matching diagram.*

**3. Scanlines.** In this section, we show that every minimal separator in a permutation graph can be obtained by using a *scanline.* Recall the definition of the matching diagram. It consists of two horizontal lines, one above the other, and a number of straight line segments, one for each vertex, such that each line segment has one end vertex on each horizontal line. Two vertices are adjacent if the corresponding line segments intersect. We say that two line segments *cross* if they have a nonempty intersection.

DEFINITION 3.1. *A* scanline *in the diagram is any line segment with one end vertex on each horizontal line. A scanline $s$ is* between *two noncrossing line segments $x$ and $y$ if the top point of $s$ is in the open interval bordered by the top points of $x$ and $y$ and the bottom point of $s$ is in the open interval bordered by the bottom points of $x$ and $y$.*

If a scanline $s$ is between line segments $x$ and $y$, then the intersection of each pair of the three line segments is empty. Consider two nonadjacent vertices $x$ and $y$. The line segments in the diagram corresponding to $x$ and $y$ do not cross in the diagram.

Hence we can find a scanline $s$ between the lines $x$ and $y$. Take out all the lines that cross the scanline $s$. Clearly, this corresponds with an $x,y$-separator in the graph. The next lemma shows that we can find all minimal $x,y$-separators in this way.

LEMMA 3.2. *Let $G$ be a permutation graph and $x$ and $y$ be nonadjacent vertices in $G$. Every minimal $x,y$-separator consists of all line segments crossing a scanline that lies between the line segments of $x$ and $y$.*

*Proof.* Let $S$ be a minimal $x,y$-separator. Consider the connected components of $G[V-S]$. Let $C_x$ be the component containing $x$ and $C_y$ be the component containing $y$. Clearly, these must also be "connected" parts in the diagram, and we may assume without loss of generality that the component containing $x$ is completely to the left of the component containing $y$. Every vertex of $S$ is adjacent to some vertex in $C_x$ and to some vertex in $C_y$ (Lemma 2.10). Notice that we can choose a scanline $s$ crossing no line segment of $G[V - S]$ and that is between $x$ and $y$. Then all lines crossing the scanline must be elements of $S$. But for all elements of $S$ the corresponding line segment must cross $s$, since it is intersecting with a line segment of $C_x$, which is to the left of $s$, and with a line segment of $C_y$, which is to the right of $s$. □

COROLLARY 3.3. *There are $O(n^2)$ minimal separators in a permutation graph with $n$ vertices.*

If $s$ is a scanline, then we denote by $S$ the set of vertices of which the corresponding line segments cross $s$. In the rest of this paper, we consider only scanlines of which the endpoints do not coincide with endpoints of other line segments.

DEFINITION 3.4. *Two scanlines $s_1$ and $s_2$ are* equivalent, *$s_1 \equiv s_2$, if they have the same position in the diagram relative to every line segment.*

Hence, if $s_1 \equiv s_2$, then the set of line segments with the top (or bottom) endpoint to the left of the top (or bottom) endpoint of the scanline is the same for $s_1$ and $s_2$.

We are only interested in scanlines that do not cross too many line segments, since these correspond with suitable separators.

DEFINITION 3.5. *A scanline $s$ is* $k$-small *if it crosses with at most $k + 1$ line segments.*

LEMMA 3.6. *There are $O(nk)$ pairwise nonequivalent $k$-small scanlines.*

*Proof.* Consider the matching diagram, with numbers $1, \ldots, n$ written from left to right and written $\pi_1, \ldots, \pi_n$ underneath. Consider a scanline $t$ and assume that the top endpoint is between $i$ and $i + 1$ and the bottom endpoint is between $\pi_j$ and $\pi_{j+1}$. Assume that $s$ line segments are such that the top endpoint is to the left of the top of $t$ and the bottom endpoint to the left of the bottom of $t$. Then the number of line segments crossing $t$ is $i + j - 2s$. Since $s \leq i$ and $s \leq j$, it follows that $i - k - 1 \leq j \leq i + k + 1$ must hold. This proves the lemma. □

## 4. Treewidth = pathwidth.

In this section, we show that a permutation graph can be triangulated optimally such that the result is an interval graph.

THEOREM 4.1. *Let $G$ be a permutation graph and $H$ be a minimal triangulation of $G$. Then $H$ is an interval graph.*

*Proof.* Assume $H$ has an astroidal triple $x, y, z$. Since $x, y$, and $z$ are pairwise nonadjacent, the corresponding line segments in the matching diagram pairwise do not cross. We may assume without loss of generality that the line segment of $y$ is between those of $x$ and $z$. Take a path $p$ between $x$ and $z$ that avoids the neighborhood of $y$. Then each line of the path lies totally to the left or totally to the right of $y$. It follows that there are $x'$ to the left of $y$ and $z'$ to the right of $y$ such that $x'$ and $z'$ are adjacent in $H$ but neither $x'$ nor $z'$ is a neighbor of $y$ in $H$. Let $S$ be a minimal $x', y$-separator in $H$. Since $H$ is a minimal triangulation, $S$ is also a minimal $x', y$-separator in $G$. By

Lemma 3.2, $S$ consists of all lines crossing some scanline $s$ between $x'$ and $y$. Clearly, the connected component of $G[V-S]$ containing $x'$ lies totally to the left of $s$ and the connected component containing $z'$ in $G[V-S]$ lies totally to the right of $s$ (notice that $z' \notin S$ since $z'$ lies totally to the right of $y$). It follows that $x'$ and $z'$ must be in different components of $G[V-S]$. Since $H$ is minimal, by Theorem 2.11 they must also be in different components of $H[V-S]$. But then $x'$ and $z'$ cannot be adjacent in $H$. It follows that there cannot be an astroidal triple, and by the characterization of Lekkerkerker and Boland ([26], stated in Lemma 2.7), $H$ is an interval graph. □

COROLLARY 4.2. *For a permutation graph $G$, the pathwidth of $G$ is equal to the treewidth of $G$.*

## 5. Candidate components.

Consider the matching diagram of $G$.

DEFINITION 5.1. *Let $s_1$ and $s_2$ be two scanlines of which the intersection is either empty or one of the endpoints of $s_1$ and $s_2$. A* candidate component $C = \mathcal{C}(s_1, s_2)$ *is a subgraph of $G$ induced by the following sets of lines:*

1. *all lines that are* between *the scanlines (in case the scanlines have a common endpoint, this set is empty);*
2. *all lines crossing at least one of the scanlines.*

We identify the candidate component $C = \mathcal{C}(s_1, s_2)$ with the diagram containing $s_1$, $s_2$ and the set of lines corresponding with vertices of $C$.

DEFINITION 5.2. *Let $k$ be an integer. A candidate component $C = \mathcal{C}(s_1, s_2)$ is* $k$-feasible *if there is a triangulation $H$ of $C$ such that $\omega(H) \leq k+1$ and such that for each scanline $s_i$ $(i = 1, 2)$ the set of lines crossing this scanline forms a clique in $H$.*

Notice that if a candidate component has at most $k+1$ vertices, then it is $k$-feasible.

DEFINITION 5.3. *Let $C = \mathcal{C}(s_1, s_2)$ be a candidate component. We define the* realizer $R(C)$ *as the graph obtained from $C$ by adding all edges between vertices of $S_1$ and between vertices of $S_2$ (i.e., the two subgraphs of $R(C)$ induced by $S_1$ and by $S_2$ are cliques).*

A candidate component $C = \mathcal{C}(s_1, s_2)$ is $k$-feasible if and only if the realizer $R(C)$ has treewidth at most $k$.

LEMMA 5.4. *If $C = \mathcal{C}(s_1, s_2)$ is a candidate component, then the realizer $R(C)$ is a permutation graph.*

*Proof.* Consider the matching diagram. Assume $s_1$ is to the left of $s_2$. First consider lines that cross only $s_1$ and with top endpoints to the right of the top endpoint of $s_1$. Let $(a_1, b_1), \ldots, (a_r, b_r)$ be these line segments, with top endpoints $a_1, \ldots, a_r$. Assume $a_1 < a_2 < \cdots < a_r$. Change the order of $b_1, \ldots, b_r$ such that $b_1 > b_2 > \cdots > b_r$. This is illustrated in Fig. 2. Now consider the line segments crossing $s_1$ of which the top endpoint is to the left of the top endpoint of $s_1$. Reorder in the same way the *top endpoints* of these line segments. The lines crossing $s_2$ are handled similarly. The resulting diagram is a matching diagram for $R(C)$. □

Let $C = \mathcal{C}(s_1, s_2)$ be a candidate component such that $C$ has at least $k+2$ vertices. The realizer $R(C)$ has treewidth at most $k$ if and only if there is a minimal vertex separator $S$ with at most $k$ vertices such that all components, with $S$ added as a clique, have treewidth at most $k$ (see Lemma 2.13). Consider the diagram of $R(C)$, obtained from the diagram of $C$ by the method described in the proof of Lemma 5.4. By Lemma 3.2, a minimal separator can be found by a scanline. Let $H$ be a minimal triangulation of $R(C)$ and let the scanline $s$ represent a minimal vertex separator in $H$ for nonadjacent vertices $a$ and $b$. The separator consists exactly of the lines crossing this scanline $s$.

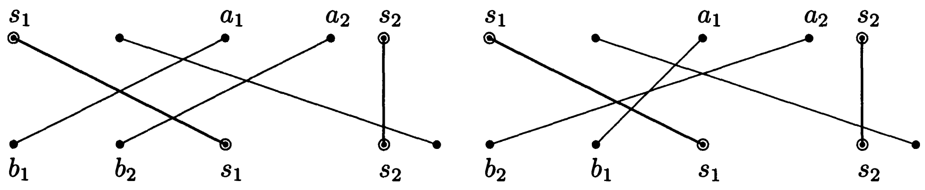


FIG. 2. *Diagrams of candidate component and realizer.*

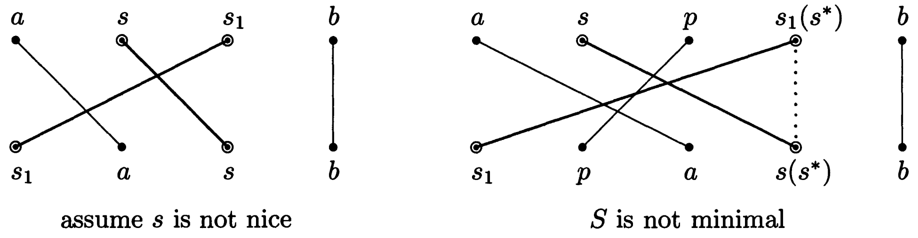


FIG. 3. *There is an equivalent nice scanline.*

DEFINITION 5.5. *Let $C = C(s_1, s_2)$ be a candidate component with realizer $R(C)$. A scanline $t$ is* nice *if the top point of $t$ is in the closed interval between the top points of $s_1$ and $s_2$ and the bottom point of $t$ is in the closed interval between the bottom points of $s_1$ and $s_2$.*

LEMMA 5.6. *There is a scanline $s^* \equiv s$ such that $s^*$ is nice.*

*Proof.* Consider the diagram of $R(C)$ with the scanlines $s_1$, $s_2$, and $s$. Without loss of generality, we assume that $s_1$ is to the left of $s_2$. The scanline $s$ separates nonadjacent vertices $a$ and $b$. Let the line segment of $a$ be to the left of the line segment of $b$. The scanline $s$ lies between the line segments of $a$ and $b$. Assume that $s$ is not nice. Without loss of generality, assume that it crosses $s_1$. Then $a \in S_1$ and $b \notin S_1$, since $a$ and $b$ are not adjacent (see the left diagram in Fig. 3). Let $s^*$ be the line segment with its top point at the top of $s_1$ and its bottom point at the bottom of $s$. We want to prove that $s \equiv s^*$. This is clearly the case if there is no line segment for which the top point is between the top points of $s$ and $s_1$. Assume that there is such a vertex $p$ (see the right diagram in Fig. 3). Notice that, since $p$ and $a$ both cross $s_1$, they are adjacent. We claim that $S^* \subset S$. Let $x$ be a line segment crossing $s^*$. If the bottom end of $x$ is to the left of the bottom end of $s^*$, then the segment $x$ clearly also crosses $s$. Assume that the bottom vertex of $x$ is to the right of the bottom vertex of $s^*$. Then the line segment also crosses $s_1$. But then $x$ and $a$ are adjacent, hence the top vertex of $x$ must be to the left of the top vertex of $a$. This implies that $x$ also crosses $s$. Clearly, $S^*$ is an $a, b$-separator in $R(C)$. Since $p \in S \backslash S^*$, $S$ cannot be a minimal $a, b$-separator in $R(C)$ and since $H$ is a minimal triangulation of $R(C)$, it cannot be a minimal $a, b$-separator in $H$ (Theorem 2.11). This, then, is a contradiction. □

This proves that all lines crossing the scanline $s$ are in $C$, and the next lemma follows.

LEMMA 5.7. *Let $C = C(s_1, s_2)$ be a candidate component with at least $k + 2$ vertices. Then $C$ is $k$-feasible if and only if there is a nice scanline $s$ such that the two candidate components $C_1 = C(s_1, s)$ and $C_2 = C(s, s_2)$ are both smaller than $C$ and are both $k$-feasible.*

DEFINITION 5.8. *Two scanlines $s_1$ and $s_2$ are* neighbors *if they have one endpoint in common and in the interval determined by the other endpoints, lies exactly one endpoint of a line segment.*

We are now ready to prove our main theorem.

THEOREM 5.9. *Let $C = \mathcal{C}(s_1, s_2)$ be a candidate component with $s_2$ to the right of $s_1$. Then $C$ is $k$-feasible if and only if there exists a sequence of scanlines $s_1 = t_1, t_2, \ldots, t_r = s_2$ such that the following conditions are satisfied:*

1. *The scanlines $t_i$ and $t_{i+1}$ are neighbors for $i = 1, \ldots, r-1$ and one endpoint of $t_{i+1}$ lies to the right of the endpoint of $t_i$.*
2. *Each $\mathcal{C}(t_i, t_{i+1})$ has at most $k+1$ vertices ($i = 1, \ldots, r-1$).*

*Proof.* First, assume that such a sequence exists. Let $t_1, \ldots, t_r$ be the sequence of scanlines. If $C$ has at most $k+1$ vertices, then $C$ is $k$-feasible. Hence, we may assume that $C$ has at least $k+2$ vertices. Then $r \geq 3$. By induction, we show that $\mathcal{C}(t_1, t_i)$ is $k$-feasible for $i = 2, \ldots, r$. If $i = 2$, then $\mathcal{C}(t_1, t_2)$ has at most $k+1$ vertices and hence is $k$-feasible. Assume that $\mathcal{C}(t_1, t_{i-1})$ is $k$-feasible and $\mathcal{C}(t_1, t_i) \neq \mathcal{C}(t_1, t_{i-1})$. Then $t_{i-1}$ is a nice scanline in $\mathcal{C}(t_1, t_i)$. $\mathcal{C}(t_1, t_{i-1})$ and $\mathcal{C}(t_{i-1}, t_i)$ are both $k$-feasible; hence, by Lemma 5.7, $\mathcal{C}(t_1, t_i)$ is also $k$-feasible.

Now assume that $C$ is $k$-feasible. Consider the case that $C$ has at most $k+1$ vertices. In this case, it is easy to see that such a sequence of scanlines exists. If $s_1$ and $s_2$ have an endpoint in common, we can take $r = 2$. If $s_1$ and $s_2$ do not have an endpoint in common, take a scanline $t$ that has one endpoint in common with $s_1$ and the other endpoint in common with $s_2$. The sequence of scanlines $s_1, t, s_2$ satisfies these requirements. Assume that $C$ has at least $k+2$ vertices. Then by Lemma 5.7, there is a nice scanline $s^*$ such that $\mathcal{C}(s_1, s^*)$ and $\mathcal{C}(s^*, s_2)$ are both $k$-feasible. The theorem follows by induction on the number of vertices of $C$. □

## 6. Algorithm.

In this section, we show how to compute the treewidth (and pathwidth) of a permutation graph $G$. Let $k$ be an integer. The algorithm we present checks whether the treewidth of the permutation graph does not exceed $k$.

Make a directed acyclic graph $W_k(G)$ as follows. The vertices of the graph are the pairwise nonequivalent $k$-small scanlines. Direct an arc from scanline $s$ to $t$ if the following hold:

1. The scanlines $s$ and $t$ are neighbors such that one endpoint of $t$ is to the right of the corresponding endpoint of $s$.
2. The candidate component $\mathcal{C}(s, t)$ has at most $k+1$ vertices.

We call this graph the *scanline graph.*

LEMMA 6.1. *Let $s_L$ be the scanline that lies totally to the left of all line segments and $s_R$ be the scanline that lies totally to the right of all line segments. $G$ has treewidth at most $k$ if and only if there is a directed path in the scanline graph from $s_L$ to $s_R$.*

*Proof.* Clearly, $s_L$ and $s_R$ are $k$-small. The result follows immediately from Theorem 5.9. □

LEMMA 6.2. *The scanline graph has $O(nk)$ vertices, and each vertex is incident with at most 4 arcs.*

*Proof.* The bound on the number of vertices is proved in Lemma 3.6. For each scanline $s$ there are at most 4 scanlines $t$ such that $s$ and $t$ are neighbors. □

We now describe the algorithm which determines if the treewidth of $G$ is at most $k$.

**Step 1.** Make a maximal list $\mathcal{L}$ of pairwise nonequivalent $k$-small scanlines.
**Step 2.** Construct the acyclic digraph $W_k(G)$.

**Step 3.** If there exists a path in $W_k(G)$ from $s_L$ to $s_R$, then the treewidth of $G$ is at most $k$. If such a path does not exist, then the treewidth of $G$ is larger than $k$.

We now discuss the running time of the algorithm in more detail.

LEMMA 6.3. *The algorithm can be implemented to run in $O(nk)$ time.*

*Proof.* By Lemma 3.6, each $k$-small scanline can be characterized by two indices $i$ and $\theta$, with $0 \le i \le n$ and $-(k+1) \le \theta \le k+1$. The scanline with these indices has its top endpoint between $i$ and $i+1$ and its bottom endpoint between $\pi_{i+\theta}$ and $\pi_{i+\theta+1}$ (with obvious boundary restrictions). Let $A(i,\theta)$ be the number of line segments of which the top endpoint is to the left of the top endpoint of this scanline and that cross the scanline. Notice that $A(0,\theta) = 0$ for $\theta = 0, \ldots, k+1$. The rest of the table follows from:

$$A(i,\theta) = \begin{cases} A(i,\theta-1) & \text{if } \pi_{i+\theta} > i \\ A(i,\theta-1) - 1 & \text{if } \pi_{i+\theta} \le i \end{cases}$$
$$\text{if } \theta > -\min(k+1,i)$$

$$A(i,\theta) = \begin{cases} A(i-1,\theta+1)+1 & \text{if } \pi_i^{-1} > i+\theta \\ A(i-1,\theta+1) & \text{if } \pi_i^{-1} \le i+\theta \end{cases}$$
$$\text{if } i \ge 1 \text{ and } \theta = -\min(k+1,i).$$

The number of line segments crossing the scanline with indices $i$ and $\theta$ is $2A(i,\theta)+\theta$. It follows that the list $\mathcal{L}$ of $k$-small scanlines can be made in $O(nk)$ time.

We now show that the scanline graph $W_k(G)$ can be constructed in $O(nk)$ time. Consider two $k$-small scanlines $s$ and $t$ that are neighbors and that have a top endpoint in common, say between $i$ and $i+1$. Assume that the bottom endpoint of $s$ is between $\pi_{j-1}$ and $\pi_j$ and the bottom endpoint of $t$ is between $\pi_j$ and $\pi_{j+1}$. According to Lemma 6.2, we must have $|m-j| \le k+1$, otherwise there cannot be an arc between $s$ and $t$. Assume without loss of generality that $j < m$. Now notice that the number of vertices of $\mathcal{C}(s,t)$ is $j-i+A(i,j-i)+A(i,j-i-1)$. This shows that the adjacency list for each $k$-small scanline can be computed in $O(k)$ time. Computing a path in $W$ from $s_L$ to $s_R$, if it exists, clearly also takes $O(nk)$ time. Hence the total algorithm can be implemented to run in $O(nk)$ time. $\square$

THEOREM 6.4. *Let $G$ be a permutation graph. Then the pathwidth and treewidth of $G$ are the same, and there is an $O(nk)$ algorithm that correctly determines whether the treewidth of $G$ is at most $k$.*

We end this section by remarking that the algorithm can be adapted such that it computes, within the same time bound, the pathwidth of a permutation graph (when the matching diagram is given). This can be seen as follows. Let the treewidth of $G$ be $k$. First, compute a number $L$ such that $L \le k \le 2L$. This can be done, using the algorithm described above $O(\log k)$ times, in time $O(nk)$ (execute this algorithm for $L = 1, 2, 4, \ldots$). Now construct the scanline graph $W_{2L}(G)$ and put weights on the arcs that express how many vertices are in the corresponding candidate component. Then search for a path from $s_L$ to $s_R$ such that the maximum over weights of arcs in the path is minimized. This maximum weight minus one gives the exact treewidth $k$ of $G$.

## 7. Conclusions.

In this paper, we described a very simple and efficient algorithm to compute the treewidth and pathwidth of a permutation graph. In fact, we have shown that, for any permutation graph, the pathwidth and treewidth are equal.

Our results can easily be extended to complements of comparability graphs for which the partial order dimension is bounded by some fixed constant $d \geq 1$ (so-called cocomparability graphs of dimension $d$) using an intersection model that was introduced in [18] (see also [24]). Then the algorithm computing the pathwidth (or treewidth) has running time $O(nk^{d-1})$. Furthermore, it can be shown that, for any cocomparability graph, the pathwidth and treewidth are equal. (Notice that the problems PATHWIDTH and TREEWIDTH are both NP-complete when restricted to cocomparability graphs [2].)

Our methods can also be applied to the minimum fill-in problem, which is equivalent to minimizing the number of edges over all triangulations of the given graph. This leads to efficient algorithms computing a minimum fill-in, namely an $O(n^2)$ algorithm for permutation graphs and an $O(n^d)$ algorithm for cocomparability graphs of dimension $d$. It is important to mention that the algorithms on cocomparability graphs of dimension $d \geq 3$ usually require an intersection model as part of the input, since their recognition problem is NP-complete [31]. Nevertheless, it has been shown recently that, using a different approach, a polynomial-time algorithm computing the pathwidth (or treewidth) for cocomparability graphs of dimension $d \geq 3$ can be designed that does not require an intersection model as part of the input [22], [24].

There are some other classes of graphs for which the exact pathwidth and treewidth can be computed efficiently. For example, cographs [10], split graphs, and interval graphs. The treewidth can also be computed efficiently for chordal graphs, circular arc graphs [29], and chordal bipartite graphs [23]. In this respect, we would like to mention the result of [19], which shows that the problem PATHWIDTH is NP-complete when restricted to chordal graphs. Recently, polynomial-time algorithms computing the treewidth of circle graphs [20] and the minimum-fill-in of chordal bipartite graphs [21] have been shown. For more details we refer to the monograph [22].

**Acknowledgments.** We would like to thank A. Brandstädt, H. Müller, A. Jacobs, D. Seese, and B. Reed for valuable discussions.

## REFERENCES

[1] S. ARNBORG, *Efficient algorithms for combinatorial problems on graphs with bounded decomposability : A survey*, BIT, 25 (1985), pp. 2–23.

[2] S. ARNBORG, D. G. CORNEIL, AND A. PROSKUROWSKI, *Complexity of finding embeddings in a k-tree*, SIAM J. Algebraic Discrete Meth., 8 (1987), pp. 277–284.

[3] S. ARNBORG, J. LAGERGREN, AND D. SEESE, *Easy problems for tree-decomposable graphs*, J. Algorithms, 12 (1991), pp. 308–340.

[4] S. ARNBORG AND A. PROSKUROWSKI, *Linear-time algorithms for NP-hard problems restricted to partial k-trees*, Discrete Appl. Math., 23 (1989), pp. 11–24.

[5] C. BERGE AND C. CHVATAL, *Topics on Perfect Graphs*, Ann. Discrete Math., 21 (1984).

[6] H. L. BODLAENDER, *Achromatic number is NP-complete for cographs and interval graphs*, Inform. Process. Lett., 31 (1989), pp. 135–138.

[7] ———, *Dynamic programming algorithms on graphs with bounded treewidth*, Lecture Notes in Comput. Sci., 317 (1988), pp. 105–119.

[8] ———, *A linear-time algorithm for finding tree-decompositions of small treewidth*, Proc. 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 226–234.

[9] ———, *A tourist guide through treewidth*, Acta Cybernet., 11 (1993), pp. 1–23.

[10] H. L. BODLAENDER AND R. H. MÖHRING, *The pathwidth and treewidth of cographs*, SIAM J. Discrete Math., 6 (1993), pp. 181–188.

[11] A. BRANDSTÄDT, *Special graph classes: A survey*, Schriftenreihe des Fachbereichs Mathematik, SM-DU-199, Universität Duisburg Gesamthochschule, Duisburg, Germany, 1991.

[12] A. BRANDSTÄDT AND D. KRATSCH, *On the restriction of some NP-complete graph problems to permutation graphs*, Lecture Notes in Comput. Sci., 199 (1985), pp. 53–62.

[13] A. Brandstädt and D. Kratsch, *On domination problems for permutation and other graphs*, Theoret. Comput. Sci., 54 (1987), pp. 181–198.

[14] S. Even, A. Pnueli, and A. Lempel, *Permutation graphs and transitive graphs*, J. Assoc. Comput. Mach., 19 (1972), pp. 400–410.

[15] M. Farber and M. Keil, *Domination in permutation graphs*, J. Algorithms, 6 (1985), pp. 309–321.

[16] J. Gimbel, D. Kratsch, and L. Stewart, *On cocolourings and cochromatic numbers of graphs*, Discrete Appl. Math., 48 (1994), pp. 111–127.

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

[18] M. C. Golumbic, D. Rotem, and J. Urrutia, *Comparability graphs and intersection graphs*, Discrete Math., 43 (1983), pp. 37–46.

[19] J. Gustedt, *On the pathwidth of chordal graphs*, Discrete Appl. Math., 45 (1993), pp. 233–248.

[20] T. Kloks, *Treewidth of circle graphs*, Lecture Notes in Comput. Sci., 762 (1993), pp. 108–117.

[21] ———, *Minimum fill-in for chordal bipartite graphs*, Tech. report RUU-CS-93-11, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, 1993.

[22] ———, *Treewidth : Computations and approximations*, Lecture Notes in Comput. Sci., 842 (1994).

[23] T. Kloks and D. Kratsch, *Treewidth of chordal bipartite graphs*, Lecture Notes in Comput. Sci., 665 (1993), pp. 80–89.

[24] T. Kloks, D. Kratsch, and J. Spinrad, *Treewidth and pathwidth of cocomparability graphs of bounded dimension*, Note 93/46, Department of Computing Science, Technical University Eindhoven, Eindhoven, The Netherlands, 1993.

[25] J. van Leeuwen, *Graph algorithms*, in Handbook of Theoretical Computer Science, A: Algorithms and Complexity Theory, North-Holland, Amsterdam, 1990, pp. 527–631.

[26] C. G. Lekkerkerker and J. Ch. Boland, *Representation of a finite graph by a set of intervals on the real line*, Fund. Math., 51 (1962), pp. 45–64.

[27] A. Sen, H. Deng, and S. Guha, *On a graph partition problem with application to VLSI layout*, Inform. Process. Lett., 43 (1992), pp. 87–94.

[28] J. Spinrad, *On comparability and permutation graphs*, SIAM J. Comput., 14 (1985), pp. 658–670.

[29] R. Sundaram, K. Sher Singh, and C. Pandu Rangan, *Treewidth of circular arc graphs*, SIAM J. Discrete Math., 7 (1994), pp. 647–655.

[30] K. Wagner, *Monotonic coverings of finite sets*, J. Inform. Process. Cybernet., 20 (1984), pp. 633–639.

[31] M. Yannakakis, *The complexity of the partial order dimension problem*, SIAM J. Algebraic Discrete Meth., 3 (1982), pp. 351–358.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 617–637, November 1995

# SALVAGE-EMBEDDINGS OF COMPLETE TREES*

SANDEEP N. BHATT†, FAN R. K. CHUNG‡, FRANK THOMSON LEIGHTON§, AND ARNOLD L. ROSENBERG¶

**Abstract.** A *salvage-embedding* (*S-embedding*) maps an $M$-leaf complete binary tree $\mathcal{G}$ into an $(N > M)$-leaf complete binary tree $\mathcal{H}$, the fraction $G$ of whose leaves have been labeled GOOD. The S-embedding maps leaves of $\mathcal{G}$ one-to-one to GOOD leaves of $\mathcal{H}$; it may be many-to-one on internal nodes. The quality of an S-embedding depends on its *harvest*, the ratio $H \overset{\text{def}}{=} M/GN$, and its *congestion*, the largest number of edges of $\mathcal{G}$ that get "routed" across the same edge of $\mathcal{H}$. We study three scenarios. In the *worst-case* scenario, given any target harvest $H \leq 1/2$, one can S-embed a $2^{\lfloor \log(HGN) \rfloor}$-leaf $\mathcal{G}$ in $\mathcal{H}$ with congestion $\log\log N +$ a constant depending only on $G$ and $H$, no matter how the GOOD leaves are distributed; this congestion cannot be lowered by more than a small constant factor. In the *expected-case* scenario—where leaves of $\mathcal{H}$ are labeled GOOD or not, independently, with fixed probability—with probability exceeding $1 - N^{-\Omega(1)}$, for any target harvest $H \leq 1/8$, one can S-embed a $2^{\lfloor HN \rfloor}$-leaf $\mathcal{G}$ in $\mathcal{H}$ with congestion $O(\log\log\log N)$. In the *salvaging* scenario, we present an algorithm that, in time $O(CN(\log N)^{3C+2})$, S-embeds in a given a leaf-labeled $\mathcal{H}$ the largest possible $\mathcal{G}$, subject to the prespecified bound $C$ on congestion. This work is inspired by the problem of salvaging a fault-free subnetwork of a *leaf-tree machine*—a tree architecture whose leaves hold "full-power" processors and whose nonleaf nodes hold "rudimentary" processors that route messages and perform simple combining tasks.

**Key words.** synchronization networks, fault tolerance, congestion, dilation

**AMS subject classification.** 6805C

## 1. Introduction.

This paper studies *salvage-embeddings* (*S-embeddings*) of small complete binary trees into large ones, inspired by the problem of tolerating faults in a type of tree-structured parallel architecture that we call a *leaf-tree machine*. Our study is among the first that obtains nontrivial bounds on the efficiency of embeddings solely on the basis of congestion.

### 1.1. Basic notions.

#### 1.1.1. Trees and forests.

The *height-$n$ complete binary tree $\mathcal{T}_n$* is the graph whose $2^{n+1} - 1$ nodes comprise the set of all binary words of length at most $n$ and

* Received by the editors November 6, 1992; accepted for publication (in revised form) September 23, 1994.

† Bell Communications Research, 435 South Street, Morristown, New Jersey 07960. The research of this author was supported in part by National Science Foundation grants MIP-86-01885 and CCR-88-07426, by National Science Foundation-Defense Advanced Research Projects Agency grant CCR-89-08285, and by Air Force Office of Science Research grant AFOSR-89-0382. Portions of this research were performed while this author was with the Department of Computer Science, Yale University, New Haven, Connecticut 06520 and visiting Bell Communications research.

‡ Department of Mathematics, University of Pennsylvania, Philadelphia, Pennsylvania 19104. Portions of this research were performed while this author was at Bell Communications Research.

§ Department of Mathematics and Laboratory for Computer Science, Massachusetts Institute for Technology, Cambridge, Massachusetts 02139. The research of this author was supported by Air Force contract OSR-86-0076, Defense Advanced Research Projects Agency Contract N00014-80-C-0622, United States Army contract DAAL-03-86-K-0171, and a National Science Foundation Presidential Young Investigator Award with matching funds from AT&T and IBM. Portions of this research were performed while this author was visiting Bell Communicatons Research.

¶ Department of Computer Science, University of Massachusetts, Amherst, Massachusetts 01003. The research of this author was supported in part by National Science Foundation grants CCR-90-13184 and CCR-92-21785. Portions of this research were performed while this author was visiting Bell Communications Research and the Department of Computer Science, Technion, Haifa 32000, Israel.

whose edges connect each node $x$ of length less than $n$ with its *children* $x0$ and $x1$. For each $\ell \in \{0, 1, \ldots, n\}$, the $2^\ell$ words/nodes of length $\ell$ form *level* $n - \ell$ of $\mathcal{T}_n$;[1] the unique node at level $n$ is the *root* of $\mathcal{T}_n$, and the $2^n$ nodes at level 0 are the *leaves* of $\mathcal{T}_n$. We say that node $x$ is a (*proper*) *ancestor* of node $y$, or, equivalently, that node $y$ is a (*proper*) *descendant* of node $x$, just when the string $x$ is a (proper) prefix[2] of the string $y$. For each node $x$ of $\mathcal{T}_n$, the *subtree of* $\mathcal{T}_n$ *rooted at* $x$ is the induced subgraph of $\mathcal{T}_n$ on the nodes $\{xy \ : \ 0 \leq |y| \leq n - |x|\}$, i.e., the set of descendants of $x$. Henceforth, we refer to "complete binary trees" simply as "trees."

Finally, our study calls for a nonstandard notion of forest. For our purposes, a *forest* is a nonempty *sequence* of trees *of distinct sizes.*

**1.1.2. Salvage-embeddings.** Let us be given a tree $\mathcal{T}_n$, the fraction $0 < G \leq 1$ of whose $2^n$ leaves have been labeled GOOD; call the fraction $G$ the *yield* of the labeling. Let us further be given a tree $\mathcal{T}_k$, where $2^k \leq G2^n$. We wish to embed $\mathcal{T}_k$ into $\mathcal{T}_n$, using only the GOOD leaves of the latter; for mnemonic emphasis, we henceforth denote by $\mathcal{G}_k$ (for *guest*) the copy of $\mathcal{T}_k$ and by $\mathcal{H}_n$ (for *host*) the (leaf-labeled) copy of $\mathcal{T}_n$.

A *salvage-embedding* (*S-embedding*, for short) *of* $\mathcal{G}_k$ *into* $\mathcal{H}_n$*,* where $2^k \leq G2^n$, is given by the following:

- an assignment $\alpha$ of the nodes of $\mathcal{G}_k$ to nodes of $\mathcal{H}_n$ that
  - maps each leaf of $\mathcal{G}_k$ to a unique GOOD leaf of $\mathcal{H}_n$ (and so is one-to-one on the leaves of $\mathcal{G}_k$), and
  - is *progressive* in that it preserves all ancestor–descendant relations among nodes of $\mathcal{G}_k$;[3]
- a routing function $\rho$ that assigns to each edge $(x, y)$ of $\mathcal{G}_k$ the unique path in $\mathcal{H}_n$ that connects nodes $\alpha(x)$ and $\alpha(y)$.

While our primary interest here is in S-embedding an individual tree $\mathcal{G}_k$ into $\mathcal{H}_n$, we usually will know the parameter $k$ only implicitly: $\mathcal{G}_k$ will usually be defined as the largest tree that can be S-embedded into $\mathcal{H}_n$ within a certain cost bound. Our S-embedding algorithms therefore use the technical device of S-embedding entire forests of trees into $\mathcal{H}_n$; the largest tree in the S-embedded forest will be the sought one. By extension of our definition of S-embedding trees, an *S-embedding of a forest of trees into* $\mathcal{H}_n$ is a set of S-embeddings of the trees in the forest *whose leaf assignments are node disjoint.* This node disjointness guarantees that if any subset of the trees in the forest are grown and combined into a single tree, an S-embedding of that single tree into $\mathcal{H}_n$ can use the leaf assignments of the S-embedding of the forest into $\mathcal{H}_n$.

**1.1.3. The costs of an S-embedding.** Let us be given an S-embedding $\langle \alpha, \rho \rangle$ of the $m$-tree forest $\langle \mathcal{G}_{k_1}, \mathcal{G}_{k_2}, \ldots, \mathcal{G}_{k_m} \rangle$, where each[4] $k_j < k_{j+1}$, into $\mathcal{H}_n$. We are interested in two costs of the S-embedding.

The *harvest* of the S-embedding is the ratio of the number of leaves in the largest tree in the forest (namely, $\mathcal{G}_{k_m}$) to the number of GOOD leaves in $\mathcal{H}_n$; symbolically,

$$\text{Harvest}(\langle \alpha, \rho \rangle) = \frac{2^{k_m - n}}{G}.$$

[1] For later notational convenience, we number tree levels from the leaves toward the root, rather than in the more conventional opposite direction.

[2] String $x$ is a *proper prefix* of string $y$ just when there is a nonnull string $z$ such that $y = xz$.

[3] The algorithmic benefits of progressiveness are mentioned in §1.3, where the motivating machine model is discussed.

[4] We insist that the trees in a forest strictly increase in size because we always seek the largest tree that can be S-embedded efficiently into $\mathcal{H}_n$: equal-size trees can be combined into a bigger single tree.

This measure is of primary interest to us, because our algorithms are designed to maximize the size of the largest tree that can be S-embedded efficiently into $\mathcal{H}_n$.

The *simple congestion* of the S-embedding is measured by focusing just on those $\rho$-routing paths that are used to S-embed $\mathcal{G}_{k_m}$ into $\mathcal{H}_n$: it is the maximum number of such paths that cross any single edge of $\mathcal{H}_n$. Symbolically,

$$\text{Congestion}^{\star}(\langle\alpha,\rho\rangle) = \max_{(x,y)\in\text{Edges}(\mathcal{H}_n)} |\{(u,v)\in\text{Edges}(\mathcal{G}_{k_m}) \mid \rho(u,v) \text{ contains edge } (x,y)\}|.$$

On the road to finding a small-congestion S-embedding of a large tree into $\mathcal{H}_n$, our algorithms will want to keep track of the congestion of the current intermediate S-embeddings of entire forests into $\mathcal{H}_n$, since trees from these intermediate forests will ultimately be combined to form the sought large tree. This is the congestion measure we focus on most of the time.

$$\text{Congestion}(\langle\alpha,\rho\rangle) =$$

$$\max_{(x,y)\in\text{Edges}(\mathcal{H}_n)} \left| \bigcup_{1\le i\le m} \{(u,v)\in\text{Edges}(\mathcal{G}_{k_i}) \mid \rho(u,v) \text{ contains edge } (x,y)\} \right|.$$

**1.2. Accomplishments.** Throughout we consider the tree $\mathcal{H}_n$ with $N = 2^n$ leaves. We denote the yield of $\mathcal{H}_n$ by $G$; i.e., we assume that the fraction $0 < G \le 1$ of $\mathcal{H}_n$'s leaves have been labeled GOOD.

**1.2.1. The problems of interest.**

THE CONGESTION-HARVEST TRADEOFF PROBLEMS. *Given n, G, and the desired harvest*[5] $0 < H \le 1/2$, *determine the smallest congestion* $C = C(n,G,H)$ *for which there is a congestion-C S-embedding of* $\mathcal{G}_{\lfloor\log(HG2^n)\rfloor}$ *into* $\mathcal{H}_n$:

(a) *in the* worst-case scenario, *i.e., no matter how the* GOOD *leaves are distributed in* $\mathcal{H}_n$;

(b) *in the* expected scenario, *i.e., with probability* $\ge 1 - 2^{-\Omega(n)}$, *when leaves of* $\mathcal{H}_n$ *are labeled* GOOD *or not, independently, with some fixed probability.*

In both of the congestion-harvest tradeoff problems, we assume that the harvest fraction $H = \Theta(1)$; i.e., we aim for a harvest that is a fixed fraction of the number of GOOD leaves.

THE HARVEST-MAXIMIZATION PROBLEM. *Given an allowable congestion (i.e., an integer)* $C < n$, *find the largest k for which there is an S-embedding of* $\mathcal{G}_k$ *into* $\mathcal{H}_n$ *with congestion* $\le C$.

In this last problem, we insist that $C < n$, since the problem of S-embedding trivializes when $C \ge n$.

**1.2.2. The results obtained.**

[5] We cannot aim at harvests exceeding 1/2: the number of harvested leaves must be a power of 2, but the largest power of 2 not exceeding $G2^n$, namely, $2^{\lfloor\log(HG2^n)\rfloor}$, may be close to $G2^{n-1}$.

THE WORST-CASE CONGESTION-HARVEST TRADEOFF. *For any yield* $0 < G \leq 1$ *and harvest* $0 < H \leq 1/2$, *one can S-embed* $\mathcal{G}_{\lfloor \log(HG2^n) \rfloor}$ *into* $\mathcal{H}_n$ *with congestion*[6] [7]

$$\log^{(2)} N + \kappa(G, H)$$

*no matter how the* GOOD *leaves are distributed in* $\mathcal{H}_n$ (*Theorem* 2.1). *Moreover, in general, this amount of congestion is necessary, to within a constant factor*; *i.e., there exist yields G with associated patterns of GN* GOOD *leaves for which any S-embedding of* $\mathcal{G}_{\lfloor \log(HG2^n) \rfloor}$ *into* $\mathcal{H}_n$ *has congestion* $\kappa(G, H) \log^{(2)} N$ (*Theorem* 2.2).

THE EXPECTED-CASE CONGESTION-HARVEST TRADEOFF. *For any harvest* $0 < H \leq 1/8$, *if the leaves of* $\mathcal{H}_n$ *are labeled* GOOD *or not, independently, with probability* 1/2 *(an arbitrary fixed constant), then, with probability exceeding* $1 - N^{-\Omega(1)}$, *one can embed* $\mathcal{G}_{\lfloor \log(H2^n) \rfloor}$ *into* $\mathcal{H}_n$ *with congestion*

$$\log^{(3)} N + \kappa(H)$$

(*Theorem* 3.1).

It remains an inviting challenge to determine whether or not a smaller congestion suffices.

THE HARVEST-MAXIMIZATION ALGORITHM. *For any congestion bound C and any labeling of the leaves of* $\mathcal{H}_n$, *one can determine in time* $O(Cn^{3C+2}2^n)$ *the largest* $\mathcal{G}_k$ *that can be S-embedded into* $\mathcal{H}_n$ *with congestion* $\leq C$ (*Theorem* 4.1).

Note that this time is a low-degree polynomial in $N = 2^n$ even when $C$ is as large as $\frac{1}{3} \log(N/\log N)$.

We study the worst-case congestion-harvest tradeoff problem in §2, the expected-case congestion-harvest tradeoff problem in §3, and the harvest-maximization problem in §4. It is worth stressing here that our study focuses on developing algorithms that effect S-embeddings with certain costs, rather than merely on proving that such S-embeddings exist. In other words, all of our upper-bound proofs are constructive.

**1.3. The inspiration for our machine model.** The idea of studying S-embeddings was suggested by the problem of tolerating faults in a genre of tree-machine that has appeared several times in the literature. We wish to stress that, although our combinatorial problem was inspired by this "real-life" problem, we do not insist that our solution to the combinatorial problem is the definitive solution to the "real" problem.

A *leaf-tree machine* (LTM for short) is a parallel architecture consisting of $N$ "full-power" processing elements (PEs), which do the "real" computing, and $N - 1$ "rudimentary" PEs, which do simple auxiliary tasks such as routing and broadcasting messages and performing simple combining and accumulating tasks. The $2N - 1$ PEs are interconnected by a network having the topology of a complete binary tree: the leaf nodes of the tree hold the "full-power" PEs of the LTM; the nonleaf nodes hold the "rudimentary" PEs.

In our formal setting, the leaf nodes of $\mathcal{H}_n$ represent the "full-power" PEs of the LTM; the nonleaf nodes represent the LTM's "rudimentary" PEs.

[6] All logarithms are to the base 2. The iterated logarithm $\log^{(k)}$ is defined by:
$$\log^{(1)} N = \log N; \qquad \log^{(k+1)} N = \log \log^{(k)} N.$$

[7] For given parameters $A, B, \ldots, Z$, we denote by $\kappa(A, B, \ldots, Z)$ a constant that depends only on the parameters; instances of "$\kappa(A, B, \ldots, Z)$" in different expressions may denote different constants. Thus, we use the $\kappa$-notation in very much the same way as the big-$O$ notation.

While interconnection networks with the topology of a tree are inherently inefficient due to the presence of communication bottlenecks, research has shown that an LTM can be a useful *auxiliary network* when adjoined to a processor network having a richer topology (say, a mesh or hypercube). A variety of computations were shown in [4] to yield to the simple, fast combining mechanism that is inherent in the structure of trees; these abilities of trees are exploited in [3] and [11], where LTMs are adjoined to data-processing machines for speedy searching, selection, and combining tasks; most recently, in [5], [6], LTMs have been adjoined to multiple instruction multiple data (MIMD) hypercubes with the end of using the trees' fast combining and broadcasting capabilities for processor synchronization, as well as other simple combining and broadcasting tasks.

The problem we study here is inspired by the fault-tolerance problem for LTMs. It is well known that aggressive very large scale integrated (VLSI) circuit designs are vulnerable to fabrication defects which are likely to disable some positive fraction of an architecture's PEs. In the context of LTMs, the "full-power" PEs are going to be much more vulnerable to such defects than are either the wires or the "rudimentary" PEs because the latter two, being small and structurally simple, can be designed using much more conservative design rules than the "full-power" PEs, and, hence, will be commensurately less vulnerable to both defects and faults; cf. [12]. Our study of S-embeddings abstracts one approach to the problem of rendering LTMs tolerant to defects in their "full-power," leaf PEs. Although this problem is nominally subsumed by studies of fault tolerance in tree architectures, such as [1], the special structure and mode of use of LTMs opens avenues to fault tolerance in LTMs that are exponentially more efficient than analogous techniques for general tree machines: we achieve the desired tolerance to faults merely by adding small-capacity queues to the edges of the LTM.

In our formal framework:

• $\mathcal{H}_n$ is the physical LTM; its GOOD leaves are the leaf PEs that are free of defects. The yield of the labeling is the yield of the leaf-PE fabrication process.

• $\mathcal{G}_k$ is the logical, ideal LTM we want to "salvage" from $\mathcal{H}_n$.

• The S-embedding is the salvage process; its congestion is the capacity of the largest edge-queue; its progressiveness ensures that, as in the ideal LTM, messages follow an up-then-down path in the salvaged LTM.

• The *dilation* [8] of an S-embedding is not relevant here because of the assumed speed of the ideal LTM relative to the speed of each individual PE; see, e.g., [5].

It is important to note that our abstraction deals only with the problem of tolerating defects in the LTM. We do *not* deal with the problem of tolerating defects in any primary network that interconnects the "full-power" PEs of the LTM, should such exist (say, as in the scenario in [5], [6]). There is abundant literature on techniques for salvaging the latter network, e.g., [2], [7], [9], [13].

**1.4. A fundamental observation.** The algorithms we present here depend in an essential way on a correspondence between the processes of adding binary representations of numbers on the one hand and devising S-embeddings on the other.

For each node $x$ of the leaf-labeled tree $\mathcal{H}_n$, define the *yield at* $x$, denoted $\text{Yield}(x)$, to be the number of GOOD leaves in the subtree rooted at $x$. Transparently, if $x$ is a leaf, then $\text{Yield}(x)$ is either 0 or 1, and if $x$ is not a leaf, then

$$\text{Yield}(x) = \text{Yield}(x0) + \text{Yield}(x1).$$

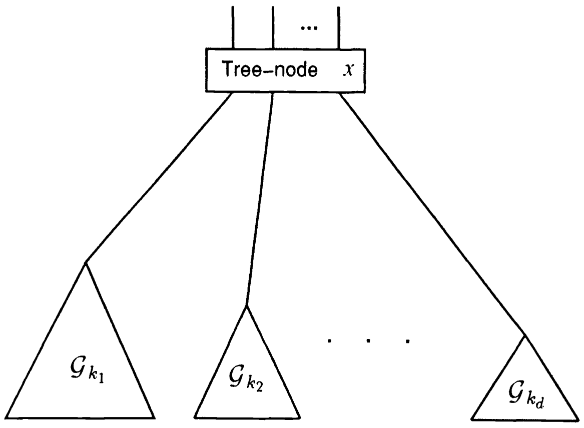


FIG. 1. *Binary representations and S-embeddings: The typical picture.*

The binary representation of the numbers Yield($x$) play a fundamental role in our study. The statement of the following lemma is depicted schematically in Fig. 1.

LEMMA 1.1. *Let $x$ be a node of $\mathcal{H}_n$. The following conditions are equivalent:*

(a) *The binary representation of* Yield($x$) *has 1's precisely in bit positions* $0 \le k_1 < k_2 < \cdots < k_d$.

(b) *There is an S-embedding of the forest* $\langle \mathcal{G}_{k_1}, \mathcal{G}_{k_2}, \ldots, \mathcal{G}_{k_d} \rangle$ *in* $\mathcal{H}_n$, *with congestion $d$ on the edge leading from node $x$ to its parent.*

*Proof.* The lemma follows by induction on the level of $x$ in $\mathcal{H}_n$.

The lemma is obvious in the base case, when $x$ is a leaf of $\mathcal{H}_n$, for then Yield($x$) is either 0 because $x$ is not GOOD or 1 because $x$ is GOOD, hence admitting a unit-congestion S-embedding of $\mathcal{G}_0$.

When $x$ is not a leaf, it has two children, $x0$ and $x1$. We establish a correspondence between the processes of adding the binary representations of Yield($x0$) and Yield($x1$) to obtain Yield($x$) on the one hand and of combining the forests $F_0$ and $F_1$ that are S-embedded at $x0$ and $x1$, respectively, to obtain the forest $F$ that is S-embedded at $x$ on the other hand. If we assume, by induction, that the lemma holds for $x0$ and $x1$, then the correspondence will extend the induction to $x$, and the lemma will follow. Figure 2 depicts the correspondence schematically.

Say that both Yield($x0$) and Yield($x1$) admit binary representations of length $m$, so Yield($x$) admits one of length $m + 1$. We construct the forest $F$ from the forests $F_0$ and $F_1$ in $m + 1$ stages, with the help of an auxiliary forest $C$ (which corresponds to the "carry bit" in binary addition). Initially, both $F$ and $C$ are empty. The $m + 1$ stages fall into three classes that are depicted in Tables 1 and 2.

TABLE 1
*Stage* 0.

| $\mathcal{G}_0 \in F_0$? | $\mathcal{G}_0 \in F_1$? | Action |
|---|---|---|
| No | No | No action |
| No | Yes | Add $\mathcal{G}_0$ to $F$ |
| Yes | No | Add $\mathcal{G}_0$ to $F$ |
| Yes | Yes | Add $\mathcal{G}_1$ to $C$ |

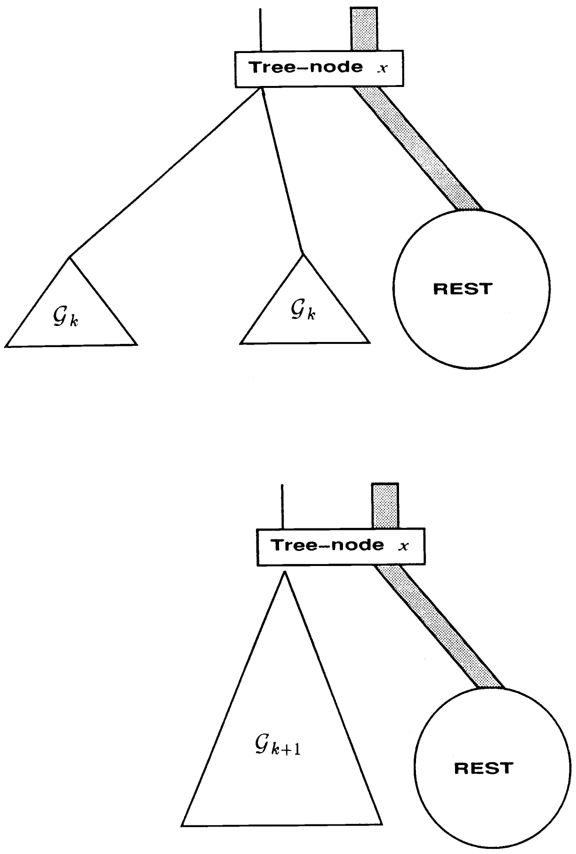


FIG. 2. *Binary representations and S-embeddings: Combining like-sized trees.*

TABLE 2
*Stages* $1 \leq i \leq m$.

| $\mathcal{G}_i \in F_0$? | $\mathcal{G}_i \in F_1$? | $\mathcal{G}_i \in C$? | Action |
|---|---|---|---|
| No | No | No | No action |
| No | No | Yes | Move $\mathcal{G}_i$ from $C$ to $F$ |
| No | Yes | No | Add $\mathcal{G}_i$ to $F$ |
| Yes | No | No | Add $\mathcal{G}_i$ to $F$ |
| No | Yes | Yes | Remove $\mathcal{G}_i$ from $C$; add $\mathcal{G}_{i+1}$ to $F$ |
| Yes | No | Yes | Remove $\mathcal{G}_i$ from $C$; add $\mathcal{G}_{i+1}$ to $F$ |
| Yes | Yes | No | Add $\mathcal{G}_{i+1}$ to $C$ |
| Yes | Yes | Yes | Add $\mathcal{G}_i$ to $F$, $\mathcal{G}_{i+1}$ to $C$ |

*Stage* $m+1$. If $\mathcal{G}_{m+1} \in C$, then add it to $F$.

The reader should easily recognize the correspondence between the two processes of interest. □

## 2. Optimizing worst-case congestion.

Given the leaf-labeled tree $\mathcal{H}_n$, the yield $0 < G \leq 1$, and a target harvest $0 < H \leq 1/2$, we wish to find an S-embedding of the tree $\mathcal{G}_{\lfloor \log(HG2^n) \rfloor}$ in the tree $\mathcal{H}_n$ that incurs as small congestion as possible (as a function of $n$, $G$, and $H$). This section is devoted to deriving both upper and lower bounds on the amount of congestion $C_{\min}$ that we must suffer in order to accomplish this task no matter how the GOOD leaves are distributed among the leaves of $\mathcal{H}_n$.

**2.1. An upper bound on worst-case behavior.** We formulate and analyze a "greedy" S-embedding algorithm which yields an upper bound on the quantity $C_{\min}$, which is optimal to within constant factors.

### 2.1.1. The algorithm.

**2.1.1.1. Overview.** The algorithm proceeds from level 0 to level $n$ in $\mathcal{H}_n$ (i.e., from the leaves toward the root), processing each node at level $\ell$ before proceeding to level $\ell+1$. As each level-$\ell$ node $x$ is encountered, the algorithm assigns the node a label $\lambda(x)$, which is the length-$(n+1)$ binary representation of the level-$\ell$ interim assessment of how many GOOD leaves can be harvested from the subtree rooted at $x$; we call this quantity the *level-$\ell$ potential of* $x$, denoted $\mathrm{Pot}(x)$. Thus, if

$$\lambda(x) = \lambda_n(x)\lambda_{n-1}(x)\cdots\lambda_0(x),$$

then

$$\mathrm{Pot}(x) = \sum_{i=0}^{n} \lambda_i(x)2^i.$$

The following features of the string $\lambda(x)$ are germane to our algorithm.

- $\mathrm{Ones}(\lambda(x)) = \{i \mid \lambda_i(x) \neq 0\}$ = the set of nonzero bit positions in $\lambda(x)$;
- $\mathrm{Wgt}(\lambda(x)) = |\mathrm{Ones}(\lambda(x))|$ = the *weight*, or number, of nonzero bit positions in $\lambda(x)$.

By Lemma 1.1, if node $x$ of $\mathcal{H}_n$ receives label $\lambda(x)$, then there is an S-embedding of the forest $\{\mathcal{G}_k \mid XS\ k \in \mathrm{Ones}(\lambda(x))\}$ in the subtree of $\mathcal{H}_n$ rooted at $x$, with congestion $\mathrm{Wgt}(\lambda(x))$. Note that the progressiveness of S-embeddings implies that $\lambda_k(x) = 0$ for all $k > \mathrm{level}(x)$. Because of Lemma 1.1, our S-embedding algorithm is fundamentally a labeling algorithm.

**2.1.1.2. The labeling/embedding procedure.** Say that $C$ is the maximum congestion we are willing to allow in any S-embedding. We exploit the fact that all labels have the same length (namely, $n+1$) by specifying each label $\lambda(x)$ implicitly via the integer $\mathrm{Pot}(\lambda(x))$. In overview, our labeling/embedding algorithm proceeds from level 0 to level $n$ in $\mathcal{H}_n$, labeling each node at level $\ell$ before any node at level $\ell+1$. A node's label is chosen as follows.

1. If node $v$ is a GOOD *leaf*, then $\mathrm{Label}(v) \leftarrow 1$.
2. If node $v$ is a *non*GOOD *leaf*, then $\mathrm{Label}(v) \leftarrow 0$.
3. If node $v$ is a *nonleaf*, then
   (a) Add the labels of $v$'s children.
   (b) "Prune" the sum: keep only the $C$ highest-order 1's.
   (c) Perform the S-embeddings mandated by the carries in the label additions at the nodes.

The detailed version of our labeling/embedding algorithm is called Algorithm WORST-CASE.

ALGORITHM WORST-CASE

**Step 0.** {Label nodes on level 0 of $\mathcal{H}_n$}

Scan the leaves of $\mathcal{H}_n$, assigning each leaf $x$ a label $\lambda(x)$ as follows.

$$\mathrm{Pot}(\lambda(x)) = \begin{cases} 1 & \text{if } x \text{ is GOOD} \\ 0 & \text{if } x \text{ is not GOOD} \end{cases}$$

```
Step $\ell > 0$. {Label nodes on level $\ell > 0$ of $\mathcal{H}_n$}
  Scan the nodes at level $\ell$ of $\mathcal{H}_n$.
    Substep $\ell.a$ {Assign the string label}
      Assign each level-$\ell$ node $x$ a label $\lambda(x)$ as follows.

                $\text{Pot}(\lambda(x)) = \text{Pot}(\lambda(x0)) + \text{Pot}(\lambda(x1))$

    Substep $\ell.b$ {Combine small embedded trees}
      if there was a chain of carries from bit positions $k-i, k-i+1, \ldots, k-1$
      of $\lambda(x0)$ and $\lambda(x1)$ into bit position $k$ of $\lambda(x)$
      then embed the roots of copies of $\mathcal{G}_{k-i+1}, \ldots, \mathcal{G}_k$ in node $x$, and route
      edges from those roots, along shortest paths, to the roots of two copies
      each of $\mathcal{G}_{k-i}, \ldots, \mathcal{G}_{k-1}$ that are embedded in proper descendants of $x$
      endif
    Substep $\ell.c$ {Honor the congestion bound $C$}
      for $k = 0$ to $\lceil \log \text{Pot}(\lambda(x)) - C \rceil$
      if $\text{Wgt}(\lambda(x)) > C$ then $\lambda_k(x) \leftarrow 0$ endif
      endfor
```

### 2.1.2. Worst-case behavior of Algorithm Worst-Case.

THEOREM 2.1. *Let some $GN$ leaves of $\mathcal{H}_n$ be labeled* GOOD, *in any way, for some $0 < G \le 1$. For any rational $0 < H \le 1/2$, Algorithm* WORST-CASE *finds an S-embedding of $\mathcal{G}_{\lfloor \log(HGN) \rfloor}$ in $\mathcal{H}_n$, with congestion*

$$C \le \log^{(2)} N - \log((1-H)G) + 1. \tag{2.1}$$

*Proof.* It is clear that, for each node $x$ of $\mathcal{H}_n$, Algorithm WORST-CASE S-embeds $\mathcal{G}_{\lfloor \log \text{Pot}(\lambda(x)) \rfloor}$ in the subtree of $\mathcal{H}_n$ rooted at node $x$; hence, overall, the algorithm S-embeds the tree $\mathcal{G}_{\lfloor \log \text{Pot}(\lambda(r)) \rfloor}$ in $\mathcal{H}_n$, where $r$ is the root of $\mathcal{H}_n$. We need only verify that the salvaged tree represents a big enough harvest when $C$ is as big as the bound in inequality (2.1).

Note first that Algorithm WORST-CASE never requires us to abandon any GOOD leaves as we work up from level 0 through level $C-1$ of $\mathcal{H}_n$, because the high-order bit of $\text{Pot}(\lambda(x))$ can be no greater than the level of $x$ in $\mathcal{H}_n$. To see what happens above this level, focus on a specific (but arbitrary) node $x$ at a specific (but arbitrary) level $\ell \ge C$ of $\mathcal{H}_n$. The congestion bound may require us, in Substep $\ell.c$ of the algorithm, to abandon one bit in each position $k \le \ell - C$ of $\lambda(x)$. This is equivalent to abandoning one GOOD-leafed copy of each tree $\mathcal{G}_k$ with $k \le \ell - C$; however, at most one tree of each size is abandoned because any two trees of the same size would have been coalesced (by embedding a new root) at this step, if not earlier. It follows that, when the algorithm processes node $x$, it abandons no more than

$$\sum_{i=0}^{\ell-C} 2^i < 2^{\ell-C+1}$$

GOOD leaves; hence, at the entire level $\ell$, strictly fewer than

$$2^{\ell-C+1}2^{-\ell}N = 2^{-C+1}N$$

previously unabandoned GOOD leaves are abandoned. Thus, the entire salvage procedure abandons fewer than

$$(n+1-C)2^{-C+1}N$$

GOOD leaves due to congestion. Since there are $GN$ GOOD leaves in all, we see that more than

$$(G-(n+1-C)2^{1-C})N$$

GOOD leaves are *not* abandoned due to congestion. Now, at the end of the algorithm, we may have to abandon almost half of these unabandoned GOOD leaves—because the number of GOOD leaves we finally use in the S-embedding must be a power of 2. The algorithm will have succeeded in harvesting the desired fraction of GOOD leaves as long as the number of salvaged GOOD leaves, which we now see to be no fewer than

$$2^{\lfloor \log([G-(n+1-C)2^{1-C}]N)\rfloor},$$

is at least as large as the number of GOOD leaves we want to harvest from $\mathcal{H}_n$, which is

$$2^{\lfloor \log(HGN)\rfloor}.$$

Elementary estimates show that if we allow our S-embedding to have congestion $C$ as large as bound (2.1), then we shall have accomplished this task. □

### 2.2. A lower bound on worst-case behavior.

THEOREM 2.2. *Let $G$ and $H$ be rationals with $0 < G < 1$ and $0 < H \le 1/2$. For infinitely many $n$, there exists a way of labeling some $GN$ leaves of $\mathcal{H}_n$* GOOD *such that every S-embedding of some $\mathcal{G}_m$ in $\mathcal{H}_n$, where $2^m \ge HGN$, has congestion $C > \kappa(G,H)\log^{(2)} N$.*

*Proof.* Let us be given an algorithm, call it Algorithm A, that solves the worst-case congestion-harvest tradeoff problem. By §2.1, we know that the congestion incurred by Algorithm A for *any* labeling of the leaves of $\mathcal{H}_n$, in particular for the advertised malicious labeling, is $C \le \kappa(G,H)\log^{(2)} N$.

**2.2.1. The bad labeling.** The labeling of $\mathcal{H}_n$ that we claim defies efficient salvage is achieved via the following algorithm, wherein $L$ is a parameter we choose later. Once we choose $L$, let us restrict attention to values of $n$ that are multiples of $L$. (Clerical modifications accommodate all other values of $n$.)

```
ALGORITHM BAD-LABEL
    Step 1. {Mark leaves that will not be labeled GOOD}
        for each level $\ell$ from 0 to $n$ in steps of $L$
        Proceed left to right along the level-$\ell$ nodes of $\mathcal{H}_n$, marking all of the leaves
        in the subtree rooted at every $2^L$th node encountered.
        endfor
    Step 2. {Label the GOOD leaves}
        Label GOOD all leaves that are not marked in Step 1.
```

This labeling scheme can be viewed as turning $\mathcal{H}_n$ into a complete $(2^L-1)$-ary tree, all of whose leaves are labeled GOOD, providing that we look only at levels whose level-numbers are divisible by $L$. Therefore, our S-embedding problem now assumes some of the flavor of the problem of efficiently embedding a complete binary tree into a complete $(2^L - 1)$-ary tree that is only slightly larger (by roughly the factor $1/H$). The results in [8] about a similar problem lead us to expect the large congestion that we now show is inevitable.

**2.2.2. Bounds on $L$ and $C$.** Before considering our S-embedding problem, we must settle on a value for the parameter $L$. Our labeling of $\mathcal{H}_n$ has left the tree with $(2^L - 1)^{n/L}$ GOOD leaves. The worst-case congestion-harvest tradeoff problem assumes that the number of GOOD leaves in $\mathcal{H}_n$ is a positive fraction $G2^n$ of the total number of leaves. Elementary estimates show that this assumption implies that $L \geq \log n - \log^{(2)} n - \kappa(G)$. In fact, it will simplify our analysis to make an even stronger bound on $L$; namely, we assume henceforth that $2^L - 1 \geq n$. In analyzing our putative Algorithm A, it is convenient to assume additionally that we are dealing with S-embeddings whose congestions satisfy $C < \frac{1}{3}(L-1)$ (or else, we have nothing to prove).[8]

**2.2.3. The analysis.** For each $\ell \in \{0, 1, \ldots, n/L\}$, establish the following notation.

- $N(\ell)$ denotes the number of leaves in the largest GOOD-leafed forest that can be S-embedded at a level-$\ell L$ node of $\mathcal{H}_n$, independent of the congestion-bound $C$. As noted earlier,

$$N(\ell) = (2^L - 1)^\ell.$$

- $M(\ell; C)$ denotes the number of leaves in the largest GOOD-leafed forest that Algorithm A S-embeds at a level-$\ell L$ node of $\mathcal{H}_n$, given the congestion-bound $C$.

Our overall strategy will be to estimate, as a function of $C$, the rate of change of the ratio

$$\Delta(\ell; C) \stackrel{\text{def}}{=} \frac{M(\ell; C)}{N(\ell)}$$

as $\ell$ increases; we thereby convert our graph-theoretic problem into a number-theoretic one. We obtain the sought bound on $C$ by showing that $C$ must be "big" if the ratio $\Delta(\ell; C)$ is to remain $\geq H$ even when $\ell$ achieves its upper limit $n/L$.

Note first that, even if there were no bound on congestion, the similarity of the labeled version of $\mathcal{H}_n$ and a complete $(2^L - 1)$-ary tree would guarantee that, for $0 \leq \ell < n/L$,

$$M(\ell + 1; C) \leq (2^L - 1) M(\ell; C).$$

In order to appreciate the effect of the congestion-bound $C$, note that, when the S-embedding of Algorithm A has congestion $\leq C$, each number $M(\ell; C)$ must be representable as the sum of no more than $C$ powers of 2; in other words, the binary representation of $M(\ell; C)$ can have weight no greater than $C$.[9]

[8] This assumption about $C$ simplifies the estimates in the upcoming argument.

[9] This is true because we focus on *progressive* S-embeddings. If we allowed arbitrary S-embeddings, then $M(\ell; C)$ would be the *algebraic* sum—i.e., the sum/difference—of at most $C$ powers of 2. The added generality of arbitrary S-embeddings would influence only constant factors in our bounds.

Our discussion of the weights of the binary representations of the numbers $M(\ell; C)$ implies, in particular, that

$$M(1;C) \leq 2^L - 2^{L-C} = (2^C - 1)2^{L-C}. \tag{2.2}$$

Starting from this upper bound on $M(1;C)$, we derive upper bounds on a subset of the other numbers $M(\ell;C)$. Specifically, we find an integer $\ell^\star > 0$ such that

$$M(k\ell^\star + 1; C) = (2^C - 1)2^{L-C}2^{k(\ell^\star L - 1)} \tag{2.3}$$

for all integers

$$k \in \left\{0, 1, \ldots, \left\lfloor \frac{n}{\ell^\star L} \right\rfloor \right\}.$$

If we restrict attention, therefore, to values of $n$ such that $\ell^\star$ divides $(n/L) - 1$[10], we find, via the family of equations (2.3), that the harvest of Algorithm A can be no greater than

$$\Delta\left(\frac{n}{L}; C\right) = \frac{M(n/L;C)}{N(n/L)} = \left(1 - \frac{1}{2^C}\right)\left(1 + \frac{1}{2^L - 1}\right)^{n/L} 2^{(L-n)/(\ell^\star L)} \tag{2.4}$$

(after some simplification). The import of expression (2.4) will become clear only when we have obtained our bound on the integer $\ell^\star$. We turn now to the task of deriving this bound.

It will become clear as we proceed that we need focus only on the $L$ high-order bit positions of the shortest binary representations of the numbers $M(\ell; C)$; therefore, we henceforth number the bit positions of the representations from left to right, i.e., high order to low order, and we focus only on bit positions $1, 2, \ldots, L$.

Consider now a specific $\ell \in \{0, 1, \ldots, n/L-1\}$ and its associated $M(\ell; C)$, defining $M(0;C) = 1$ by convention. Note the effect of proceeding from $M(\ell; C)$ to $M(\ell+1; C)$, thence to $M(\ell + 2; C)$, and so on. Each step in this progression, say proceeding from $M(\ell + i; C)$ to $M(\ell + i + 1; C)$, consists of multiplying the "current" number, $M(\ell+i; C)$, by $2^L - 1$ (thereby going up $L$ levels in $\mathcal{H}_n$), followed by "pruning" all but the $C$ highest-order 1's in the product. Note that the multiplication part of this step affects only the rightmost 1 in bit positions $1, 2, \ldots, L$ of $M(\ell + i; C)$. Specifically, the effect of the multiply-then-"prune" step is as follows. Say that the rightmost 1 of $M(\ell + i; C)$ appears in bit position $k$. Then the $L$ high-order bit positions of $M(\ell + i; C)$ form a string

$$\xi_1\xi_2 \cdots \xi_{k-1}100 \cdots 0 \tag{2.5}$$

whose weight is

$$\mathrm{Wgt}(\xi_1\xi_2 \cdots \xi_{k-1}100 \cdots 0) = w \leq C \tag{2.6}$$

and whose terminal string of 0's has length $L - k$. The multiplication replaces this string with the like-length string

$$\xi_1\xi_2 \cdots \xi_{k-1}011 \cdots 1.$$

[10] This is the condition that implicitly defines the promised "infinitely many $n$" of the theorem.

When this product-string has weight exceeding $C$, the subsequent "pruning" replaces it by the like-length, weight-$C$ string

$$\xi_1\xi_2\cdots\xi_{k-1}011\cdots100\cdots0.$$

Note that the rightmost 1 in the resulting bit string has moved to a bit position $> k$. The reader can verify easily that in subsequent multiply-then-"prune" steps, the rightmost 1 continues to "migrate" rightward, in the sense illustrated in Fig. 3. The important thing to note in the figure is that eventually the 1 that started in bit position $k$ is *annihilated*, in the sense that it, and all 1's "spawned" by it in the course of the successive multiply-then-"prune" steps, disappear from the $L$ high-order bit positions of the sequence of numbers $M(\ell; C)$, to the detriment of the harvest-ratio $\Delta(\ell; C)$.

$$\boxed{\begin{array}{cl}
\xi_1\xi_2\cdots\xi_{k-1}10^{L-k} & \\
\Downarrow & \text{MULTIPLY by } 2^L-1 \\
\xi_1\xi_2\cdots\xi_{k-1}01^{L-k} & \\
\Downarrow & \text{PRUNE} \\
\xi_1\xi_2\cdots\xi_{k-1}01^{C-w}10^{L-C-k+w-1} & \\
\Downarrow & \text{MULTIPLY by } 2^L-1 \\
\xi_1\xi_2\cdots\xi_{k-1}01^{C-w}01^{L-C-k+w-1} & \\
\Downarrow & \text{PRUNE} \\
\xi_1\xi_2\cdots\xi_{k-1}01^{C-w}010^{L-C-k+w-2} & \\
\vdots & \vdots \\
\Downarrow & \text{MULTIPLY-and-PRUNE} \\
\xi_1\xi_2\cdots\xi_{k-1}0\eta_1\eta_2\cdots\eta_{L-k} & \\
\vdots & \vdots \\
\Downarrow & \text{MULTIPLY-and-PRUNE} \\
\xi_1\xi_2\cdots\xi_{k-1}0^{L-k+1} &
\end{array}}$$

FIG. 3. *The "migration" of the rightmost* 1.

We discover the sought integer $\ell^\star$ by analyzing the rate of "migration" of 1's in the binary representations of the numbers $M(\ell; C)$ as we perform multiply-then-"prune" steps. Specifically, we bound the number of multiply-then-"prune" steps it takes to annihilate a 1 that began in bit position $k \in \{2, 3, \ldots, C\}$ of $M(1; C)$. Since each such annihilation loses us a significant fraction of the GOOD leaves, we wish to maximize the stretches of time between annihilations. We accomplish this first by having $M(1; C)$ assume its maximum possible value (cf. bound (2.2)), and second by "pruning" as few leaves as possible after each multiplication. A corollary of this strategy is that we always strive to have the bit string in positions $1, 2, \ldots, L$ of our expression for the values $M(\ell; C)$ have maximum possible weight—but, of course, never more than $C$.

Let us focus on the 1 in bit position $k$ of $M(\ell + i; C)$; cf. equations (2.5, 2.6). Once this 1 begins to "migrate"—which occurs when it becomes the rightmost 1 in the surviving $L$ high-order bits—and until it is annihilated, the configuration of the $L$ high-order bits of $M(\ell + i; C)$ has the form

$$\xi_1\xi_2\cdots\xi_{k-1}0x,$$

where the bit string $x$ in bit positions $k+1, k+2, \ldots, L$ has weight $\leq C - w + 1$ (by equation (2.6)). Clearly, in the course of subsequent multiply-then-"prune" steps, while the initial bit string $\xi_1\xi_2\cdots\xi_{k-1}0$ remains intact, no terminal bit string $x$ recurs. (In fact, the numerical value of $x$ decreases monotonically during this phase of the migration.) It follows that the number of multiply-then-"prune" steps required to annihilate a 1 that resides in bit position $k$, when that 1 is the rightmost in the then-current $L$ high-order bits, cannot exceed

$$T(w,k) \overset{\text{def}}{=} \sum_{i=0}^{C-w+1} \binom{L-k}{i}, \tag{2.7}$$

since the summation yields the number of length-$(L-k)$ bit strings whose weights do not exceed $C-w+1$ and the terminal bit strings that replace $x$ will always have such length and weight. The notation $T(w,k)$ is appropriate here, since the number of steps depends only on the weight and length of the bit string $\xi_1\xi_2\cdots\xi_{k-1}$.

We are now ready to derive a bound on the value of $\ell^\star$. To this end, let us consider a sequence of multiply-then-"prune" steps, starting with any $M(a;C)$ whose $L$ high-order bits have the same form as those of $M(1;C)$, namely,

$$1^C0^{L-C}. \tag{2.8}$$

Say that we first perform enough multiply-then-"prune" steps to annihilate the 1 that resides in bit position $C$ of $M(a;C)$, so that the $L$ high-order bits of the then-current $M(\ell;C)$ have the form

$$1^{C-1}0^{L-C+1}.$$

The reasoning that leads to expression (2.7) says that this occurs after no more than $T(C,C)$ steps. Say next that we continue our series of multiply-then-"prune" steps until we have annihilated the 1 that resides in bit position $C-1$ of $M(a;C)$, so that the $L$ high-order bits of the then-current $M(\ell;C)$ have the form

$$1^{C-2}0^{L-C+2}.$$

Again invoking the reasoning that leads to (2.7), this requires at most an additional $T(C-1,C-1)$ steps. Continuing this reasoning, at most another $T(C-2,C-2)$ steps will allow us to annihilate the 1 that resides in bit position $C-2$ of $M(a;C)$, to arrive at $L$ high-order bits of the form

$$1^{C-3}0^{L-C+3}.$$

Finally, it is clear that, by the end of no more than

$$\sum_{i=0}^{C-2} T(C-i,C-i) = \sum_{i=0}^{C-2}\sum_{j=0}^{i+1} \binom{L-C+i}{j} \tag{2.9}$$

multiply-then-"prune" steps (starting with $M(a;C)$), we shall have annihilated every 1 that began in bit positions $2,3,\ldots,C$ of $M(a;C)$. At this point, the $L$ high-order bits of the then-current $M(\ell;C)$ will have the form

$$10^{L-1}.$$

Let us now perform one additional multiply-then-"prune" step. We claim that, starting from $M(a;C)$, we have performed the sought number of steps that we are seeking, namely, $\ell^\star$. To see this, note that in the current $M(a+\ell^\star;C)$:

• the $L$ high-order bits again have the form (2.8), as they did with $M(a;C)$; and

• since these $L$ high-order bits have weight $C$, we know that all other bits of the $M(a+\ell^\star;C)$ must be 0.

These two facts mean that

$$M(a+\ell^\star;C) = 2^{\ell^\star L-1}M(a;C). \tag{2.10}$$

(The term $-1$ in the exponent of $2^{\ell^\star L-1}$ reflects the "loss" of the high-order bit in the last multiplication.) The reader can easily replicate the reasoning leading to equation (2.10) to verify that the claimed value of $\ell^\star$ in fact satisfies the family of equations (2.3) that define the sought parameter $\ell^\star$.

It follows via the preceding reasoning (cf. expression (2.9)) that

$$\ell^\star \le \sum_{i=0}^{C-2}\sum_{j=0}^{i+1}\binom{L-C+i}{j}. \tag{2.11}$$

We can derive a more perspicuous and useful bound on $\ell^\star$ via the following easily verified fact about binomial coefficients.

LEMMA 2.3. *For all integers $R$ and $S$ satisfying $R > 3S+2$,*

$$\sum_{i=0}^{S}\binom{R}{i} < \binom{R}{S+1}.$$

Because $C < \frac{1}{3}(L-1)$ (by hypothesis), Lemma 2.3 allows us to replace inequality (2.11) by the simpler

$$\ell^\star < 2\sum_{i=0}^{C-2}\binom{L-C+i}{i+1} = 2\binom{L-1}{C-1} - 2 < 2\left(\frac{(L-1)e}{C-1}\right)^{C-1}. \tag{2.12}$$

At this point, we can return our attention to the bound (2.4) on the harvest of Algorithm A, evaluating that bound in the light of bound (2.12) on $\ell^\star$.

We are now prepared to investigate in detail the implications of our bound on $\ell^\star$ on the harvest $\Delta(n/L;C)$ of Algorithm A, as presented in (2.4). We begin by finding simple upper bounds on the expression (2.4), which will simplify our argument. These bounds rely on the following elementary facts:

$$\begin{aligned} 2^L - 1 &\ge n,\\ 1-2^{-C} &< 1,\\ (1+1/n)^n &\le e,\\ e^{1/L} &< 3. \end{aligned}$$

Applying these bounds to expression (2.4), we infer that

$$\Delta\left(\frac{n}{L};C\right) < \Gamma\left(\frac{n}{L};C\right) \overset{\text{def}}{=} 3\cdot 2^{(L-n)/(\ell^\star L)}. \tag{2.13}$$

Now, note that Algorithm A must have harvest $\ge H$. It follows, therefore, that we must have $\Gamma(n/L;C) > H$. This means, by manipulating expression (2.13), that

$$\ell^\star L > \kappa(H)(n-L).$$

By bound (2.12), this means that

$$(C-1)\log\left(\frac{(L-1)e}{C-1}\right) > \kappa(H)\log(n-L).$$

The reader can easily verify that this last inequality implies that $C > \kappa(H)\log n$, thereby completing the proof. □

**3. Optimizing expected congestion.** This section is devoted to deriving an analog of the development in §2 that exposes the amount of congestion one must incur in order to survive "random" faults in $\mathcal{H}_n$. In order to discuss random faults and the expected behavior of a salvage algorithm, we must have a fault model in mind. We adopt the model that predominates in the literature by assuming that the leaves of our trees $\mathcal{H}_n$ fail to be GOOD independently, with probability 1/2.[11] We turn now to the task of deriving an upper bound on $C_{\min}$ for random faults. We have not yet settled whether or not our bound on $C_{\min}$ can be lowered; this question presents an inviting challenge.

**3.1. An algorithm with good expected behavior.** Using the simple fault model just described, we find that a modified version of Algorithm WORST-CASE of §2 produces S-embeddings which incur congestion that is only *triply* logarithmic in the size of the harvested $\mathcal{G}_m$, with extremely high probability, providing that we lower our demands a bit. Specifically, we reduce our demand that our algorithm be able to harvest any fraction $H \le 1/2$ of the GOOD leaves of $\mathcal{H}_n$ to the demand that our algorithm be able to harvest any fraction $H \le 1/8$ of the GOOD leaves of $\mathcal{H}_n$.

THEOREM 3.1. *Let the leaves of $\mathcal{H}_n$ be labeled* GOOD *or not, independently, with probability* 1/2. *For any rational $0 < H \le 1/8$, with probability $\ge 1 - 2^{-\Omega(n)}$, a modification of Algorithm* WORST-CASE *will find an S-embedding having congestion*

$$C \le \log^{(3)} N - \log\frac{1-4H}{4} + 1 \tag{3.1}$$

*of some $\mathcal{G}_m$ in $\mathcal{H}_n$, where $2^m \ge HN$.*

*Proof.* The major insight leading to the desired modification of Algorithm WORST-CASE resides in the following combinatorial fact.

LEMMA 3.2. *Let the leaves of $\mathcal{H}_n$ be labeled* GOOD *or not, independently, with probability* 1/2. *If we partition the leaves of $\mathcal{H}_n$ into blocks of size* $10n$ *in any way, then with probability $\ge 1 - 2^{-\Omega(n)}$, at least* $2.5n$ *leaves in each block are* GOOD.

*Proof.* The proof proceeds by a series of transformations and estimates. Focus first on a single block of $10n$ leaves.

$$\begin{aligned}\Pr(< 2.5n \text{ GOOD leaves}) &= \Pr(> 7.5n \text{ not-GOOD leaves}) \\ &= 2^{-10n}\sum_{k=7.5n+1}^{10n} (\text{number of ways to choose } k \text{ not-GOOD leaves}).\end{aligned}$$

This last sum is readily transformed to

$$\sum_{k=0}^{2.5n-1}\binom{10n}{k}2^{-10n} \le 2\binom{10n}{2.5n}2^{-10n} \le 2\left(\frac{10e}{2.5}\right)^{2.5n}2^{-10n} \le 2\left(\frac{\epsilon}{2}\right)^{n}$$

[11] Changing the probability 1/2 to any other fixed probability $p$ merely changes the constants in our results.

for some $\epsilon < 1$. It follows that

$$\Pr(\geq 2.5n \text{ GOOD leaves}) \geq \left(1 - 2\left(\frac{\epsilon}{2}\right)^n\right)^{2^n/10n} \geq \exp(-c_1\epsilon^n/n) \geq 1 - \frac{1}{n2^{c_2 n}}$$

for some constants $c_1, c_2 > 0$. □

Lemma 3.2 tells us that when we look at the labels assigned by our greedy algorithm to nodes at or above level $\lceil \log 10n \rceil$ of a randomly labeled instance of $\mathcal{H}_n$, then, with very high probability, we find every node having a label $\lambda(x)$ for which $\text{Pot}(\lambda(x)) \geq 2.5n$. This suggests that the following (informally stated) S-embedding algorithm achieves the goals of the theorem with the indicated high probability. The reader should note that only Step 0 of the algorithm is not guaranteed to work as desired. To simplify exposition, we assume that $10n$ divides $2^n$ and that $2.5n$ is a power of 2; removing these assumptions is merely a clerical task.

```
ALGORITHM EXPECTED-CASE
 Step 0. {Select GOOD leaves of $\mathcal{H}_n$ for salvage}
     Scan the leaves of $\mathcal{H}_n$ in blocks of $10n$ leaves; within each block, select $2.5n$
     to be salvaged; remove the label GOOD from all unselected leaves.
 Step 1. {Salvage small trees within each block}
     Invoke Algorithm WORST-CASE within each block, with harvest fraction 1/2.
 Step 2. {Hook up all the small salvaged trees}
     Embed the "top" of a complete binary tree to hook together the $2^n/10n$ small,
     $2^{1.25n}$-leaf trees salvaged in Step 1.
```

By Lemma 3.2, Step 0 of Algorithm EXPECTED-CASE succeeds, with high probability, to collect the desired number of GOOD leaves within each block. By Theorem 2.1, the congestion incurred by Algorithm WORST-CASE when salvaging the small trees as mandated in Step 1 is no greater than the bound (3.1). Since Step 1 salvages just one tree from each block, the embedding of Step 2 incurs no further congestion. This completes the proof of Theorem 3.1; modulo details are left to the reader. □

## 4. Optimizing worst-case harvest.

Algorithm WORST-CASE of §2.1 is guaranteed to be efficient, both in running time—it operates in time $O(2^n)$, which is linear in the size of $\mathcal{H}_n$—and in harvest—it harvests the desired fraction $H$ of the GOOD leaves. But, it is easy to find examples where a nongreedy strategy allows one to harvest a much larger fraction of the GOOD leaves. In particular, when the GOOD leaves are spread out sparsely, any greedy approach abandons many more GOOD leaves than it has to. One finds an analogous deficiency in a "lazy" salvage strategy, one that coalesces small trees as late as possible rather than as early as possible; lazy strategies abandon too many GOOD leaves when the leaves are packed densely, in clumps. It might be of practical interest, therefore, to find a computationally efficient algorithm that harvests optimally many GOOD leaves while honoring a prespecified limit on congestion. This section presents such an algorithm.

### 4.1. The algorithm.

#### 4.1.1. Overview of the algorithm.

Our salvage algorithm keeps a record of all possible salvage decisions that are consistent with the bound on congestion (which we denote by $C$). It then selects as its harvest the largest of the trees in the record. It follows that the harvested tree has optimal size (given the bound $C$).

Our algorithm's record keeping consists of labeling each node $x$ at each level $\ell$ of $\mathcal{H}_n$ with a *set* $\Lambda(x)$ of length-$(\ell + 1)$ vectors of nonnegative integers. The set $\Lambda(x)$

records all possible salvage options available at $x$, given the bound $C$, with each vector $\vec{\nu} \in \Lambda(x)$ indicating one option. Specifically, say that $\vec{\nu} = \langle \nu_0, \nu_1, \ldots, \nu_\ell \rangle$; then:

- for all $0 \le k \le \ell$, there is an S-embedding in the subtree of $\mathcal{H}_n$ rooted at $x$ of a GOOD-leafed forest $\mathcal{F}$ containing $\nu_k$ disjoint copies of $\mathcal{G}_k$; and
- $\sum_{k=0}^{\ell} \nu_k \le C$, so that the bound on congestion is always honored.

When we get to the root $r$ of $\mathcal{H}_n$ (where $\ell = n$), the algorithm selects the largest $m$ for which some vector $\vec{\nu} \in \Lambda(r)$ has $\nu_m > 0$. Our harvest, then, is a GOOD-leafed copy of $\mathcal{G}_k$.

**4.1.2. Details of the algorithm.** We associate each level-$\ell$ node $x$ of $\mathcal{H}_n$ with a trie (i.e., a digital search tree [10]) of height $\ell - 1$. This trie will store the labeling-set $\Lambda(x)$ in the obvious way. We now present the details of Algorithm OPTIMAL-HARVEST.

```
ALGORITHM OPTIMAL-HARVEST
    Step 0. {Label nodes on level 0 of $\mathcal{H}_n$}
        Assign each leaf $x$ a labeling set $\Lambda(x)$, as follows.
            $\Lambda(x) = \begin{cases} \{\langle 1 \rangle\} & \text{if } x \text{ is GOOD} \\ \{\langle 0 \rangle\} & \text{if } x \text{ is not GOOD} \end{cases}$
    Step $\ell > 0$. {Label nodes on level $\ell > 0$ of $\mathcal{H}_n$}
        Assign each level-$\ell$ node $x$ a labeling set $\Lambda(x)$, as follows.
          Substep $\ell.a$ {Assemble the base-labeling vector sets}
            If both $\Lambda(x0)$ and $\Lambda(x1)$ are nonempty, then
            for each pair of length-$\ell$ vectors $\vec{\lambda} \in \Lambda(x0)$ and $\vec{\mu} \in \Lambda(x1)$,
            place the length-$(\ell+1)$ vector $\vec{\nu}$ in $\Lambda(x)$, where
                $\nu_k = \begin{cases} 0 & \text{if } k = \ell \\ \lambda_k + \mu_k & \text{if } k \ne \ell \end{cases}$
            endfor
            else if precisely one of $\Lambda(x0)$ and $\Lambda(x1)$ is nonempty, then
            set $\Lambda(x)$ equal to that nonempty set
            else set $\Lambda(x) = \emptyset$
          Substep $\ell.b$ {Refine the base-labeling vector sets in all ways}
            Repeat the following process for each vector $\vec{\nu} \in \Lambda(x)$, until no new
            vectors are produced:
            for each $k = 0, 1, \ldots, \ell - 1$, in turn
            place the $\lfloor \frac{1}{2}\nu_k \rfloor$ vectors $\{\vec{\nu}^{(1)}, \vec{\nu}^{(2)}, \ldots, \vec{\nu}^{(\lfloor \nu_k/2 \rfloor)}\}$ in $\Lambda(x)$, where each
            vector $\vec{\nu}^{(i)}$ agrees with $\vec{\nu}$ except in positions $k, k+1$, and in those posi-
            tions:
                $\vec{\nu}_k^{(i)} = \vec{\nu}_k - 2i$
                $\vec{\nu}_{k+1}^{(i)} = \vec{\nu}_{k+1} + i$
            endfor
          Substep $\ell.b$ {Honor the congestion bound $C$ in all ways}
            for each vector $\vec{\nu} \in \Lambda(x)$
            if $\sum_{k=0}^{\ell} \nu_k > C$
            then replace $\nu$ in $\Lambda(x)$ by all possible vectors $\vec{\nu}'$ such that
        • $\nu'_k \le \nu_k$ for all $0 \le k \le \ell$
        • $\sum_{k=0}^{\ell} \nu'_k \le C$
    endif
    endfor
Step $n+1$. {Harvest a maximum size tree}
  Use the labeling sets to embed a copy of $\mathcal{G}_m$ in $\mathcal{H}_n$, where $m$ is the largest
  integer such that some vector $\vec{\nu}$ in the labeling set $\Lambda(r)$ of the root of $\mathcal{H}_n$ has
  $\nu_m > 0$. Specifically:
    Substep $n+1.a$ {Embed the root of a copy of $\mathcal{G}_m$}
      Embed the root of $\mathcal{G}_m$ at a node of $\mathcal{H}_n$ with the highest level-number,
      whose labeling set contains a vector $\vec{\nu}$ with $\nu_m > 0$.[12]
    Substep $n+1.b$ {The recursive step}
      Say that we have just embedded a node $y$ of $\mathcal{G}_m$ at level $\ell$ of $\mathcal{H}_n$, and
      say that node $y$ is the root of a subtree $\mathcal{G}_k$ of $\mathcal{G}_m$.
      Then embed the children of $y$ at (not necessarily distinct) nodes of $\mathcal{H}_n$
      with the highest level-number $\le \ell$, that have labeling sets containing
      vectors $\vec{\mu}$ and $\vec{\nu}$ with $\mu_k + \nu_k \ge 2$.   □
```

### 4.2. Timing analysis.

THEOREM 4.1. *Let the leaves of $\mathcal{H}_n$ be labeled* GOOD *or not, in any way, and let $1 < C \le n$. Algorithm* OPTIMAL-HARVEST *finds, in time*

$$\text{TIME}(n) = O(Cn^{3C+2}2^n),$$

*an S-embedding of some $\mathcal{G}_m$ in $\mathcal{H}_n$, having congestion $\le C$ and having optimal harvest among embeddings with congestion $C$.*

*Proof.* The correctness and the quality of the output of Algorithm OPTIMAL-HARVEST being obvious, we concentrate on the timing analysis. The number of vectors in the set $\Lambda(x)$ for a level-$\ell$ node $x$ of $\mathcal{H}_n$ can be no greater than

$$\sum_{k=0}^{C} \binom{\ell+k}{k} = \binom{\ell+C+1}{C} < \ell^C.$$

This bound is verified by analogy with the problem of assigning $\le C$ balls to $\ell+1$ urns. (We are selecting $\le C$ trees, each having one of $\ell+1$ heights, to be carried along to the next step of the algorithm.)

At each level-$\ell$ node $x$, we pair the length-$\ell$ vectors from the label sets of its children $x0$ and $x1$, in all possible ways. We then add each pair together componentwise, and we append a 0 (at the "high-order" end) of each sum vector (to increase its length). The pairing operation leads to fewer than $\ell^{2C}$ pairs of vectors, so the addition step produces fewer than $\ell^{2C}$ sum vectors. Producing a sum vector takes $O(\ell)$ steps. Hence, this part of the processing of node $x$ takes time $O(\ell^{2C+1})$.

Next, we adjust each sum vector in order to produce all possible salvage options. This involves selecting, in all possible ways, one level $k$ such that we can combine paired level-$k$ trees into single level-$(k+1)$ trees. This level can be selected in at most $\ell$ ways, and the combination process requires no more than $O(1)$ operations

[12] Our insistence on the *highest* level-number in Step $n+1$ serves to keep the *dilation* of the S-embedding low, while neither decreasing the harvest nor violating the congestion bound. Note, however, that this insistence could increase the actual congestion incurred by the embedding.

per pair. Since the set $\Lambda(x)$ initially contains fewer than $\ell^{2C}$ vectors, and since (by induction) no entry in any vector in either $\Lambda(x0)$ or $\Lambda(x1)$ can exceed $C$, this part of the processing of node $x$ takes time $O(C\ell^{2C+1})$.

Finally, we "prune" the vectors in $\Lambda(x)$ in order to honor the bound on congestion. Since each vector $\langle \nu_0, \nu_1, \ldots, \nu_\ell \rangle$ at a level-$\ell$ node is in the worst case (before pruning) the sum of two vectors from level-$(\ell-1)$ nodes, it is possible that $\sum_{k=0}^{\ell} \nu_k = 2C$. Hence, when we prune a vector in all possible ways, we may be adjusting it by adding as many as

$$\binom{\ell + C + 1}{C} \leq (const)\ell^C$$

"correction vectors." Each correction is a vector addition requiring $O(\ell)$ steps. Since $\Lambda(x)$ may have grown as large as $O(\ell^{2C+1})$ by this time (due to its expansion during the embedding of new roots), the time required for pruning $\Lambda(x)$ may be as much as (but can be no more than)

$$O(\ell) \cdot O(\ell^C) \cdot O(\ell^{2C+1}) = O(\ell^{3C+2}).$$

Since $\ell \leq n$ in this timing analysis, and since the processing we are analyzing takes place at every node of $\mathcal{H}_n$ (although the processing at lower-level nodes is simpler because they require no pruning), it follows that the time required for this algorithm is

$$\text{TIME}(n) = O(Cn^{3C+2}2^n).$$

This is certainly within the realm of computational feasibility even when $C$ is as big as, say,

$$C = \frac{1}{3}\left(\frac{n}{\log n} - 3\right),$$

which makes TIME$(n)$ quadratic in the size of $\mathcal{H}_n$, and all the more so when $C$ is more modest in size. $\square$

**5. Conclusion.** Our focus on complete *binary* trees throughout this study simplifies notation and calculations. The ideas in our study translate readily to complete trees of any fixed, uniform degree.

**Acknowledgments.** The authors thank Don Coppersmith for suggestions that streamlined a portion of the proof of Theorem 2.2 and an anonymous referee for several suggestions that improved the presentation.

## REFERENCES

[1] A. Agrawal, *Fault-tolerant computing on trees*, manuscript, 1990.

[2] F. S. Annexstein, *Fault tolerance in hypercube-derivative networks*, Proc. 1st ACM Sympos. on Parallel Algorithms and Architectures, 1989, pp. 179–188.

[3] J. L. Bentley and H. T. Kung, *A tree machine for searching problems*, IEEE Internat. Conf. Parallel Processing, 1979, pp. 257–266.

[4] S. Browning, *The tree machine: A highly concurrent computing environment*, Ph.D. thesis, California Institute of Technology, 1980.

[5] R. D. CHAMBERLAIN, *Multiprocessor synchronization network: Design description*, Tech. report WUCCRC-90-12, Washington University, 1990.

[6] ———, *Matrix multiplication on a hypercube architecture augmented with a synchronization network*, manuscript, 1991.

[7] J. HASTAD, F. T. LEIGHTON, AND M. NEWMAN, *Fast computation using faulty hypercubes*, Proc. 21st ACM Symp. on Theory of Computing, 1989, pp. 251–263.

[8] J.-W. HONG, K. MEHLHORN, AND A. L. ROSENBERG, *Cost tradeoffs in graph embeddings*, J. Assoc. Comput. Mach., 30 (1983), pp. 709–728.

[9] C. KAKLAMANIS, A. R. KARLIN, F. T. LEIGHTON, V. MILENKOVIC, P. RAGHAVAN, S. RAO, C. THOMBORSON, AND A. TSANTILAS, *Asymptotically tight bounds for computing with faulty arrays of processors*, Proc. 31st IEEE Sympos. on Foundations of Computer Science, 1990, pp. 285–296.

[10] D. E. KNUTH, *The Art of Computer Programming: Vol.* 3, *Sorting and Searching*, Addison–Wesley, Reading, MA, 1973.

[11] C. E. LEISERSON, *Systolic priority queues*, Proc. 1979 CalTech Conf. on VLSI, 1979, pp. 199–214.

[12] C. MEAD AND L. CONWAY, *Introduction to VLSI Systems*, Addison–Wesley, Reading, MA, 1980.

[13] P. RAGHAVAN, *Robust algorithms for packet routing in a mesh*, Proc. 1st ACM Symp. on Parallel Algorithms and Architectures, 1989, pp. 344–350.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 638–651, November 1995

# UPPER AND LOWER BOUNDS ON CONSTRUCTING ALPHABETIC BINARY TREES*

MARIA KLAWE† AND BRENDAN MUMEY‡

**Abstract.** This paper studies the long-standing open question of whether optimal alphabetic binary trees can be constructed in $o(n \lg n)$ time. We show that a class of techniques for finding optimal alphabetic trees which includes all current methods yielding $O(n \lg n)$-time algorithms are at least as hard as sorting in whatever model of computation is used. We also give $O(n)$-time algorithms for the case where all the input weights are within a constant factor of one another and when they are exponentially separated.

**Key words.** alphabetic binary trees, data structures, algorithms

**AMS subject classifications.** 05C10, 05C35

## 1. Overview.

The problem of finding optimal alphabetic binary trees can be stated as follows: Given a sequence of $n$ positive weights $w_1, \ldots, w_n$, construct a binary tree whose leaves have these weights, such that the tree is optimal with respect to some cost function and also has the property that the weights on the leaves occur in order as the tree is traversed from left to right. A tree that satisfies this last requirement is said to be *alphabetic*. Although more general cost functions can be considered (as is done in [4] and [7]), we concentrate here on the usual function, namely $\sum w_i l_i$, where $l_i$ is the level of the $i$th leaf from the left in the tree. The first $O(n \lg n)$-time solution was given in Hu and Tucker [5] in 1971, following algorithms with higher complexity in [3] and [6]. If we remove the restriction that the tree must be alphabetic, then the problem becomes the well-known problem of building Huffman trees, which is known to have $\Theta(n \lg n)$-time complexity in the comparison model. Modifications of the Hu–Tucker algorithm also running in $O(n \lg n)$ time but with simpler proofs are given in [2] and [4]. The only recent progress on this problem has been made by Ramanan [8], who showed that it is possible to verify that a given alphabetic tree on a sequence of weights is optimal in $O(n)$ time when the weights in the sequence are either within a constant factor or exponentially separated (notions we define precisely later). However, it seems substantially more difficult to actually construct the optimal tree in linear time in the constant factor case.

The next section summarizes current methods and introduces the concepts needed to frame our results. In §3, we introduce a technique, *region processing*, which forms the basis of our linear-time algorithms. We start with a fairly simple $O(n)$-time algorithm for finding the optimal alphabetic tree when the weights are within a factor of two. We also observe that the basic region-processing method solves the case where the input weights are exponentially separated in $O(n)$ time. We generalize this technique in §4 to the case where all the weights are within a constant factor of one another. The generalization depends on solving a new *generalized selection* problem, which may be of interest in its own right. In §5, we give reductions of sorting

* Received by the editors September 1, 1993; accepted for publication (in revised form) October 12, 1994. This research was supported in part by the Natural Sciences and Engineering Council of Canada.

† Department of Computer Science, University of British Columbia, Vancouver, British Columbia V6T 1Z2, Canada.

‡ Department of Computer Science and Engineering, University of Washington, Seattle, Washington 98195.

problems to Hu–Tucker-based algorithms and region-processing methods. This provides $\Omega(n \lg n)$-time lower bounds for Hu–Tucker-based algorithms in the comparison model and indicates that region-processing methods are unlikely to yield an $o(n \lg n)$ algorithm.

## 2. Current methods.

We give a brief description of the Hu–Tucker algorithm to the extent necessary to explain our results. Complete descriptions and explanations can be found in [5], [4], [7]. All Hu–Tucker-based methods begin by building an intermediate tree, called the *lmcp tree*, whose leaves hold the given set of input weights, though not necessarily in the correct order. The levels of the input weights in the lmcp tree are recorded, and this information is used to build an alphabetic tree on the input weights, with each input weight occurring at the same level as in the lmcp tree. Constructing this alphabetic tree can easily be done in $O(n)$ time, as shown in [5]. Since the cost function depends only on the levels of the leaf nodes, the cost of the alphabetic tree is the same as the cost of the lmcp tree. Hu and Tucker proved that the lmcp tree has optimal cost in a class of trees that contains all alphabetic trees, and hence it follows that the alphabetic tree constructed is optimal. We are able to prove that, in the comparison model, constructing the lmcp tree requires $\Omega(n \lg n)$ time in the worst case, but since it suffices to know only the levels of the leaf weights in the lmcp tree and not its full structure, we can improve on the performance of the Hu–Tucker algorithm in a number of cases.

The Hu–Tucker algorithm maintains a *worklist* of weighted nodes in the lmcp tree that have not yet been assigned their sibling and parent. The basic step in the algorithm consists of selecting two nodes from the worklist to be paired off as siblings in the lmcp tree, removing these nodes from the worklist, and inserting a new node (their parent) in the position of the leftmost replaced node with weight equal to the sum of the two removed nodes. Initially the worklist is the list of leaf nodes with the weights $w_1, \ldots, w_n$ in order. Nodes in the worklist are designated either *crossable* or *noncrossable*. Initially all nodes are noncrossable. When any two nodes are paired off, the resulting parent node is designated crossable. Two nodes in the worklist are *compatible* if they are adjacent, or if all the nodes which separate them are crossable. The symbol $v$ will refer to a node in the worklist and $w(v)$ will refer to its weight. The level of a node $v$ in the tree is denoted by $l(v)$. Define an order on the nodes in the worklist by $v_x < v_y$ if $w(v_x) < w(v_y)$ or if $w(v_x) = w(v_y)$ and $v_x$ is to the left of $v_y$ in the list. A pair of compatible nodes $(v_a, v_b)$ is said to be a *local minimum compatible pair* (*lmcp*) if and only if the following two conditions hold:

1. $v_b \leq v_x$ for all nodes $v_x$ compatible with node $v_a$.
2. $v_a \leq v_y$ for all nodes $v_y$ compatible with node $v_b$.

We note that the order relationship given captures the tie-breaking rules of [5] and [2].

The lmcp tree is constructed by repeatedly combining lmcps from the worklist until a single node remains which will be the root of the lmcp tree. This is usually implemented by a stack-based algorithm that starts at the beginning of the worklist and moves a pointer along the worklist until an lmcp is found. After removing the nodes in the lmcp and inserting the new parent node, the pointer is moved back one node and the search for lmcps resumes. To check whether an lmcp has been found, the algorithm compares the smallest node $x$ before the pointer node $y$ that is compatible with $y$ with the smallest node $z$ after $y$ that is compatible with $y$. If $x < z$, the algorithm concludes that $x$ and $y$ form an lmcp; otherwise, it moves the pointer forward one node. The total number of pointer moves is $O(n)$, since $O(n)$ nodes are placed in the worklist in total, and the number of backward moves is bounded by the number of lmcps found, which is also $O(n)$. Hu–Tucker methods take $O(n \lg n)$ time because they maintain information on which node has the minimum weight in intervals of crossable nodes in order to find the nodes $x$ and $z$. Updating this information when an lmcp is found can take $O(\lg n)$ time. In general, the *construction* of the lmcp tree is not unique, since the lmcps may be combined in different orders, but, as proved in [5], the resulting tree is unique. Thus, for any node $v$ in the worklist, we can define the *lmcp partner* of $v$ to be the node that is the sibling of $v$ in the lmcp tree.

**3. Region-based methods.** We present a new approach for finding optimal alphabetic binary trees based on partitioning nodes in the worklist in consecutive runs. Define the *category* of weight $w$ to be $\lfloor \lg (w/w_{\min}) \rfloor$, where $w_{\min}$ is the smallest of the initial weights. A maximal-length sequence of nodes with the same category is called a *region*. In our presentation, we assume that we explicitly compute the category of each weight, since this simplifies the description and explanation of our approach. However, it is possible to avoid the possibly nonunit cost of the lg operations needed to determine the categories explicitly by modifying the algorithm to treat the category numbers of weights as unknowns that can be compared at unit cost. We omit the details of this modification, as they are not crucial to the understanding of the main algorithm.

By keeping a stack of regions and considering only regions whose adjacent regions have higher category, we can restrict most of our attention to the pairings occurring within these regions. We call this *region processing.* This is motivated by the situation where all input weights are within a factor of two. If this is the case, it is easy to determine the leaf levels in the lmcp tree using Theorem 3.1.

THEOREM 3.1. *Given a sequence of $n$ crossable nodes that are within a factor of two, after the first $\lfloor (n+1)/2 \rfloor$ lmcps have been found and combined, the new sequence will consist of $\lfloor n/2 \rfloor$ nodes whose weights are again within a factor of two. Furthermore, if we keep combining lmcps, the resulting lmcp tree will be balanced, with the leaves differing in level by at most one. Specifically, the $2(n - 2^{\lfloor \lg n \rfloor})$ smallest weights will be at level $\lfloor \lg n \rfloor + 1$ and the others will be at level $\lfloor \lg n \rfloor$.*

*Proof.* We note that, since all the nodes are crossable, this reduces the problem to building a Huffman tree, where the result is known. We present a new proof, which provides insight to the actual behavior of the algorithm and motivates our results to follow.

Let the initial sequence of nodes in the worklist be $v_1, \ldots, v_n$ and let $c$ be a real number such that $c \leq w(v_i) < 2c$ for $i = 1$ to $n$. Whenever two nodes form an lmcp and combine, the weight of the new node is greater than $2c$, so it will not be involved in another lmcp until there are less than two nodes smaller than $2c$. When $n$ is odd, after $(n-1)/2$ pairings have occurred, the worklist contains only one node of weight less than $2c$, namely the largest-weight node present in the original sequence. We call this node the *wallflower.* The wallflower forms an lmcp with the smallest-weight newly formed node. When $n$ is even, the largest-weight node present in the original sequence merges with another original node. Thus, regardless of whether $n$ is odd or even, the rightmost (there may be more than one) largest-weight node will merge during the $\lfloor (n+1)/2 \rfloor$th lmcp pairing. At this stage, the worklist will contain exactly $\lfloor n/2 \rfloor$ nodes, none of which are original nodes, and their weights will be within a factor of two, as we show below.

This is obvious if $n$ is even, so suppose $n$ is odd, and let $v$ be the node with the smallest weight, $w(v) = w(v_i) + w(v_j)$, among the first $(n-1)/2$ newly formed nodes.

Clearly, the rest of the first $(n-1)/2$ newly formed nodes have weights less than $2w(v)$. Let $v_k$ be the wallflower. The next node formed is the parent of $v$ and $v_k$ and has weight $w(v_k)+w(v_i)+w(v_j)$. Now, since the original weight sequence was within a factor of two, $w(v_k) < w(v_i)+w(v_j) = w(v)$, so $w(v_k)+w(v_i)+w(v_j) < 2w(v)$, which completes the proof. One further observation that will be important is that the weight of the parent of the wallflower is strictly greater than the weight of the other $(n-1)/2$ nodes in the current worklist.

Let us call the pairings up to this point a *phase* of the algorithm and consider how the phase affects the levels of the leaves in the lmcp tree. Obviously, the phase contributes one to the level of each leaf in the lmcp tree if $n$ is even. When $n$ is odd, this is true for all the leaves except for the two whose parent was paired with the wallflower. These two, which we call the wallflower's stepchildren, have had their level increase by exactly two. Since the wallflower's parent has the unique largest weight in the worklist at the end of the phase, at the end of each later phase this node's ancestor always has the unique largest weight in the worklist. Thus each later phase contributes exactly one to the level of the wallflower's stepchildren. Applying this argument to the stepchildren of wallflowers from later phases proves that the level of any two leaves in the lmcp tree differs by at most one. Since the lmcp tree has optimal cost, the smallest-weight original nodes must be at the bottom level, i.e., the largest-numbered level. Thus for some integer $x$, we have the $2x$ smallest-weight original nodes on level $\lfloor \lg n \rfloor + 1$ and the remaining $n-2x$ original nodes on level $\lfloor \lg n \rfloor$. We require $x + n - 2x = 2^{\lfloor \lg n \rfloor}$, so $x = n - 2^{\lfloor \lg n \rfloor}$. □

Based on this theorem, it is easy to give a simple linear-time algorithm for finding an optimal alphabetic binary tree on a sequence of input weights which differ at most by a factor of two. (Garcia and Wachs also give a linear-time method for this case in [2].) In point form, the algorithm for finding the levels of the leaves in the alphabetic tree is:

1. Initialize the worklist to contain the original input sequence. Note that all nodes are noncrossable.
2. Use a stack-based method to find lmcps and pair them off, removing each pair of nodes from the worklist and placing the parent in a temporary list but not in the worklist. These newly formed nodes can be left out of the worklist because their weights are greater than any of the original weights, and hence need not be considered in the search for lmcps. This process continues until there are zero or one nodes left in the worklist, and as discussed in the remarks on stack-based algorithms in §2, requires only $O(n)$ time because of the absence of crossable nodes in the worklist. If a single node $x$ remains ($n$ is odd and $x$ is the wallflower), scan through the temporary list of newly formed crossable nodes to find the smallest node $y$. Pair $x$ with $y$ and replace $y$ in the temporary list by its parent.
3. At this stage we have $m = \lfloor n/2 \rfloor$ crossable nodes in the temporary list. Moreover, the new nodes are still within a factor of two, by the same argument as in the proof of the preceding theorem.
4. We can now, by the preceding theorem, directly find the levels of every leaf in the lmcp tree for the remaining $m$ crossable nodes in $O(n)$ time, using a linear-time selection algorithm [1] to find the $2(m - 2^{\lfloor \lg m \rfloor})$th weight in the temporary list. This node and nodes with smaller weights have level $\lfloor \lg m \rfloor + 1$, and the remaining nodes are assigned level $\lfloor \lg m \rfloor$. Given this, it is trivial to compute the levels of the nodes in the original input sequence in

an additional $O(n)$ time.

5. With knowledge of the leaf levels, we can construct the optimal alphabetic tree for the input sequence in $O(n)$ time, using the technique in [5].

A similar technique can be applied to predict how nodes in a region $R$ with lowest category number combine to form nodes in a region with the next category number. Notice that when the number of nodes in $R$ is odd, its wallflower will pair with the smallest-weight node in the set consisting of the lmcps formed out of $R$ and the compatible nodes from the two regions adjacent to $R$. When the gap in category number between adjacent regions is large enough, this method yields faster performance than the Hu–Tucker algorithm. The complete algorithm is described in [7]. Its basic idea is to maintain a stack of the current regions in the worklist and process the region at the top of the stack if its adjacent regions have greater category. If not, the stack pointer is advanced. The cost of processing a region of size $r$ is $O(r \lg r)$. Since processing a region yields a new region of half the size, it is easy to verify that this method has $O(n \lg n)$ running time. If the input weights $\{w_i\}$ are exponentially separated, i.e., if there is a constant $C$ such that for all integers $k$, $|\{i : \lfloor \lg w_i \rfloor = k\}| < C$, then it is also easy to verify that this method yields an $O(n)$-time algorithm, since each region can be processed in constant time as the size is bounded by $2C$. The ideas in Theorem 3.1 can also be used to reduce the cost of processing a region of size $r$ to below $O(r \lg r)$ when the difference in category numbers is great enough, which may be useful in implementations. Details are given in [7].

**4. The constant factor case.** We now describe the linear-time algorithm for weights within a constant factor, i.e., such that $\max\{w_i/w_j\} < \sigma$ for some constant $\sigma$. As before, it suffices to determine the levels of the leaf nodes in the lmcp tree. We use a region-based method to process the weights region by region in increasing order by category number until we are left with a single region of crossable nodes. We then apply Theorem 3.1 to determine the lmcp tree levels of the nodes in this final region and work backwards to find the lmcp tree levels of the original weights. In order to achieve the linear-time bound, when processing a region, we cannot afford to determine which nodes pair together in lmcps or the weights of the lmcps formed. Instead, we work with coarser information about the structure of the lmcp tree. An interval of nodes in a region's worklist is *lmcp-closed* if the lmcp partner of each node in the interval is also in the interval. Our algorithm works by partitioning the region's worklist into lmcp-closed intervals and replacing each lmcp-closed interval by a *node group* representing the lmcps formed out of that interval. From the definition of the lmcp, it is easy to see that moving an interval of larger crossable nodes to the right of an interval of smaller crossable nodes or pushing a larger crossable node to the right of a smaller noncrossable node does not affect the construction of the lmcp tree. Our algorithm uses such rearrangements of the worklist in finding the partition into lmcp-closed intervals.

The worklist thus is now an ordered list of node groups in which each noncrossable node appears as a singleton node group but intervals of crossable nodes within a region may appear in groups of arbitrary size. A set of nodes in the worklist is *realizable* if it is the union of a set of node groups in the worklist. The algorithm performs certain types of selection operations on realizable sets of nodes in the worklist. For example, when the worklist consists of crossable nodes whose weights are within a factor of two, the algorithm determines the smallest $k$ of these nodes in order to apply Theorem 3.1.

Since we will generally not have an explicit list of the weights of the nodes in the realizable set on which we wish to perform selection, we will introduce the concept of *a coarse-selection system*, namely a structure for (nonexplicitly) representing a set of elements, together with a particular set of selection operations that can be performed efficiently on the set. We will then show that each realizable set has a coarse-selection system. Performing a selection operation on a realizable set may require that some of the node groups in the realizable set be refined, in order that the result be in the form of realizable sets. For example, suppose $N$ is a realizable set of nodes in the worklist. Determining the largest (smallest) node $v$ in $N$ requires replacing the node group containing $v$ by a node group list in which $v$ is a singleton node group, unless $v$ is already a singleton. Similarly, determining the $k$ smallest nodes in $N$ requires a node-group list in which the desired set is the union of a set of node-groups in the refined list. Thus we will ensure that the selection operations we provide for realizable sets determine the appropriate refinements. We now define coarse-selection systems.

DEFINITION 4.1. *For any $\Delta \geq 1$, we say a (multi) set $S$ has a $\Delta$* coarse-selection system *if:*

1. *$\forall \alpha \in [0,1]$, in $\Delta|S|$ time we can produce two disjoint sets $S_\alpha^-$ and $S_\alpha^+$, each with $\Delta$ coarse-selection systems such that $S = S_\alpha^- \cup S_\alpha^+$, $\forall x \in S_\alpha^-$ and $\forall y \in S_\alpha^+$, $x \leq y$, and $|S_\alpha^-| = \lfloor \alpha |S| \rfloor$. (We call this an $\alpha$-*partition *of $S$.)*
2. *$\forall x \geq 0$, in $\Delta|S|$ time, we can compute the rank of $x$ in $S$, denoted by $r_S(x)$, and produce two sets $S^{\leq x}$ and $S^{>x}$, each with $\Delta$ coarse-selection systems such that $S^{\leq x} = \{y \in S : y \leq x\}$ and $S^{>x} = \{y \in S : y > x\}$. (The rank of $x$ in $S$ is the number of elements in $S$ less than or equal to $x$.)*
3. *In $\Delta|S|$ time we can compute $|S|$.*
4. *If $|S| = 1$ we can determine the unique element of $S$ explicitly in $\Delta$ time.*

In addition, when interpreted in the context of node-group lists, we require that the sets $S_\alpha^-, S_\alpha^+, S^{\leq x}, S^{>x}$ be realizable. Note that the definition of a coarse-selection system implies that, given a $\Delta$ coarse-selection system for $S$, we can explicitly determine the largest (smallest) element of $S$ in $2\Delta|S|$ time. We use the term *layer $h$* for the regions in the worklist with category number $h$ and process the regions in the worklist a layer at a time beginning with the smallest numbered layer. Processing layer $h$ consists of creating node-group lists representing the new nodes formed in layer $h+1$. Consider the question of creating a node-group list representing the new nodes, $T$, formed from a single region $R$ of $r$ nodes. If $r$ is even, because the regions adjacent to $R$ in the worklist have higher category numbers, $R$ is lmcp-closed and the node-group list for $T$ is a single node group. If $r$ is odd, then the only node of $R$ whose lmcp partner is not in $R$ is its wallflower $z$. It is straightforward to prove that $z$ is the largest node in the subset $\{y \in R : y$ is crossable or $y$ is noncrossable and is in an odd-numbered position from an end of $R\}$. Note that this subset is realizable and that $z$ can be identified by coarse selection. Thus we create a node group, $g_l$, representing the lmcps formed from the nodes on the left of $z$, and another one, $g_r$, for those from the right, respectively. To determine the lmcp partner of $z$, we need to know the smallest node $v$ in $g_l \cup g_r$, which again is realizable. We complete the processing of $z$ by comparing $v$ with the smallest compatible nodes on either side of $g_l, g_r$ in the worklist (found using selection on realizable sets), and we replace $z$ and its partner by a singleton node group representing this lmcp. This singleton node group may be in layer $h+2$, in which case we place it as far to the right as possible (in front of the first node to the right that is in layer $h+2$ or higher). The remaining challenge is to construct the coarse-selection systems for realizable sets, which is done by induction on layer number.

Our inductive hypothesis will be that, for any node-group list representing the nodes in a region of layer $h$, and any set of nodes, $A$, that is realizable with respect to that node-group list, there is a coarse-selection system for $A$. The base case is covered by the usual linear-time selection algorithm, since all nodes in the bottom layer are noncrossable. Thus the only possible node-group list for the bottom layer is the standard list of single nodes, so all the weights of the nodes in the list are known explicitly. A key tool is the construction of a coarse-selection system for the union of sets with coarse-selection systems. This is provided by the following theorem.

THEOREM 4.2. *Let $A = \cup_{i=1}^{n} A_i$, where the $A_i$ are disjoint and nonempty and each $A_i$ has a $\Delta$ coarse-selection system. Then $A$ has a $36\Delta$ coarse-selection system.*

*Proof.* Let $x$ be any value. We can compute the rank of $x$ in $A$ easily, since $r_A(x) = \sum_{i=1}^{n} r_{A_i}(x)$. Moreover, $A^{\leq x} = \cup_{i=1}^{n} A_i^{\leq x}$ and $A^{>x} = \cup_{i=1}^{n} A_i^{>x}$. The time cost for this is the cost of finding $r_A(x)$ plus the cost of constructing the $A_i^{\leq x}$ and $A_i^{>x}$. This is $\sum_{i=1}^{n} \Delta|A_i| + \sum_{i=1}^{n} \Delta|A_i| = 2\Delta|A|$.

For $\alpha \in [0,1]$, we construct $A_\alpha^-$ and $A_\alpha^+$ as follows. For each $i$, compute $A_{i_{1/2}}^-$, $A_{i_{1/2}}^+$, and $m_i = \min A_{i_{1/2}}^+$. This can all be done in $2\Delta|A|$ time. We now compute the median $m$ of the multiset $M = \cup_{i=1}^{n} M_i$, where $M_i$ contains exactly $|A_i|$ copies of $m_i$, by using a standard selection algorithm. This can be done in $6|A|$ time using the selection algorithm of Blum et al. [1]. Now compute $r_A(m)$ as above, in $\Delta|A|$ time. If $r_A(m) = \lfloor \alpha|A| \rfloor$, we are done, as we can take $A_\alpha^- = A^{\leq m}$ and $A_\alpha^+ = A^{>m}$. If not, we may assume $r_A(m) > \lfloor \alpha|A| \rfloor$, since a symmetric argument handles the other case. Let $J = \{i : m_i \geq m\}$, let $B = A - \cup_{i \in J} A_{i_{1/2}}^+$, and note that every element in $A - B$ is at least $m$. If $|B| < \lfloor \alpha|A| \rfloor$, since $r_A(m) > \lfloor \alpha|A| \rfloor$, there must be at least $\lfloor \alpha|A| \rfloor - |B|$ elements in $A - B$ that equal $m$. Thus, it suffices to identify a subset $D$ of these elements, with $|D| = \lfloor \alpha|A| \rfloor - |B|$, and take $A_\alpha^- = B \cup D$. To find $D$, we first find $(A_{i_{1/2}}^+)^{\leq m}$ for each $i$ in $J$. Every element in $\cup_{i \in J} (A_{i_{1/2}}^+)^{\leq m}$ must equal $m$, and thus it suffices to take $D$ to be any subset of $\cup_{i \in J} (A_{i_{1/2}}^+)^{\leq m}$ of the appropriate size. Such a subset can easily be obtained by taking each $(A_{i_{1/2}}^+)^{\leq m}$ until adding another set will result in more than $\lfloor \alpha|A| \rfloor - |B|$. At this point, coarse selection can be used on this $(A_{i_{1/2}}^+)^{\leq m}$ to obtain a subset that will bring the total number of elements to exactly $\lfloor \alpha|A| \rfloor - |B|$. Thus, in this case, we will have obtained $A_\alpha^-$ and $A_\alpha^+$ in at most $(6 + 5\Delta)|A|$ time. If $|B| \geq \lfloor \alpha|A| \rfloor$, we may take $(A - B) \subset A_\alpha^+$, since every element in $A - B$ is at least $m$. Note that $\sum_{i \in J} |A_i| \geq \frac{1}{2}|A|$ by the definition of $M$. Hence $|A - B| = |\cup_{i \in J} A_{i_{1/2}}^+| \geq \frac{1}{4}|A|$, and so we reduce the problem to finding a $\beta$-partition in $B$, where $\beta = \alpha \frac{|A|}{|B|}$. We set $A_\alpha^- = B_\beta^-$ and $A_\alpha^+ = \cup_{i \in J} A_{i_{1/2}}^+ \cup B_\beta^+$. In this case, we reduce the problem to one at most $3/4$ of the original size in $(6 + 3\Delta)|A|$ time. Since $B$ is a union of sets with $\Delta$ coarse-selection systems, an easy inductive argument on the size of $A$ shows that we can produce $A_\alpha^-$ and $A_\alpha^+$ in $\frac{1}{1-3/4}(6 + 3\Delta)|A| \leq 36\Delta|A|$ time.

The fact that $A^{\leq x}$, $A^{>x}$, $A_\alpha^-$, and $A_\alpha^+$ each have $36\Delta$ coarse-selection systems again follows easily by induction on $|A|$ since they are unions of sets with $\Delta$ coarse-selection systems. $\square$

We are now ready to begin the inductive proof of the existence of coarse-selection systems. We assume that, given any node-group list representing the nodes in a region of layer $h$ and a set of nodes that is realizable with respect to that node-group list, the set has a $\Delta$ coarse-selection system. Given this assumption, we show how to construct a $D\Delta$ coarse-selection system for any set $S$ of nodes in a region $X$ of layer $h+1$ such that $S$ is realizable with respect to a node-group list for $X$. The value of $D$ is a constant independent of $h$. By the definition of a node-group list, it is clear that any node-group list for $X$ inherently provides node-group lists for the regions in layer $h$ that contain the children of nodes in $X$. By the preceding theorem, we may assume that there are no singleton node groups in the representation of $S$, since otherwise we can use the usual linear-time selection algorithm for the set $S^*$ of nodes in $S$ occurring as singletons, and we can use the selection systems for $S^*$ and $S - S^*$ to get a selection system for $S$. This assumption says that there is a set $\{R_i\}$ of disjoint lmcp-closed realizable intervals in layer $h$ such that $S$ is the lmcps formed from $V = \cup_i R_i$. We first show how to find the smallest-weight node in $S$ by proving that, in $O(\Delta|S|)$ time, we can reduce the problem to finding the smallest-weight node in a realizable subset $S'$ of $S$, where $|S'| \leq |S|/2$. This reduction process may involve refining some node-group lists for regions in layer $h$, and such refinements increase the number of realizable sets. This is why our inductive assumption ensures the existence of $\Delta$ coarse-selection systems for realizable sets, independent of which node-group list is used in the definition of realizability. Finding the smallest node is a special case of finding an $\alpha$-partition, but the algorithm is slightly simpler. Moreover, since it is a subroutine used in finding general $\alpha$-partitions, presenting it first clarifies the exposition.

The set $V$ is realizable, so, in $\Delta|V|$ time, we can find the 1/2-partition $V = V_{1/2}^- \cup V_{1/2}^+$. For each $R_i$, we write $R_i^- = R_i \cap V_{1/2}^-$ and $R_i^+ = R_i \cap V_{1/2}^+$. We assume, by reordering if necessary, that for each interval $C$ of crossable nodes in $R_i$, we have $C \cap R_i^-$ preceding $C \cap R_i^+$.

We now describe an algorithm which we will run on $R_i$ to partition its nodes into three lmcp-closed sets, $R_i = R_i^{--} \cup R_i^{++} \cup R_i^{-+}$, according to whether the node and its lmcp partner are in the same class in the partition $R_i = R_i^- \cup R_i^+$. The set $R_i^{--}$ is the set of nodes $x$ such that both $x$ and its lmcp partner, $p(x)$, are in $R_i^-$. The sets $R_i^{++}$ and $R_i^{-+}$ are defined analogously. For each node $x$ in $R_i^{-+}$ (the set in which $x$ and $p(x)$ are in different classes), the algorithm explicitly determines $x$ and $p(x)$ and hence can create a singleton node group for the lmcp of $x$ and $p(x)$.

We use the terms $-$*interval* [$+$*interval*] to refer to a maximal interval of nodes in $R_i$ which lies entirely in $R_i^-$ [$R_i^+$]. Obviously, $R_i$ is an alternating sequence of –intervals and +intervals. Also, –intervals and +intervals are realizable sets. We first note that, if any two consecutive –intervals are separated by a +interval that does not contain noncrossable nodes, we may push the +interval to the right of the right-hand –interval without affecting the formation of lmcps. Thus, in linear time, we can rearrange each $R_i$ so that there is at least one noncrossable node in each +interval, except for possibly one on the right end of $R_i$. If the number of nodes in a –interval, $I$, is even, then for each $x \in I$ we have $p(x) \in I$. This follows from the fact that $S$ is realizable and that each node group of $S$ represents the lmcps formed out of a consecutive interval in layer $h$. Next, for each –interval, $I$, with an odd number of nodes, we use the $\Delta$ coarse-selection system to find its *local wallflower*, i.e., the largest node in $I$ which either is crossable or is noncrossable and in an odd-numbered position relative to $I$. Note that each local wallflower $x$ is now represented by a singleton node group, and we know its weight. Let $I'$ be the set resulting from removing the local wallflower from $I$, if it has one. It is not hard to prove that, for each $x \in I'$, we have $p(x) \in I'$, so we set $R_i^{--}$ to be the union of the $I'$. We now remove the node groups representing the nodes in $R_i^{--}$ from the node-group list of $R_i$.

We will process the list of node groups that remain in $O(\Delta|R_i|)$ time to determine the lmcp partner of each local wallflower and define $R_i^{-+}$ as the set of local wallflowers (i.e., the nodes in $R_i^-$ which still remain in the list) together with their lmcp partners. $R_i^{++}$ is $R_i - (R_i^{--} \cup R_i^{-+})$.

In order to determine the lmcp partner of each local wallflower, we first identify, for each end of a +interval, the smallest-weight node in the +interval compatible from that end of the interval. For each +interval that contains at most one noncrossable node, we also identify its smallest-weight crossable node, if one exists. This can be done in $O(\Delta|R_i|)$ time using coarse-selection systems. We now use the standard stack-based method described at the end of §2, where the worklist consists of the local wallflowers surrounded by the identified neighboring nodes from the +intervals, in order. The stack pointer is initially placed on the leftmost local wallflower. We stop when all the local wallflowers have been paired off, i.e., when their lmcp partners have been determined. It is straightforward to check that, upon removal of an lmcp involving a local wallflower, $x$, the necessary information on the affected +intervals can be updated in constant time, and this guarantees the linear-time bound.

For $j = --, ++, -+$, let $V^j$ be the union of the nodes in the $R^j$, and let $S^j$ be the nodes formed from $V^j$. We note that all the nodes in $S^{--}$ are less than or equal to the nodes in $S^{++}$, though it is possible that there are nodes in $S^{-+}$ that are smaller than some in $S^{--}$ and others in $S^{-+}$ that are greater than some in $S^{++}$. In addition, we know that both $|S^{--}|$ and $|S^{++}|$ are less than $|S|/2$ since $|S^{--}| = |S^{++}|$. We also know all the nodes (and their weights) explicitly in $V^{-+}$ and hence can find the smallest node in $S^{-+}$ in $O(|S^{-+}|)$ time. Thus, it suffices to find the smallest node in $S^{--}$, and taking $S' = S^{--}$ completes the proof. The analogous technique works to find the largest node in $S$ or the rank of a node $x$ and the sets $S^{\leq x}$ and $S^{>x}$ in $O(\Delta|S|)$ time. We will call the process of determining the sets $S^{--}, S^{-+}, S^{++}$ *sifting*.

Now suppose we wish to find $S_\alpha^-$ and $S_\alpha^+$ for some $\alpha \in [0, 1]$. We assume $\alpha \leq 1/2$, since the case $\alpha > 1/2$ is analogous. Let $\beta = \max(\alpha, 3/7)$. We repeat the sifting process as before, except that we find the $\beta$-partition $V = V_\beta^- \cup V_\beta^+$. For each set $R_i$, we now set $R_i^- = R_i \cap V_\beta^-$ and $R_i^+ = R \cap V_\beta^+$ and define the sets $R^j, V^j, S^j$ as before for $j = --, -+, ++$.

Let $\gamma = |V^{--}|/|V| = |S^{--}|/|S|$. For the sake of simplicity, we ignore floors and ceilings for the moment. It is not hard to see that we have $|V^{-+}| = 2(\beta - \gamma)|V|$ and $|V^{++}| = (\gamma + 1 - 2\beta)|V|$. Thus $|S^{-+}| = 2(\beta - \gamma)|S|$ and $|S^{++}| = (\gamma + 1 - 2\beta)|S|$. Using the algorithm described above, we find, in $O(\Delta|S|)$ time, the largest node $s^-$ in $S^{--}$ and the smallest node $s^+$ in $S^{++}$, respectively. Let $S^{-+} = S_1 \cup S_2 \cup S_3$, where $S_1$ contains the nodes in $S^{-+}$ less than or equal to $s^-$ and $S_3$ contains the nodes in $S^{-+}$ greater than or equal to $s^+$. We can find these sets using the Blum et al. [1] linear-time selection algorithm because the nodes (and their associated weights) in $S^{-+}$ are known explicitly.

Let $A = S^{--} \cup S_1$, $\delta = |A|$, and $Z = S^{-+}$. If $|Z| \geq \alpha|S|$, we set $\rho = \alpha|S|/|Z|$, and using the standard linear-time selection algorithm, we find a $\rho$-partition $Z = Z_\rho^- \cup Z_\rho^+$. We now prove that there is always one of the sets $A, S - A, Z_\rho^+$ whose nodes we can remove from $S$, because we can assume that they are in one of the sets of the $\alpha$-partition. Moreover, we prove that the set we remove contains at least 1/7th of the nodes in $S$.

First, note that each node in $S - A$ has weight at least as large as any node in $A$, so if $|A| \geq \alpha|S|$, then we place the nodes in $S - A$ in $S_\alpha^+$ and reduce the problem to finding the $\alpha(|S|/|A|)$-partition of $A$. Symmetrically, if $|A| \leq \alpha|S|$, we place the nodes in $A$ in

$S^-_\alpha$ and reduce the problem to finding the $(1-\alpha)(|S|/(|S|-|A|))$-partition of $S-A$. A similar argument applies to removing the nodes in $Z^+_\rho$ when we have $|Z| \geq \alpha|S|$, and we reduce the problem to finding the $\alpha(|S|/(|S|-|Z^+_\rho|))$-partition of $S - Z^+_\rho$. We now consider the sizes of the sets involved. If $\gamma \leq \beta/3$, we have $|Z| \geq 4\beta|S|/3 \geq \alpha|S|$ and $|Z^+_\rho| = (2(\beta-\gamma)-\alpha)|S| \geq (\beta - 2\gamma)|S| \geq \beta|S|/3 \geq |S|/7$, since $\beta \geq \alpha$ and $\beta \geq 3/7$. Now suppose $\gamma \geq \beta/3$. We have $\gamma \geq 1/7$, so $|A| \geq |S^{--}| \geq |S|/7$ and $|S-A| \geq |S^{++}| \geq |S^{--}|$. Thus, in all cases, there is a set of size at least $|S|/7$ that can be removed, and we have reduced the problem to a realizable set of size at most $6|S|/7$ in $O(\Delta|S|)$ time.

It is easy to use the above ideas to compute, in $O(\Delta|S|)$ time, the rank in $S$ of any node $x$, as well as find $S^{\leq x}$ and $S^{>x}$. Moreover, computing $|S|$ is trivial from the node-group list for $S$. Combining these observations yields a $D\Delta$ coarse-selection system for any realizable set in layer $h+1$, where the constant $D$ is independent of $h$. It is interesting to note that the largest portion of $D$ is a result of applying Theorem 4.2 to merge the selection system for the singleton node groups with the selection system for the larger node groups.

The arguments above yield an $O(D^h)$ coarse-selection system for realizable sets in layer $h$. By dividing all the original weights by the smallest weight, we may assume that they lie between 1 and $\sigma$, and hence we must process at most $\lceil \lg \sigma + 1 \rceil$ layers before reaching the point where the worklist contains only crossable nodes. At this point the weights are within a factor of two, we have an $O(D^{\lg \sigma}|S|) = O(n)$ coarse-selection system, and we can apply Theorem 3.1 to determine the levels of these nodes, which we then use to determine the levels of the original weights.

## 5. Hardness results.

We begin with a simple hardness result that shows that constructing the intermediate lmcp tree produced by Hu–Tucker-based algorithms in any model of computation is at least as difficult as sorting in that model. We also give a more complicated reduction from sorting to any algorithm which computes enough partial information about the lmcp tree. This partial information is something we expect any region-based method must compute.

### 5.1. Finding the lmcp tree.

We will need the following simple lemma, whose proof we omit since it follows immediately from the observations made at the beginning of the proof of Theorem 3.1.

LEMMA 5.1. *Let $x_1, x_2, \ldots, x_n$ be distinct real numbers drawn from $[2,4)$. Let $y_i = \frac{1}{2}x_{\lceil i/2 \rceil}$, for $i = 1 \ldots 2n$. If $(y_1, \ldots, y_{2n})$ is given as input to any lmcp-finding algorithm, the set of the first $n$ lmcps found, disregarding order, will be*

$$\{(y_1, y_2), (y_3, y_4), \ldots, (y_{2n-1}, y_{2n})\}.$$

THEOREM 5.2. *We can reduce sorting sequences of size $n$ to finding the lmcp tree in $O(n)$ time.*

*Proof.* Assume $n$ is even. Let $x_1, x_2, \ldots, x_n$ be drawn from $[2,4)$. Define the $y_i$ as above and consider the behavior of some lmcp-combining algorithm on the input sequence $y_1, \ldots, y_{2n}$. According to Lemma 5.1, after $n$ lmcps have been combined, there will be $n$ crossable nodes in the worklist with the weights $x_1, \ldots, x_n$. The only lmcp in the list is the smallest pair of nodes in $\{x_1, \ldots, x_n\}$ that combine to form a new node with weight at least 4. The next lmcp will be the second smallest pair of nodes from $\{x_1, \ldots, x_n\}$ and so on. Hence the next $n/2$ lmcps found sort $\{x_1, \ldots, x_n\}$ by pairs. Moreover, the fully sorted order of the $x_i$ can be recovered from the lmcp tree (independent of how it was constructed) by searching the tree depth first and always searching the least-weight subtree first, since the nodes corresponding to $\{x_1, \ldots, x_n\}$ will be encountered in sorted order. This shows that sorting can be reduced to finding the lmcp tree in $O(n)$ time. $\square$

**5.2. Region-based methods.** In light of the linear-time algorithm for the constant factor case, it is natural to look for an $o(n \lg n)$-time method based on region processing. As before, we would hope to avoid determining all the lmcps but still determine the leaf levels in the lmcp tree. The wallflower is the difficult case to handle because it seems necessary to know explicitly which node it pairs with (as this increases the level of the leaves of this node by one). In particular, the wallflower may pair with the lmcp formed from the two smallest nodes in its region, and so it seems necessary that this information be easy to find for every region considered. We will say that an alphabetic tree-finding algorithm is *region based* if, from the information it computes, it is possible in $O(n)$ time to determine, for the smallest two nodes at each level, the set of leaves in the subtree of the lmcp tree rooted by each of these nodes. Note that this information is easy to compute if regions in the worklist are processed by increasing category order and the smallest two nodes are explicitly found in every region processed. This is because the smallest two nodes at each level in the lmcp tree are the smallest two nodes for some region, and we can easily keep track of the eventual level of the pair of smallest nodes for every region and pick the smallest pair at every level. The following theorem provides an $\Omega(n \lg n)$ lower bound in all models of computation for which an information-theoretic argument can be applied.

THEOREM 5.3. *Any region-based algorithm for finding alphabetic trees can be used, with $O(n)$ additional work, to sort sequences possessing a particular structure. Moreover, the number of distinct orderings among sequences with this structure is $2^{\Omega(n \lg n)}$.*

*Proof.* We show the existence of a sufficently large class of input sequences, such that for any sequence in the class, a region-based algorithm determines the structure of the lmcp tree. The proof is completed by showing that, for these sequences, the sorted order can be determined from the lmcp tree in $O(n)$ time.

The input sequences we consider consist of approximately $\sqrt{n}$ regions, each containing about $\sqrt{n}$ nodes and such that the category of a given region is one more than the region on its left. We assume $n = k^2 + 3k + 4$, where $k$ is a positive integer. The first region will contain weights with values in $[1, 2)$, the next $[2, 4)$, then $[4, 8)$, etc. Denote the $j$th value in the $i$th region by $y_{i,j}$. The first region will have $4k + 4$ weights; the remaining have $2(k-1), 2(k-2), 2(k-3), \ldots, 2$ weights, respectively. Note that $4k+4+2(k-1)+2(k-2)+\cdots+2 = k^2+3k+4$. Let $x_1 < x_2 < \cdots < x_{2^{k+1}}$ be real numbers in $[2, 4)$. The values for the $\{y_{i,j}\}$ will be determined from the $\{x_i\}$. As the proof depends on the crossability of nodes, the values come in pairs so that the leaf nodes initially combine in pairs (this will be proved in Lemma 5.4).

Consider the following recursively generated binary tree built from the $\{x_i\}$. If internal nodes are assigned the sum of the weights of their children, then it has the property that the left child of any node is always less than the right.

Figure 1 shows the tree built for $k = 3$. The tree built for $k = 2$ is the subtree rooted at the left child of the root. The tree for $k = 4$ has this tree as the left child of its root, with the right child of the root consisting of an arm with leaf weights $x_{17} + \cdots + x_{24}, x_{25} + \cdots + x_{28}, x_{29} + x_{30}, x_{31}, x_{32}$ from left to right.

The purpose of this tree is to assign values to the $\{y_{i,j}\}$. Randomly distribute consecutive pairs $(y_{1,j}, y_{1,j+1})$, $j = 1, 3, \ldots, 4k+3$, among the $2k+2$ lowest terminal leaves in this tree. For $j = 1, \ldots, 4k+4$, let $y_{1,j}$ be half the weight of the leaf that

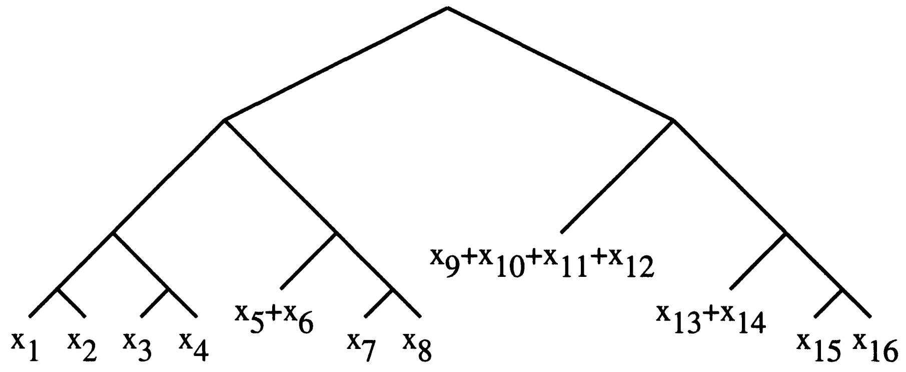


FIG. 1. *Tree generated from* $\{x_i\}$.

it is associated with. Then assign values to consecutive pairs of the $2(k-1)\{y_{2,j}\}$ by distributing them among the next lowest terminal leaves and so on. This new tree is called the *ordering tree* and is shown in Fig. 2. It records how the weights were assigned, and also their sorted order.

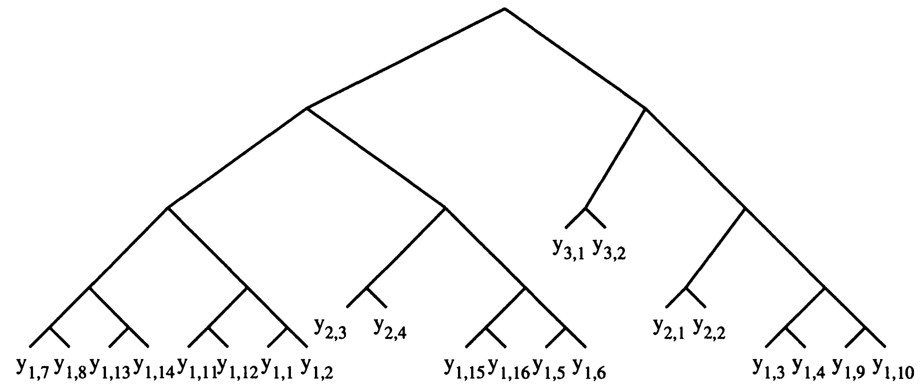


FIG. 2. *The ordering tree.*

The input weight list is as follows, with regions distinguished by height.

$$\begin{array}{lll} & & y_{3,1}, y_{3,2} \\ & y_{2,1}, y_{2,2}, y_{2,3}, y_{2,4} & \\ y_{1,1}, y_{1,2}, \cdots, y_{1,15}, y_{1,16} & & \end{array}$$

With an additional $O(n)$ time, we can determine the smallest two nodes at each level of the lmcp tree from the information computed by any region-based algorithm. To finish the proof, we first need a lemma.

LEMMA 5.4. *If the children in the lmcp tree are ordered according to weight, then the lmcp tree is isomorphic to the ordering tree.*

*Proof.* We may assume that we begin by combining all the lmcps in the lowest (largest-level) region. From Lemma 5.1 we know that since the weights come in consecutive pairs of the same weight, these pairs will eventually form lmcps and combine, in agreement with the ordering tree. At this stage, the lowest region in the worklist consists of crossable nodes interspersed with some noncrossable ones, which again come in pairs. It is easily seen from the ordering tree that there is always an even number of crossable nodes smaller than the consecutive pairs of noncrossable nodes in the lowest region. Thus, we know that these crossable nodes will pair off first, and then the consecutive pairs of noncrossable leaf nodes will pair off as is shown in the ordering tree. It is clear from the ordering tree that this process continues and the lmcp tree, with every internal node's children ordered by increasing weight, is isomorphic to the ordering tree. Lemma 5.4 is proven. $\square$

Consider the children of the root of the lmcp tree. By assumption, we know their weights and which leaves each child roots. By the lemma, the smallest node roots all the leaves on the left branch of the ordering tree, while the second smallest node will root all the leaves on the right branch. In time proportional to the number of leaves we find, we can traverse the right branch of our tree and find all the leaves and hence weights $\{y_{i,j}\}$ that are on the right branch of the ordering tree. Since there are only a constant number of leaves per level, we can afford to sort each level, and hence we begin sorting each of the regions in the initial input list. We now use this idea recursively on the subtree rooted at the smallest node of the root. This lets us find all the leaves in the right branch of the left branch from the root in the ordering tree. Again, we may sort the weights present at each level and append them to the beginning of the sorted region lists created previously. This will take time proportional to the number of nodes in this branch. By repeating this process, we will completely determine every input weight's location in the ordering tree, and, from this information, we can produce sorted lists of the weights in each region in the input. All this takes only $O(n)$ time to do.

The input sequences that we consider are subject to the restriction that the first $4k+4$ weights come before the next $2(k-1)$, which come before the next $2(k-2)$, and so on. The total number of different orderings of these sequences is

$$\begin{aligned}
&(2k+2)!(k-1)!(k-2)!\cdots(2)! \\
&> (\lfloor k/2 \rfloor !)^{\lfloor k/2 \rfloor} \\
&> \lfloor k/4 \rfloor^{\lfloor k/4 \rfloor \lfloor k/2 \rfloor} \\
&= \Omega(k^{\Theta(k^2)}).
\end{aligned}$$

Since $k = \Theta(n^{\frac{1}{2}})$, this number is $\Omega(n^{\Theta(n)}) = 2^{\Omega(n \lg n)}$. Theorem 5.3 is proven. $\square$

**6. Conclusions.** In this paper, we have extended the ideas of Hu and Tucker for constructing optimal alphabetic binary trees. In particular, we have used their basic idea of *lmcp-tree construction* together with the new idea of *region processing* to give $O(n)$-time algorithms to solve the cases where the input weights are within a constant factor, or exponentially separated. The constant factor case makes use of a new technique for doing generalized selection in $O(n)$ time. We show that any natural method employing either the idea of lmcp-tree construction or the idea of region processing may force us to sort a substantial amount of the input. The basic question of whether there is a general $o(n \lg n)$-time algorithm for finding optimal alphabetic binary trees for this problem remains open. In fact, this question is open even for the highly restricted case where no wallflowers are encountered in the construction of the lmcp tree before the point where the worklist contains only crossable nodes.

## REFERENCES

[1] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan, *Time bounds for selection*, J. Comput. System Sci., 7 (1972), pp. 448–461.

[2] A. M. Garsia and M. L. Wachs, *A new algorithm for minimum cost binary trees*, SIAM J. Comput., 6 (1977), pp. 622–642.

[3] E. N. Gilbert and E. F. Moore, *Variable length encodings*, Bell System Technical Journal, 38 (1959), pp. 933–968.

[4] T. C. Hu, D. J. Kleitman, and J. K. Tamaki, *Binary trees optimum under various criteria*, SIAM J. Appl. Math., 37 (1979), pp. 246–256.

[5] T. C. Hu and A. C. Tucker, *Optimal computer search trees and variable-length alphabetical codes*, SIAM J. Appl. Math., 21 (1971), pp. 514–532.

[6] D. E. Knuth, *Optimum binary search trees*, Acta Inform., 1 (1971), pp. 14–25.

[7] B. M. Mumey, *Some new results for constructing optimal alphabetic binary trees*, Master's thesis, University of British Columbia, 1992.

[8] P. Ramanan, *Testing the optimality of alphabetic trees*, Theoret. Comput. Sci., 93 (1992), pp. 279–301.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 652–660, November 1995

# ON FAMILIES OF SETS OF INTEGRAL VECTORS WHOSE REPRESENTATIVES FORM SUM-DISTINCT SETS*

DUŠAN B. JEVTIĆ†

**Abstract.** We study the size of a collection $\{\mathcal{P}_1, \ldots, \mathcal{P}_T\}$ whose members $\mathcal{P}_i$, $i = 1, \ldots, T$, are disjoint sets of integral vectors such that $\sum_{i=1}^{T} \bar{x}_i$ are all distinct and each $n$-tuple $\bar{x}_i$ comes from a different set $\mathcal{P}_i$. In particular, if $\mathcal{P}_i = \{\bar{0}_n, \bar{x}_i\}$, we have a well-known problem on maximum cardinality of sum-distinct sets of integral vectors. We state bounds on $n^{-1} \sum_{i=1}^{T} \log_2 |\mathcal{P}_i|$ and give a construction that meets the lower bound.

**Key words.** subset-sum, uniquely decodable, residual matrix, detecting matrix

**AMS subject classifications.** 11B75, 11B13

## 1. Introduction.

Let $\mathcal{K} \triangleq \{0, 1, \ldots, k\}$ for a given positive integer $k$ and denote by $\bar{0}_n$ the $n$-tuple of zeros. A collection $\{\mathcal{P}_1, \ldots, \mathcal{P}_T\}$ such that $\mathcal{P}_i \subset \mathcal{K}^n$ and $\mathcal{P}_i \cap \mathcal{P}_j = \{\bar{0}_n\}$ for $i \neq j$ will be called a *disjoint code* in $(\mathcal{K}^n, +)$ if the $\prod_{i=1}^{T} |\mathcal{P}_i|$ sums

$$\bar{x}_1 + \bar{x}_2 + \cdots + \bar{x}_T, \qquad \bar{x}_i \in \mathcal{P}_i, \tag{1.1}$$

are all distinct and each $n$-tuple $\bar{x}_i$ belongs to a different set $\mathcal{P}_i$.[1] In (1.1), $+$ stands for a component-wise real-number addition. The *rate* $R_i$ of a *component* $\mathcal{P}_i$ is defined by $|\mathcal{P}_i| = 2^{nR_i}$. The *rate sum*, denoted by $R_n(T)$, is the sum of $T$ component rates. For a $T$ component code we then have $R_n(T) = n^{-1} \log_2 |\mathcal{P}|$, where $\mathcal{P} \triangleq \mathcal{P}_1 \times \cdots \times \mathcal{P}_T$ and $\times$ denotes the Cartesian (direct) product of sets. Rate sum is a measure of the size of a code often used in information and coding theory. Our aim is to establish bounds on $R_n(T)$.

The above object was studied in [2] for $n = 1$. For $k = 1$, a weak upper bound on $R_n(T)$ may be found in [4]. Note that $\bar{0}_n \in \mathcal{P}_i$ is an additive zero and hence any subcollection of $\{\mathcal{P}_1, \ldots, \mathcal{P}_T\}$ is a disjoint code. This is important in multiple-access communications where no-transmission from user $i$ corresponds to $\bar{x}_i = \bar{0}_n$ in (1.1), thus making a resulting signal from any number of at most $T$ active users unique.

Call a set $\{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_m\}$, $\bar{x}_i \in \mathcal{K}^n$, *sum distinct* in $(\mathcal{K}^n, +)$ if all the $2^m$ possible sums $\sum_i \epsilon_i \bar{x}_i$, where $\epsilon_i \in \{0, 1\}$ and $\bar{x}_i \in \mathcal{K}^n$, are distinct $n$-tuples in $\{0, 1, \ldots, mk\}^n$. Thus, a $T$ component code generates $\prod_i (|\mathcal{P}_i| - 1)$ different sum-distinct sets of cardinality $T$. It is, however, unlikely to have more than one such set if we insist that $T$ is maximum for given $k$ and $n$.[2] If $\mathcal{P}_i = \{\bar{0}_n, \bar{x}_i\}$, the collection $\{\mathcal{P}_1, \ldots, \mathcal{P}_T\}$ is called a disjoint *signature code*. Its rate-sum is $n^{-1}T$, where $T$ is the number of components or the cardinality of

$$\{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_T\}, \qquad \bar{x}_i \in \mathcal{K}^n. \tag{1.2}$$

By the definition of a code, the set (1.2) must have the property that all of its $2^T$ subset-sums are distinct $n$-tuples in $\{0, 1, \ldots, kT\}^n$. This relates the signature code

* Received by the editors March 30, 1994; accepted for publication October 21, 1994.

† Computer Engineering Department, Santa Clara University, Santa Clara, California 95053.

[1] The full name here would be "disjoint uniquely decodable code." We will use the term disjoint code or just code.

[2] This statement is based on a futile computer search conducted in [2] for $n = 1$.

to the coin-weighing problem, e.g., [1], [6], and [9], as well as to the superimposed code, e.g., [3] and [5].

Upper and lower bounds on $R_n(T)$ are established in §§2 and 3, respectively. A construction which meets the lower bound is shown in §4. Related remarks are given in §5.

## 2. An upper bound.

A matrix with entries from $\mathcal{K}$ will be called a $\mathcal{K}$-matrix. Let $m_i = |\mathcal{P}_i|$ and $m = \sum_i m_i$. It is convenient to arrange the code $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_T\}$ into an $n$ by $m$ $\mathcal{K}$-matrix $\mathcal{C}$,

$$\mathcal{C} \triangleq (\mathcal{P}_1|\mathcal{P}_2|\cdots|\mathcal{P}_T), \tag{2.1}$$

such that $m_i$ column-vectors of $\mathcal{P}_i$ form the $i$th $n$-by-$m_i$ submatrix (block) of $\mathcal{C}$. Denote by $\omega(\bar{u})$ the number of 1's in a binary vector $\bar{u}$ and let $\mathcal{U}_i \triangleq \{\bar{z} \in \{0,1\}^{m_i} | \omega(\bar{z}) = 1\}$ and $\mathcal{U} \triangleq \mathcal{U}_1 \times \cdots \times \mathcal{U}_T$ be the sets of binary vectors compatible with the format of $\mathcal{C}$. Then $\{\mathcal{P}_1, \ldots, \mathcal{P}_T\}$ is a code if

$$\bar{\epsilon} = \mathcal{C}\bar{u}, \qquad \bar{u} \in \mathcal{U} \tag{2.2}$$

has at most one solution in $\bar{u} \in \mathcal{U}$ for any integral $n$-tuple $\bar{\epsilon}$. A matrix $\mathcal{C}$ for which $\bar{\epsilon}$ uniquely determines $\bar{u} \in \mathcal{U}$ will be called sum distinct.

If $\bar{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^T$, we can write (2.2) as

$$\epsilon_i = \bar{c}_i \bar{u}, \qquad i = 1, 2, \ldots, n, \tag{2.3}$$

where $\bar{c}_i$ is the $i$th row $m$-vector of $\mathcal{C}$. The simplest upper bound on $R_n(T)$ is obtained by realizing that $|\mathcal{P}|$ distinct points $\mathcal{C}\bar{u}$ should lie inside an $n$-dimensional sphere of radius $kT$ (the radius follows from (2.3) and the format of $\bar{u}$). That is, $|\mathcal{P}| \le V(kT, n)$, where $V(r,n) = (r\sqrt{\pi})^n / \Gamma(1 + \frac{n}{2})$ is the volume of the sphere of radius $r$. Hence,

$$R_n(T) \le \log_2 kT\sqrt{\pi} - n^{-1} \log_2 \Gamma\left(1 + \frac{n}{2}\right), \qquad \Gamma(z) \triangleq \int_0^\infty t^{z-1} e^{-t} dt. \tag{2.4}$$

By (2.3), there are at most $(1 + kT)^n$ distinct points $\mathcal{C}\bar{u}$ in $\{0, 1, \ldots, kT\}^n$. Furthermore, the $2^T$ sums (1.1) are all distinct. Hence, $2^T < (1 + kT)^n$ and, due to $T < (1 + k)^n$, it follows that $T < (n^2 + n)\log_2(1 + k)$. The substitution of this upper bound into $2^T < (1 + kT)^n$ yields

$$T < n \log_2 k + n \log_2[1 + (n^2 + n)\log_2(1 + k)]. \tag{2.5}$$

Hence, for large $n$, the upper bound on $n^{-1}T$ is approximately $\log_2 n^2 k \log_2(1 + k)$. The above estimate could be further improved by a repeated substitution of the most recent upper bound on $T$ into $2^T < (1 + kT)^n$.

Our final estimate of $R_n(T)$ in this section comes from a random coding argument. The intent is to improve the upper bound (2.4).

THEOREM 1. *$R_n(T) \le c + \log_2 k\sqrt{T}$, where $c < \log_2 \sqrt{2e\pi}$.*

*Proof.* If $\bar{u}_i$ is a random vector with uniform distribution on $\mathcal{U}_i$, then

$$E\bar{u}_i = \sum_{\bar{a} \in \mathcal{U}_i} \bar{a} P\{\bar{u}_i = \bar{a}\} = \frac{1}{m_i} \sum_{\bar{a} \in \mathcal{U}_i} \bar{a} = \frac{1}{m_i} \bar{1}_{m_i},$$

where $\bar{1}_{m_i}$ is the column $m_i$-vector of 1's. Let $R_{\bar{u}_i} \triangleq E(\bar{u}_i - E\bar{u}_i)(\bar{u}_i - E\bar{u}_i)^T$ and let $\bar{p} = (p_1, \dots, p_{m_i})$ such that $p_j \in \mathcal{K}$ for all $j$. Then,

$$\begin{aligned}\bar{p}R_{\bar{u}_i}\bar{p}^T &= \bar{p}(E\bar{u}_i\bar{u}_i^T)\bar{p}^T - \bar{p}(E\bar{u}_iE\bar{u}_i^T)\bar{p}^T \\ &= \frac{1}{m_i}\sum_{\bar{a}\in\mathcal{U}_i}(\bar{p}\bar{a})^2 - \frac{1}{m_i^2}(\bar{p}\bar{1}_{m_i})^2 \\ &\le \frac{m_i - 1}{m_i^2}\sum_{j=1}^{m_i} p_j^2\end{aligned}$$

and thus $\bar{p}R_{\bar{u}_i}\bar{p}^T \le (1-m_i^{-1})k^2$ for any $\bar{p} \in \mathcal{K}^{m_i}$. Hence, if $R_{\bar{u}} = \mathrm{dg}(R_{\bar{u}_1}, R_{\bar{u}_2}, \dots, R_{\bar{u}_T})$ and $\bar{c}_i$ is a row vector of $\mathcal{C}$, then

$$\bar{c}_iR_{\bar{u}}\bar{c}_i^T \le k^2T - k^2\sum_{i=1}^{T} m_i^{-1}. \tag{2.6}$$

Denote by $H(X)$ the entropy of a random variable $X$. The proof uses the estimate,

$$H(\bar{X}) \le \frac{n}{2}\log 2\pi e(\sigma_1^2\sigma_2^2\cdots\sigma_n^2)^{\frac{1}{n}}, \tag{2.7}$$

where $\bar{X} = (X_1, \dots, X_n)$ and $\sigma_i^2 = E(X_i - EX_i)^2$ for $i = 1, 2, \dots, n$. For the proof of (2.7) see, for example, [7].

If $\bar{u}$ in (2.2) is a random variable, so is the subset-sum $\bar{\epsilon}$. Let $\mathcal{R}(\mathcal{C}) \triangleq \{\mathcal{C}\bar{u} | \bar{u} \in \mathcal{U}\}$ and set $P\{\bar{\epsilon} = \bar{z}\} = 0$ for $\bar{z} \notin \mathcal{R}(\mathcal{C})$. Then, $P\{\bar{\epsilon} = \bar{\epsilon}_o\} = P\{\bar{\epsilon} = \mathcal{C}\bar{x}_o\} = P\{\bar{u} = \bar{x}_o\}$ and $\bar{x}_o \in \mathcal{U}$ is unique since $\mathcal{C}$ is sum distinct. Thus, $H(\bar{\epsilon}) = H(\bar{u})$ for any distribution of $\bar{u} \in \mathcal{U}$. Set $X_i = \bar{c}_i\bar{u}$ and by (2.7),

$$\begin{aligned}H(\bar{u}) &= H(\bar{c}_1\bar{u}, \bar{c}_2\bar{u}, \dots, \bar{c}_n\bar{u}) \\ &\le \frac{n}{2}\log 2\pi e + \frac{1}{2}\sum_{i=1}^{n}\log \bar{c}_iR_{\bar{u}}\bar{c}_i^T,\end{aligned}$$

where $R_{\bar{u}} = E(\bar{u}-E\bar{u})(\bar{u}-E\bar{u})^T$. We relate $H(\bar{u})$ to $R_n(T)$ by taking the components $\bar{u}_1, \bar{u}_2, \dots, \bar{u}_T$ of $\bar{u}$ to be independent random variables with uniform distributions on $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_T$, respectively. Then $H(\bar{u}) = \sum_i H(\bar{u}_i)$ and $H(\bar{u}_i) = \log_2 |\mathcal{P}_i|$. By (2.6) and the above inequality we then have

$$n^{-1}\log_2 |\mathcal{P}| \le \log_2 \sqrt{2\pi e} + \frac{1}{2}\log_2\left(k^2T - k^2\sum_{i=1}^{T} m_i^{-1}\right) \tag{2.8}$$

and thus the theorem is proved. □

In what follows, we will establish $R_n(T) \ge \log_2 k\sqrt{n}$. Since $\sqrt{2e\pi} \approx 4$, by (2.5) and Theorem 1, we see that $R_n(T)$ behaves as $\log_2 k\sqrt{n}$ for large $n$.

**3. A lower bound.** A lower bound on $R_n(T)$ will be established by constructing a set of residue-class-representing (RCR) vectors from a given nonsingular $\mathcal{K}$-matrix. If $\{\bar{x}_1, \dots, \bar{x}_m\}$ is sum distinct in $(\mathcal{K}^n, +)$, any arrangement of its $m$ column $n$-vectors into an $n$-by-$m$ matrix $C$ has the property that $\bar{\epsilon} = C\bar{u}$ uniquely determines $\bar{u} \in \{0, 1\}^m$ for any $\bar{\epsilon} \in \mathcal{Z}^n$. Hence, we will call $C$ a sum-distinct matrix as well.

Let $B = (b_{ij})$ be a nonsingular $\mathcal{K}$-matrix of order $n$ and let $\mathcal{U}$ be a unimodular matrix such that $B\mathcal{U} = \mathcal{B}$, where $\mathcal{B}$ is an $n$-by-$n$ lower triangular matrix. We write

$$\mathcal{B} = \begin{pmatrix} \beta_{11} & 0 & \dots & 0 \\ \beta_{21} & \beta_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{n1} & \beta_{n2} & \dots & \beta_{nn} \end{pmatrix}. \tag{3.1}$$

Since $B$ is nonsingular, $\beta_{ii} \in \mathcal{Z} - \{0\}$ and $\beta_{ji}$, $j = i, \dots, n$, are in a complete set of residues modulo $\beta_{ii}$. If $\Lambda_B$ and $\Lambda_{\mathcal{B}}$ are the integral lattices generated by column-vectors of $B$ and $\mathcal{B}$, respectively, then $\Lambda_B = \Lambda_{\mathcal{B}}$. We will denote the column-vectors of $\mathcal{B}$ by $\bar{\beta}_1, \bar{\beta}_2, \dots, \bar{\beta}_n$ and the corresponding fundamental parallelotope by $\Pi_{\mathcal{B}}$. The representation $B\mathcal{U} = \mathcal{B}$ (also called Hermite right equivalent form) is always possible, see, for example, [8].

To each diagonal element $\beta_{ii}$ from $\mathcal{B}$ we associate a sum-distinct integer set

$$\Gamma_i = \{\gamma_1^{(i)}, \gamma_2^{(i)}, \dots, \gamma_{k_i-1}^{(i)}, \gamma_{k_i}^{(i)}\}, \quad \gamma_{k_i}^{(i)} = \beta_{ii}, \tag{3.2}$$

such that $|\gamma_j^{(i)}| < |\beta_{ii}|$ for all $j = 1, \dots, k_i - 1$ and all elements of $\Gamma_i$ have the same sign. To each $\Gamma_i$ we then associate $\bar{\theta}_{k_i}^{(i)} = \bar{\beta}_i$ and the $(k_i - 1)$ *residual vectors*

$$\bar{\theta}_j^{(i)} = \gamma_j^{(i)} \bar{e}_i + r_{i+1,i}^{(j)} \bar{e}_{i+1} + \dots + r_{n,i}^{(j)} \bar{e}_n, \quad j = 1, \dots, k_i - 1, \tag{3.3}$$

where $\bar{e}_i$ is the $i$th column vector of $I_n$ and $I_n$ is the identity matrix of order $n$. In (3.3), $r_{i+\ell,i}^{(j)}$ are integers such that $|r_{i+\ell,i}^{(j)}| \le |\beta_{i+\ell,i}|$ and $r_{i+\ell,i}^{(j)}$ and $\beta_{i+\ell,i}$ have the same sign. This is necessary to keep each $\bar{\theta}_j^{(i)}$ inside $\Pi_{\mathcal{B}}$. Then the $(\Sigma_{i=1}^n k_i)$-element set

$$\mathcal{L} = \bigcup_{i=1}^{n} \{\bar{\theta}_j^{(i)} \mid j = 1, \dots, k_i\}$$

is sum distinct. In other words, the $2^{k_i}$ possible sums of the $i$th component

$$\epsilon_1 \bar{\theta}_1^{(i)} + \epsilon_2 \bar{\theta}_2^{(i)} + \dots + \epsilon_{k_i-1} \bar{\theta}_{k_i-1}^{(i)} + \epsilon_{k_i} \bar{\beta}_i, \qquad \epsilon_i \in \{0, 1\}, \tag{3.4}$$

are distinct and each of the sums (3.4) is incongruent (mod $\Lambda_{\mathcal{B}}$) to any other sum from a different component. Hence, if $r(B)$ denotes the total number of residual vectors induced by a $\mathcal{K}$-matrix $B$ of order $n$, then

$$r(B) \ge -n + |\Gamma_1| + |\Gamma_2| + \dots + |\Gamma_n|. \tag{3.5}$$

By the above described construction, the vector-congruence problem in $\Lambda_B$ was reduced to a componentwise vector-congruence problem in a geometrically equivalent lattice $\Lambda_{\mathcal{B}}$. Hence, if $B$ is a $\mathcal{K}$-matrix of order $n$ and $|\det B| > 1$, we can augment $B$ with $r(B)$ residual vectors so that $(B|B_R)$ is a sum-distinct $n$-by-$m$ $\mathcal{K}$-matrix. We will call the $n$-by-$(m-n)$ $\mathcal{K}$-matrix $B_R$ a *matrix of residual vectors.*

For any positive integer $\ell_i \le \log_2(1 + |\beta_{ii}|)$, the set $\{2^0, 2^1, \dots, 2^{\ell_i - 1}, |\beta_{ii}|\}$ is sum distinct. From (3.5) and $|\Gamma_i| \ge \ell_i + 1$, we then have $r(B) \ge \sum_i \lfloor \log_2 |\beta_{ii}| \rfloor$. A simple condition which implies $r(B) \ge \lfloor \log_2 |\det B| \rfloor$ will be given in §5. Note that $r(B) \not\ge \lfloor \log_2 |\det B| \rfloor$. The smallest counterexample is the set $\{3, 5, 6, 7\}$. (Take $n = 1$, $B = (7)$, and $k = 7$.) As a side point, an upper bound on $r(B)$ may be obtained from an upper bound on $|\Gamma_i|$, say, $|\Gamma_i| \le \lceil \log_2 |\beta_{ii}| \rceil + 1 + \frac{1}{2}\log_2(2 + \frac{3\log_2|\beta_{ii}|}{2})$, which, to the author's knowledge, seems to be the best available.[3]

THEOREM 2. *Let $n = 2^r - 1$, where $r \in \mathcal{Z}^+$. Then there exists a sequence $(B_n)$ of $\mathcal{K}$-matrices such that*

$$r(B_n) \ge n[r(k) - 1] + \frac{n+1}{2}\log_2(n+1). \tag{3.6}$$

*Proof.* Define the sequence of integral matrices $\{B_n \mid n = 2^r - 1,\ r \in \mathcal{Z}^+\}$ recursively as $B_1 = (k)$ and

$$B_{2n+1} = \begin{pmatrix} B_n & \bar{0}_n & B_n \\ \bar{0}_n^T & k & \bar{k}_n^T \\ B_n & \bar{k}_n & B_n^c \end{pmatrix}, \qquad B_n^c = \bar{k}_n \bar{1}_n^T - B_n, \tag{3.7}$$

where $\bar{k}_n = k\bar{1}_n$. Let $\mathcal{U}_n$ be a unimodular matrix such that $B_n\mathcal{U}_n = \mathcal{B}_n$, where $\mathcal{B}_n$ is a lower triangular integral matrix of order $n$. Furthermore, let $O_n$ be the zero matrix of order $n$. It is shown by inspection that matrices $B_{2n+1}$,

$$\mathcal{U}_{2n+1} = \begin{pmatrix} \mathcal{U}_n & \bar{0}_n & -\mathcal{U}_n \\ \bar{0}_n^T & 1 & -\bar{1}_n^T\mathcal{U}_n \\ O_n & \bar{0}_n & \mathcal{U}_n \end{pmatrix}, \quad \text{and} \quad \mathcal{B}_{2n+1} = \begin{pmatrix} \mathcal{B}_n & \bar{0}_n & O_n \\ \bar{0}_n^T & k & \bar{0}_n^T \\ \mathcal{B}_n & \bar{k}_n & -2\mathcal{B}_n \end{pmatrix}$$

satisfy $B_{2n+1}\mathcal{U}_{2n+1} = \mathcal{B}_{2n+1}$. Let $h$ be the largest element in a sum-distinct set $\mathcal{X} \subset \mathcal{Z}^+$ and let $z \in \mathcal{Z}^+$ be an odd integer smaller than $2h$. The set $\{z\} \cup \{2t \mid t \in \mathcal{X}\}$ is sum distinct and has cardinality $|\mathcal{X}| + 1$, and its largest element is $2h$. Hence, $r(B_1) = r(k)$ and

$$\begin{aligned} r(B_{2n+1}) &= r(B_n) + r(k) + r(2B_n) \\ &\ge 2r(B_n) + n + r(k). \end{aligned}$$

The theorem follows from the above inequality by using standard techniques in solving linear recurrence equations. □

From the fact that $r(k) \ge \lfloor \log_2 k \rfloor$, inequality (3.6), and the definition of the rate sum, it follows that

$$R_n(T) \ge \lfloor \log_2 k \rfloor + \frac{1 + n^{-1}}{2}\log_2(1+n), \quad n = 2^r - 1,\ r \in \mathcal{Z}^+. \tag{3.8}$$

Note that the above bound is often conservative, since $r(k)$ may be underestimated. For example, $\lfloor \log_2 7 \rfloor = 2$ while $r(7) = 3$ for the previously mentioned set $\{3, 5, 6, 7\}$. It is thus better to write $r(k)$ instead of $\lfloor \log_2 k \rfloor$ in (3.8).

## 4. Inverse mapping.

Suppose we are given an $n$-by-$m$ sum-distinct $\mathcal{K}$-matrix $D$, and let $\bar{\epsilon}$ be a given $n$-tuple obtained by summing up any number of column-vectors from $D$. That is,

$$\bar{\epsilon} = D\bar{u}, \qquad \bar{u} \in \{0,1\}^m. \tag{4.1}$$

Call *inverse mapping* a procedure which recovers the column-vectors of $D$ which are in $\bar{\epsilon}$ (i.e., a procedure which calculates $\bar{u}$). Clearly, (4.1) has a solution if (and only if)

[3] A precise upper bound on the cardinality $T$ of a sum-distinct set in $(\mathcal{K}, +)$ is an open problem. It is suspected that $T < c + \log_2 k$ where $c \in \mathcal{Z}^+$ does not depend on $k$.

$\bar{\epsilon}$ is one of the $2^m$ distinct $n$-tuples from $\mathcal{R}(D)$. In that case, the $j$th column-vector from $D$ is in $\bar{\epsilon}$ if $u_j = 1$. Since (4.1) represents $n$ equations in $m > n$ unknowns, the additional $(m-n)$ equations, which are specified by $\bar{\epsilon} \in \mathcal{R}(D)$, the sum-distinctness of $D$, and $\bar{u} \in \{0,1\}^m$, must be hidden in relationships involving coordinates of $\bar{\epsilon}$.[4] Let $B$ be a submatrix of $D$ with $|\det B| > 1$ and write $D = (B|B_R)$, where $B_R$ is the matrix of residual vectors of $B$. If $\bar{u}^T = (\bar{u}_B^T, \bar{u}_R^T)$, we can rewrite (4.1) as

$$\bar{\epsilon} = B\bar{u}_B + B_R\bar{u}_R. \tag{4.2}$$

If $\bar{u}_R$ is known, the system of $n$ equations in $n$ unknowns $B\bar{u}_B = \bar{\epsilon} - B_R\bar{u}_R$ can be solved for $\bar{u}_B \in \{0,1\}^n$ by using standard techniques. This is a general idea in RCR constructions. We now present a construction which is based on Theorem 2.

Let $\ell = \lfloor \log_2 k \rfloor$ and define $\bar{a}_\ell^T = (2^0, 2^1, \ldots, 2^{\ell-1})$ if $\ell > 0$ and $\bar{a}_0 = (0)$. Furthermore, let $D_n$ be an $n$-by-$m$ $\mathcal{K}$-matrix and define a $(2n+1)$-by-$(2m+\ell+n+1)$ $\mathcal{K}$-matrix $D_{2n+1}$ in terms of $D_n$ and $\bar{a}_\ell$ as

$$D_{2n+1} = \begin{pmatrix} D_n & \bar{0}_n & D_n & \bar{1}_n\bar{a}_\ell^T & O_n \\ \bar{0}_m^T & k & \bar{k}_m^T & \bar{a}_\ell^T & \bar{0}_n^T \\ D_n & \bar{k}_n & D_n^c & O_{n\times\ell} & I_n \end{pmatrix}, \quad D_n^c = \bar{1}_n\bar{k}_m^T - D_n,$$

where $O_{p\times q}$ is a $p$-by-$q$ zero matrix. Then, the following statements hold true.

(i) If $D_n$ is sum distinct, so is $D_{2n+1}$.

(ii) The sum-distinct set formed by column-vectors of $D_{2n+1}$ is RCR; residual vectors of $D_{2n+1}$ are the last $(\ell+n)$ column-vectors.

(iii) Let $m(n)$ denote the number of sum-distinct $n$-vectors in submatrix $D_n$. If $D_n$ has the same structure as $D_{2n+1}$ (i.e., if $D_n$ is defined recursively), then

$$m(n) = n\ell + \frac{n+1}{2}\log_2(n+1), \qquad n = 2^r - 1, \qquad r \in \mathcal{Z}^+. \tag{4.3}$$

We will prove (i) by showing that $\bar{\epsilon} = D_{2n+1}\bar{u}$, $\bar{u} \in \{0,1\}^{2m+\ell+n+1}$, has a unique solution in $\bar{u}$ if $\bar{\epsilon} \in \mathcal{R}(D_{2n+1})$. To that end, let the binary column-vector $\bar{u}$ be partitioned as $\bar{u}^T = (\bar{u}_1^T, t, \bar{u}_2^T, \bar{z}^T, \bar{w}^T)$, where $t \in \{0,1\}$, $\bar{z} \in \{0,1\}^\ell$, $\bar{w} \in \{0,1\}^n$, and $\bar{u}_i \in \{0,1\}^m$ for $i = 1,2$. Furthermore, let the $(2n+1)$-vector $\bar{\epsilon}$ be partitioned as $\bar{\epsilon}^T = (\bar{\epsilon}_1^T, s, \bar{\epsilon}_2^T)$, where $\bar{\epsilon}_1$ and $\bar{\epsilon}_2$ are column $n$-vectors with nonnegative integral components and $s$ is a nonnegative integer. Then $\bar{\epsilon} = D_{2n+1}\bar{u}$ can be written as

$$\bar{\epsilon}_1 = D_n\bar{u}_1 + D_n\bar{u}_2 + \bar{1}_n(\bar{a}_\ell^T\bar{z}), \tag{4.4a}$$

$$s = kt + \bar{k}_m^T\bar{u}_2 + \bar{a}_\ell^T\bar{z}, \tag{4.4b}$$

$$\bar{\epsilon}_2 = D_n\bar{u}_1 + t\bar{k}_n + (\bar{1}_n\bar{k}_m^T - D_n)\bar{u}_2 + \bar{w}. \tag{4.4c}$$

Note that (4.4b) can be written as $\bar{a}_\ell^T\bar{z} = s - (\omega(\bar{u}_2) + t)k$ and hence, due to $\bar{a}_\ell^T\bar{z} < k$ and the definition of $\bar{a}_\ell$, the binary vector $\bar{z}$ is uniquely determined by

$$\bar{a}_\ell^T\bar{z} = s \bmod k. \tag{4.5}$$

[4] A brute-force approach would be to choose $2^{m-n}$ binary tuples $(u_1, \ldots, u_{m-n})$ and check (for each such tuple) if $n$ remaining components of $\bar{u}$ satisfy (4.1). By (3.8), this approach is quite expensive if $kn$ is large.

Premultiplying (4.4b) by $\bar{1}_n$ and replacing $(t\bar{k}_n + \bar{1}_n \bar{k}_m^T \bar{u}_2)$ by $\bar{1}_n(s - \bar{a}_\ell^T \bar{z})$ in (4.4c) yields the system

$$\begin{aligned}\bar{\epsilon}_1 &= D\bar{u}_1 + D\bar{u}_2 + \bar{1}_n(\bar{a}_\ell^T \bar{z}),\\ \bar{\epsilon}_2 &= D\bar{u}_1 - D\bar{u}_2 - \bar{1}_n(\bar{a}_\ell^T \bar{z}) + \bar{w} + s\bar{1}_n,\end{aligned}$$

which has $2n$ equations in $2m + n$ unknowns, $\bar{u}_1$, $\bar{u}_2$, and $\bar{w}$. By adding the above two equations, we have $2(D_n\bar{u}_1 + s\bar{1}_n) + \bar{w} = \bar{\epsilon}_1 + \bar{\epsilon}_2 + s\bar{1}_n$ and hence

$$\bar{w} = \bar{\epsilon}_1 + \bar{\epsilon}_2 + s\bar{1}_n \pmod 2. \tag{4.6}$$

Once $\bar{z}$ and $\bar{w}$ are known (that is, once residual vectors are extracted) we have

$$D_n\bar{u}_1 = \frac{1}{2}(\bar{\epsilon}_1 + \bar{\epsilon}_2 - \bar{w} - s\bar{1}_n), \tag{4.7a}$$

$$D_n\bar{u}_2 = \frac{1}{2}(\bar{\epsilon}_1 - \bar{\epsilon}_2 + \bar{w} + s\bar{1}_n) - (s \bmod k)\bar{1}_n, \tag{4.7b}$$

which, since $D_n$ is sum distinct, determine $\bar{u}_1$ and $\bar{u}_2$ uniquely. Once $\bar{u}_2$ is known, we calculate $t \in \{0, 1\}$ from (4.4b) as

$$t = \left\lfloor \frac{s}{k} \right\rfloor - \omega(\bar{u}_2), \tag{4.8}$$

and thus the $(2m + \ell + n + 1)$ binary vector $\bar{u}$ is determined completely.

The claim in (ii) follows from (4.5), (4.6), and the definition of a residual vector. We leave it to the reader to verify that $I_n$ represents the residual vectors obtained from $-2\mathcal{B}_n$. To verify (iii), let $m(2n+1)$ denote the number of sum-distinct column $(2n+1)$-vectors in $D_{2n+1}$. By the construction of $D_{2n+1}$, $m(2n+1) = 2m(n) + n + \ell + 1$, $m(0) = 0$, where $m(n)$ is the number of sum-distinct $n$-vectors in $D_n$. The substitutions $n + 1 = 2^k$ and $\phi(k) = m(2^k - 1)$ yield $\phi(k + 1) = 2\phi(k) + 2^k + \ell$, from which (4.3) follows by using standard techniques in solving linear recurrence equations.

If $D_n$ is defined recursively, there has to be a recursive inverse mapping for determining the subset vectors from a given subset-sum. The mapping is defined by (4.5), the algorithm which recovers $\bar{z}$ from the known product $\bar{a}_\ell^T \bar{z}$, and (4.6)–(4.8). To stop the recursive calls in (4.7a) and (4.7b), we may use $D_1 = (k \mid \bar{a}_\ell^T)$. The inverse mapping requires storing matrix $D_n$ only. Let $\tau_n$ be its running time for the sum-distinct matrix $D_n$. From (4.5)–(4.8), it follows that $\tau_{2n+1} = 2\tau_n + c_1 n + c_2 \log_2 k + c_3$, where $c_1$, $c_2$ and $c_3$ are machine-dependent constants. By using standard techniques, we obtain

$$\tau_n = \frac{n-1}{2}(c_3 - c_1 + c_2 \log_2 k) + \frac{n+1}{2}\tau_1 + c_1 \frac{n+1}{2} \log_2 \frac{n+1}{2}, \quad n+1 = 2^r, \quad r \in \mathcal{Z}^+.$$

**5. Remarks.** If $r(B)$ is the number of residual vectors induced by a square $\mathcal{K}$-matrix $B$, then $2^{r(B)} \approx |\det B|$. This is so because $|\det B|$ is the volume of the fundamental parallelotope formed by column-vectors of $B$ and *most* of the sums (3.4) are inside the parallelotope. If we write $\mathcal{B}$ in (3.1) as

$$\mathcal{B} = \begin{pmatrix} L & \bar{0} \\ \bar{\alpha}^T & \beta_{nn} \end{pmatrix}, \quad \bar{\alpha}^T = (\beta_{n1}, \beta_{n2}, \ldots, \beta_{n,n-1}), \tag{5.1}$$

then from $r(B) \geq \sum_i \lfloor \log_2 |\beta_{ii}| \rfloor$ we see that $r(B) \geq \lfloor \log_2 |\det B| \rfloor$ is implied by either $|\beta_{ii}| = 2^{\ell_i}$ for all $i$ or $L = I_{n-1}$ in (5.1). The first condition was used in the sequence $(B_n | n = 3, 7, \ldots)$ given by (3.7); here we give a simple sufficient condition for $L = I_{n-1}$.

REMARK 1. *Let $p$ be a permutation of column-vectors of a nonsingular $\mathcal{K}$-matrix $B$ of order $n$ such that*

$$pB = \begin{pmatrix} A & \bar{b} \\ \bar{c}^T & t \end{pmatrix} \tag{5.2}$$

*and* $\gcd(|\det A|, |\det B|) = 1$. *Then* $r(B) \geq \lfloor \log_2 |\det B| \rfloor$.

*Proof.* Let $\mathcal{B}$ in (5.1) be the lower triangular form of $pB$. Furthermore, let $d_n \triangleq \gcd(|\det A|, |\det B|)$ and $\bar{z} = (z_1, \ldots, z_{n-1})^T$, $z_i \in \mathcal{Z}$ for $i = 1, \ldots, n$, such that $\bar{\alpha}^T L^{-1} A + \beta_{nn} \bar{z}^T = \bar{c}^T$ and $\bar{\alpha}^T L^{-1} \bar{b} + \beta_{nn} z_n = t$. Then $\mathcal{B}\mathcal{U}^{-1} = pB$ becomes

$$\begin{pmatrix} L & \bar{0} \\ \bar{\alpha}^T & \beta_{nn} \end{pmatrix} \begin{pmatrix} L^{-1}A & L^{-1}\bar{b} \\ \bar{z}^T & z_n \end{pmatrix} = \begin{pmatrix} A & \bar{b} \\ \bar{c}^T & t \end{pmatrix}.$$

Since $\mathcal{U}^{-1}$ has integral entries, $|\det\ L^{-1}A| \in \mathcal{Z}^+$ or $|\det A| = s|\beta_{nn}|^{-1}|\det pB|$, where $s \in \mathcal{Z}^+$. Due to $|\det pB| = |\det B|$ and $|\det B| = \prod_{i=1}^n |\beta_{ii}|$, we have $d_n = c\prod_{i=1}^{n-1} |\beta_{ii}|$, where $c \in \mathcal{Z}^+$, and thus the claim follows.

It is not difficult to show that if $d_n = 1$, then $\det A$ and components of $(\text{adj } A)\bar{b}$ are mutually prime. Furthermore, if $|\beta_{nn}|$ is to be maximum, due to $\beta_{nn} z_n = t - \bar{\alpha}^T \bar{b}$, we have $z_n = \pm 1$. Thus, if $d_n = 1$ and $|\beta_{nn}|$ is maximum, then

$$\begin{pmatrix} I_{n-1} & \bar{0} \\ \bar{\alpha}^T & \beta_{nn} \end{pmatrix} \begin{pmatrix} A & \bar{b} \\ \bar{z}^T & 1 \end{pmatrix} = \begin{pmatrix} A & \bar{b} \\ \bar{c}^T & t \end{pmatrix},$$

which can be used in the construction of matrix $B$. □

In §3, $\{3, 5, 6, 7\}$ was a counterexample to $r(B) \leq \lfloor \log_2 |\det B| \rfloor$, and in §4, we used the same set to claim $r(k) > \lfloor \log_2 k \rfloor$. Let $n = 3$ and $k = 7$ and consider the 3-by-13 matrix

$$D_3 = \begin{pmatrix} 3 & 5 & 6 & 7 & 0 & 3 & 5 & 6 & 7 & 3 & 5 & 6 & 0 \\ 0 & 0 & 0 & 0 & 7 & 7 & 7 & 7 & 7 & 3 & 5 & 6 & 0 \\ 3 & 5 & 6 & 7 & 7 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5.3}$$

Let $\bar{u} = (u_1, \ldots, u_{13})^T$ and $\bar{\epsilon} = (\epsilon_1, \epsilon_2, \epsilon_3)^T$. From

$$\begin{aligned} (1, -1, 1) D_3 \bar{u} &= \epsilon_1 - \epsilon_2 + \epsilon_3 \\ &= 2(3u_1 + 5u_2 + 6u_3 + 7u_4) + u_{13}, \end{aligned}$$

we see that $u_{13}$ is uniquely determined as

$$u_{13} = \epsilon_1 - \epsilon_2 + \epsilon_3 \pmod 2. \tag{5.4}$$

Since $\{3, 5, 6, 7\}$ is sum distinct, the same identity determines $(u_1, u_2, u_3, u_4)$ uniquely as

$$3u_1 + 5u_2 + 6u_3 + 7u_4 = \frac{1}{2}(\epsilon_1 - \epsilon_2 + \epsilon_3 - u_{13}). \tag{5.5}$$

Furthermore, premultiplying $D_3 \bar{u} = \bar{\epsilon}$ by $(0, 1, 0)$ yields

$$3u_{10} + 5u_{11} + 6u_{12} = \epsilon_2 \bmod 7. \tag{5.6}$$

Even though $\{3,5,6\}$ is sum distinct, the inverse mapping does not seem to be unique, since ($\epsilon_2$ mod 7) is zero when either $(u_{10},u_{11},u_{12})^T=\bar{0}_3$ or $\bar{1}_3$. Suppose that the subset-sum is $(14,14,0)^T$ and that $(u_{10},u_{11},u_{12})^T=\bar{0}_3$ is assumed in (5.6). Then from $(1,0,0)D_3\bar{u}=\epsilon_1$, we have

$$3u_6+5u_7+6u_8+7u_9=\frac{1}{2}(\epsilon_1+\epsilon_2-\epsilon_3+u_{13})-3u_{10}-5u_{11}-6u_{12}.$$

Hence, $(u_6,u_7,u_8,u_9)=(1,1,1,0)$ and the error goes undetected. If we are convinced that $D_3$ is sum distinct (say by displaying its $2^{13}$ subset-sums), the inverse mapping may be made unique by requiring that if ($\epsilon_2$ mod 7) is zero, both solutions ($\bar{1}_3$ and $\bar{0}_3$) of (5.6) are run, yielding $\bar{u}^{(1)}$ and $\bar{u}^{(0)}$, respectively. After comparing $D_3\bar{u}^{(0)}$ and $D_3\bar{u}^{(1)}$ against the subset-sum $\bar{\epsilon}$, the correct one is taken.

Our final comment concerns a direction not pursued here. If $\mathcal{BU}=\mathcal{B}$ and $\mathcal{D}=(\mathcal{B}|\mathcal{B}_R)$, then there exists an $m$-by-$m$ unimodular matrix $\mathcal{U}_m$ such that $D\mathcal{U}_m=\mathcal{D}$ and $D=(B|B_R)$ is a sum-distinct $\mathcal{K}$-matrix. Hence,

$$\bar{\epsilon}=D\bar{u}=\mathcal{D}\mathcal{U}_m^{-1}\bar{u}=\mathcal{D}\bar{t},$$

where $\bar{t}$ is an $m$-vector from $\mathcal{Z}^n$. If $\mathcal{D}\bar{t}=\bar{\epsilon}$ has a unique solution in $\bar{t}$, it is possible to recover $\bar{u}$ as $\mathcal{U}_m\bar{t}$ uniquely. Such inverse mapping would clearly require a choice of $\mathcal{B}_R$ which guarantees this unique solution.

## REFERENCES

[1] D. G. Cantor and W. H. Mills, *Determination of a subset from certain combinatorial properties*, Canad. J. Math., 18 (1966), pp. 42–48.

[2] P. Erdös and D. B. Jevtić, *Problem* 91-2: *Partial solution*, SIAM Rev., 34 (1992), pp. 309–310.

[3] T. Ericson and L. Györfi, *Superimposed codes in $R^n$*, IEEE Trans. Inform. Theory, IT-34 (1988), pp. 877–880.

[4] D. B. Jevtić, *Disjoint uniquely decodable codebooks for noiseless synchronized multiple access adder channels generated by integer sets*, IEEE Trans. Inform. Theory, IT-38 (1992), pp. 1142–1146.

[5] W. K. Kautz and R. C. Singleton, *Nonrandom binary superimposed codes*, IEEE Trans. Inform. Theory, IT-10 (1964), pp. 363–377.

[6] B. Lindström, *On a combinatorial problem in number theory*, Canad. Math. Bull., 8 (1965), pp. 477–490.

[7] R. J. McEliece, *The theory of information and coding*, in Encyclopedia of Mathematics and its Applications, vol. 3, Addison–Wesley, Reading, MA, 1977, pp. 38–39.

[8] M. Newman, *Integral Matrices*, Academic Press, New York, 1972.

[9] S. Söderberg and H. S. Shapiro, *A combinatory detection problem*, Amer. Math. Monthly, 70 (1963), pp. 1066–1070.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 661–669, November 1995

# AN APPROXIMATION ALGORITHM FOR PREEMPTIVE SCHEDULING ON PARALLEL-TASK SYSTEMS*

RAMESH KRISHNAMURTI† AND BHAGIRATH NARAHARI‡

**Abstract.** This paper addresses the problem of preemptive scheduling on parallel-task systems. A parallel-task system consists of several independent tasks, each of which can be executed on one or more processors. Du and Leung introduced the problem of minimizing the schedule length in parallel-task systems and showed that it is strongly NP-hard for both nonpreemptive and preemptive scheduling of independent tasks. This paper presents a polynomial-time approximation algorithm for preemptive scheduling of a parallel-task system with $n$ processors and $w$ tasks. We derive a tight worst-case performance bound of $r$ for the approximation algorithm, where $r$ is the maximum factor by which we can increase the number of processors on which a task can be executed. For example, $r = 2$ in the model defined by Du and Leung for parallel-task systems in which a task can be executed on any integral number of processors.

**Key words.** parallel-task system, preemptive scheduling, schedule length, polynomial time, approximation algorithm

**AMS subject classifications.** 68M20, 90B35, 11Y16

## 1. Introduction.

Parallel-task systems allow a task to be executed on one or more processors, with execution times varying with the number of processors used to execute the task. A number of parallel architectures [6], which fit the model of parallel-task systems, have been designed in recent years. The problem of finding a minimum-length preemptive schedule for a set of independent parallel tasks was shown by Du and Leung [3] to be strongly NP-hard for an arbitrary number of processors. For a fixed number of processors, the preemptive-scheduling problem is shown to be NP-hard but solvable in pseudopolynomial time. In our model, we assume that the efficiency of any task (efficiency is defined as the speedup divided by the number of processors) is nonincreasing with an increasing number of processors; i.e., the processor-time product is a nondecreasing function of the number of processors allocated for a task. We also assume that tasks can be preempted, but the number of processors that execute the task at any point in time remains constant.

This paper presents a polynomial-time approximation algorithm for preemptive scheduling of a set of independent tasks on parallel-task systems for an arbitrary number of processors. We derive a tight worst-case performance bound of $r$ for the approximation algorithm, where $r$ is the maximum factor by which we can increase the number of processors on which a task can be executed. The parameter $r$ depends on the properties of the parallel-task system. For example, in the model discussed in [3], it follows that $r = 2$ (this occurs when the number of processors for a task is increased from 1 to 2), and thus the algorithm generates a schedule with schedule length at most twice that of the optimal schedule. Furthermore, for hypercube-based systems [6], where tasks are executed on subcubes of size $2^i$, we again have $r = 2$.

A number of studies have, under different problem models, dealt with the problem of scheduling tasks on systems with multiple processors. Garey and Johnson [4],

* Received by the editors October 19, 1991; accepted for publication (in revised form) November 11, 1994. This research was supported in part by National Sciences and Engineering Research Council of Canada operating grant OGP-0036809 and National Science Foundation grant NSF-CCR-9209345.

† School of Computing Science, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada (ramesh@cs.sfu.ca).

‡ Department of Electrical Engineering and Computer Science, The George Washington University, Washington, DC 20052 (narahari@seas.gwu.edu).

[5] study the multiprocessor scheduling problem with resource constraint (where the resource could be processors) and show that this problem is NP-hard even for the special case in which there is only one resource, with unit execution time for each task. Multiprocessor-scheduling problems in which each task requires a specific number of processors have received significant attention in the literature [1], [2]. It is shown in [1] that nonpreemptive scheduling on these systems is NP-hard for arbitrary $n$, where $n$ is the number of processors. However, for each fixed $n$, an optimal preemptive schedule can be found in polynomial time. Scheduling on multiprocessor-task systems that models communication delays between tasks with precedence constraints has also been addressed [9]. The reader is referred to [3] for a detailed exposition on different scheduling models and their relation to the parallel-task system model. The parallel-task system model adds a new dimension to past scheduling models in the sense that the schedule must also determine the number of processors to be used for each task. For parallel-task systems, [7] presents an approximation algorithm for nonpreemptive scheduling of independent tasks on parallel-task systems, when the number of tasks is at most equal to the number of processors. Apart from Du and Leung [3], the problem of preemptive scheduling on parallel-task systems has not been addressed in the literature.

The next section defines our notation and formulates the preemptive scheduling problem. The approximation algorithm is presented in §3, and its performance is analyzed in §4.

**2. Notation and problem model.** For a typical schedule, each task is assigned a collection of a number of processors, to which we refer as the *processor collection*, and its size is the number of processors in the collection. Let $w$ denote the number of tasks and $n$ the number of processors. For any integer $a$, let $[a]$ denote the set $\{1, \ldots, a\}$.

For each task $i \in [w]$, the characteristics of task $i$ are abstracted into a tuple $(q_i, p_i, t_i)$ where

- $q_i$, for $q_i \in Z^+$, denotes the number of choices of different processor collections (number of processors) on which the task $i$ can be executed,
- $p_i$ is a function $p_i : [q_i] \rightarrow [n]$, which defines the number of processors for each choice $a_j \in [q_i]$, and
- the function $t_i : [q_i] \rightarrow Z^+$ defines the execution time of task $i$ for each choice of processor collection.

Task $j$ can be executed on any processor collection consisting of $p_j(a_j)$ processors, with processing time (execution time) $t_j(a_j)$ for $a_j \in \{1, \ldots, q_j\}$. We assume that the processing time of a task decreases with an increasing number of processors and that the processor-time product is a nondecreasing function of the number of processors allocated to the task. These properties are defined as follows.

(i) $p_i(1) = 1$, $p_i(k) < p_i(k+1)$, and $t_i(k) > t_i(k+1)$ for all $i \in [w]$ and $1 \leq k \leq q_i - 1$. This condition can be satisfied for *all* parallel-task systems by simply removing all processor-collection choices that violate this property.

(ii) $p_i(k)t_i(k) \leq p_i(k+1)t_i(k+1)$ for all $i \in [w]$ and $1 \leq k \leq q_i - 1$. When the number of processors is increased by a factor of $\ell$, its execution time is decreased by a factor at most $\ell$.

Our problem is to find a preemptive schedule with minimum completion time for all tasks. A preemptive schedule must determine the number of processors $p_i(a_i)$ for each task $i$, and schedule them on the processors in the system; we assume that a task can be preempted but the number of processors assigned to it remains constant.

This preemptive-scheduling problem is shown to be strongly NP-hard in [3].

The parameter $r$ denotes the maximum factor by which we can increase the number of processors used to execute a task, and $r$ is defined as

$$r = \max_{i \in \{l \mid l \in [w],\, q_l > 1\}} \left\{ \max_{j \in [q_i - 1]} \left\{ \frac{p_i(j+1)}{p_i(j)} \right\} \right\}.$$

When $q_i = 1$, i.e., a task can be executed only on one processor, we define $r$ to be 1, and thus $r$ is always well defined. For the model in [3], it follows that $r = 2$ (this occurs when the number of processors for a task is increased from 1 to 2, i.e., $p_i(2)/p_i(1) = 2$).

For any preemptive schedule, a task is called a *dominating task* if its execution time in the schedule is equal to the completion time of the schedule. A task that is assigned to one processor is called a *uniprocessor task.*

Given a set $U_1$ of uniprocessor tasks (i.e., tasks running on a single processor) and $n$ processors, an optimal preemptive schedule can be generated using McNaughton's wraparound algorithm [8]. It is shown in [8] that the optimal preemptive schedule time is given by $\max\{T_{\text{avg}}, T_{\max}\}$ where $T_{\text{avg}} = (1/n)\sum_{i \in U_1} t_i(1)$ is the average execution time of the tasks and $T_{\max} = \max_{i \in U_1} t_i(1)$ is the maximum among all task execution times. The above result implies that for any time $T$, where $T \geq \max\{T_{\max}, T_{\text{avg}}\}$, the minimum number of processors $m$ required to preemptively schedule tasks in $U_1$ (using McNaughton's algorithm) is given by $m = \lceil \sum_{i \in U_1} t_i(1)/T \rceil$. This property is exploited by our approximation algorithm.

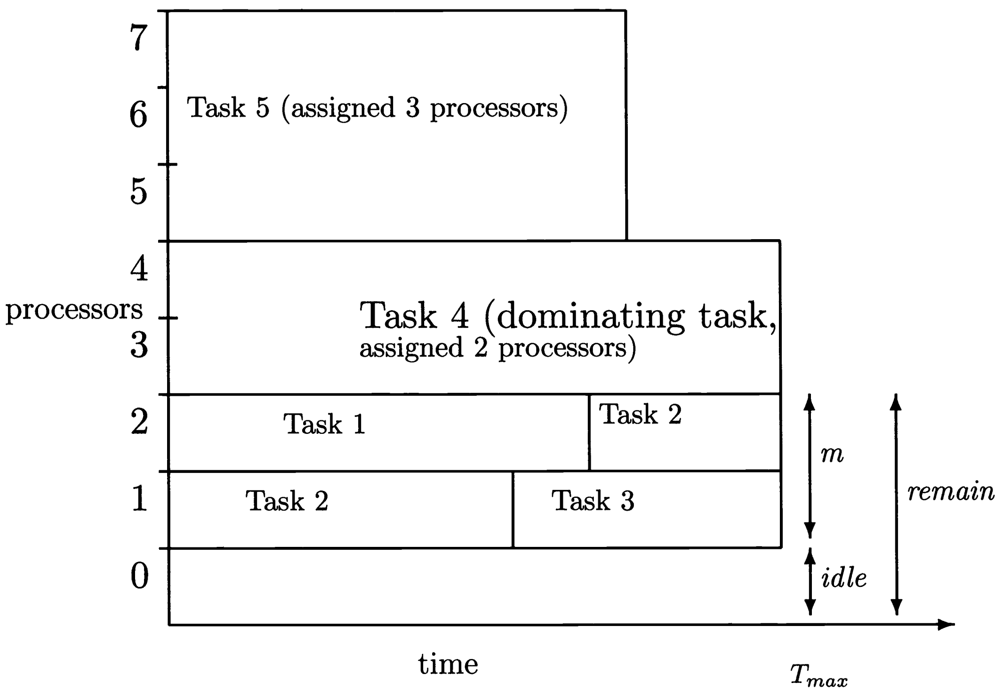
## 3. An approximation algorithm.

The approximation algorithm is based on a local improvement strategy. It starts with an initial schedule, assigning one processor to each task, and iteratively searches for improvements by assigning more processors to the dominating task. The initial preemptive schedule is constructed by using McNaughton's wraparound rule. The algorithm then attempts to allocate more processors to the dominating task (the task with the largest processing time). It terminates if there are not enough unused (*idle*) processors or if the number of processors that can be assigned to the task is equal to the maximum number allowed, i.e., if $a_i = q_i$, for dominating task $i \in [w]$. Tasks that are assigned more than one processor are nonpreemptively scheduled on processors that are dedicated solely to them and are executed concurrently from start to completion; we call this a *parallel schedule.* A detailed description of the algorithm follows.

1. Let $T_{\max} = \max_{i \in [w]}\{t_i(a_i)\} = t_j(a_j)$ be the maximum among all task execution times (with task $j$ having this time), and let $T_{\text{avg}} = \sum_{i \in U_1} t_i(a_i)/\min(\textit{remain}, |U_1|)$ denote the average time of uniprocessor tasks using all remaining processors (processors not used by the parallel schedule). Initially, $\textit{remain} = n$ and $U_1$ consists of all tasks. The current schedule time $T_s$ is set to $T_s = \max\{T_{\max}, T_{\text{avg}}\}$. If $T_{\text{avg}} > T_{\max}$ then the algorithm terminates, sets $m = \textit{remain}$, and goes to step 4, else there is a dominating task, namely task $j$.

2. If dominating task $j$ is in $U_1$ then remove it from $U_1$ and remove one processor from *remain*, i.e., $U_1 = U_1 - \{j\}$ and $\textit{remain} = \textit{remain} - 1$. Determine the minimum number of processors, $m$, needed to preemptively schedule $U_1$ (using the wraparound rule) within time $T_s$ where $m = \lceil \sum_{i \in U_1} t_i(a_i)/T_s \rceil$.

3. Increase the number of processors assigned to the dominating task from $p_j(a_j)$ to $p_j(a_j + 1)$ if there are enough unused, i.e., *idle*, processors available where $\textit{idle} = \textit{remain} - m$. Update the number of remaining processors not used in the parallel schedule to $\textit{remain} = \textit{remain} - (p_j(a_j + 1) - p_j(a_j))$ and goto Step 1 (continue



7
6
5
4
processors
3
2
1
0
Task 5 (assigned 3 processors)
Task 4 (dominating task, assigned 2 processors)
Task 1
Task 2
Task 2
Task 3
m
remain
idle
time
$T_{max}$

FIG. 1. *A schedule generated by the approximation algorithm.*

iterating). Terminate the algorithm if we cannot increase the number of processors assigned to task $j$, i.e., if either $idle < p_j(a_j + 1) - p_j(a_j)$ or if $a_j = q_j$, and goto Step 4.

4. For each task $j$, where $j \notin U_1$, schedule $j$ on $p_j(a_j)$ processors from time 0 to $t_j(a_j)$. Using McNaughton's algorithm, preemptively schedule tasks in $U_1$ on $m$ processors.

Figure 1 shows an example of a typical schedule derived by the algorithm; the figure also illustrates the various parameters $(remain, idle, m)$. Note that the uniprocessor tasks are preemptively scheduled using McNaughton's wraparound rule.

In the approximation algorithm, the number of iterations is clearly no more than $n$. Furthermore, once the number of processors used for a task is increased, it is never decreased. Thus, the number of iterations is no more than the total number of processor-collection choices for all tasks—given by $\sum_{i\in[w]} q_i$. Therefore the number of iterations is at most $\min\{n, \sum_{i\in[w]} q_i\}$. At each iteration, the maximum among the execution times $t_i(a_i)$, for all $i \in [w]$, can be found in $O(\log w)$ time if these are stored in a priority queue. Finally, the preemptive scheduling of the tasks in $U_1$ takes time $O(w)$. Therefore, the approximation algorithm has complexity $O(\min\{n, \sum_{i\in[w]} q_i\} \log w)$. Further, since at most $n$ tasks are preemptively scheduled, the number of preemptions are bound by $n - 1$. From the construction of the algorithm, only uniprocessor tasks are preempted. Since preempting a task running on $k > 1$ processors is more expensive than preempting a task running on a single processor, the algorithm achieves the desirable side effect of keeping the preemption overhead small.

**4. Performance analysis of the approximation algorithm.** We now show that our scheduling algorithm derives a schedule whose completion time is within $r$ of the completion time of an optimal schedule. The proof utilizes the property that the processor-time product is nondecreasing for all tasks. It is shown that an optimal schedule must use at least as many processors for each task as the number assigned by our algorithm, and for a dominating task it must assign more processors. However, increasing the number of processors to the next allowable processor-collection size can only decrease the time by a factor of at most $r$. Also, increasing the number of processors for a task cannot decrease the processor-time product (area) in the schedule.

Notation.

- $S_a$, $T_a$ is the schedule produced by the approximation algorithm and its completion time.
- $S_o$, $T_o$ is some optimal schedule and its completion time.
- $S_f$, $T_f$ is some feasible schedule and its completion time.
- $a_i$ is the processor-collection choice for task $i$ in $S_a$.
- $o_i$ is the processor-collection choice for task $i$ in $S_o$.
- $f_i$ is the processor-collection choice for task $i$ in $S_f$.

Recall that $m$ denotes the number of processors on which tasks in the set $U_1$ (assigned to run on a single processor) are preemptively scheduled.

Given any feasible schedule $S_f$, Lemma 4.1, whose proof follows from definitions, gives a lower bound on the completion time $T_f$ in terms of the number of processors $p_i(a_i)$ assigned to each task $i$ and the corresponding execution times of the tasks in $S_f$. Lemma 4.2 proves that at least $m-1$ processors are fully utilized in the preemptive schedule. This follows directly from the properties of the wraparound rule in McNaughton's preemptive scheduling algorithm [8]. Finally, from Property (i), we have Proposition 4.3, which states that if a dominating task in $S_a$ has been allocated the maximum possible number of processors (i.e., $p_i(q_i)$), then $S_a$ is optimal.

LEMMA 4.1. *For all feasible schedules $S_f$, $T_f \geq \sum_{i=1}^{w} p_i(f_i)t_i(f_i)/n$.*

LEMMA 4.2. *For a schedule $S_a$ derived by the algorithm, $\sum_{i\in U_1} t_i(1) > (m-1)T_a$. In addition, if the algorithm terminates because $T_{\text{avg}} \geq T_{\max}$, then $\sum_{i\in U_1} t_i(1) = mT_a$.*

PROPOSITION 4.3. *Assume that in the schedule $S_a$, there exists some $k \in [w]$ such that task $k$ is a dominating task and $a_k = q_k$. Then $S_a$ is optimal.*

We now prove that the approximation algorithm produces a sequence of feasible schedules with nonincreasing completion times $T_s$ (schedule times).

LEMMA 4.4. *The sequence of completion times $T_s$ of schedules produced by the approximation algorithm is nonincreasing.*

*Proof.* The completion time $T_s$ at any iteration is given by $T_s = \max\{T_{\text{avg}}, T_{\max}\}$. To show that the sequence $T_s$ is nonincreasing, we need only show that $T_{\max}$ and $T_{\text{avg}}$ at any iteration are not larger than the value of $T_s$ at the previous iteration. Consider $T_{\text{avg}}$ at any iteration. The number of processors available to preemptively schedule tasks in $U_1$, i.e., the value of *remain* at the start of this iteration (in step 1) is no smaller than the value of $m$ (computed at step 2) at the end of the previous iteration, where $m = \lceil \sum_{i\in U_1} t_i(1)/T_s \rceil$ and the set $U_1$ is the same as that used to compute $m$ in step 2 of the previous iteration. Thus, $T_{\text{avg}}$ is no greater than the value of $T_s$ at the previous iteration. $T_{\max}$ is the maximum among all task times and the number of processors assigned to a task never decreases between iterations. Thus, it follows from Property (i) that $T_{\max}$ is nonincreasing. Since $T_{\text{avg}}$ and $T_{\max}$ are nonincreasing, $T_s$ is nonincreasing. $\square$

For all tasks $i \in [w]$, for any schedule $S_a$ produced by our algorithm we define the variable $a'_i$ to be $a'_i = a_i + 1$ if task $i$ is a dominating task in $S_a$ and $a'_i = a_i$ otherwise.

The variable $a'_i$ indicates the smallest choice of processor collection, i.e., $p_i(a'_i)$ is the minimum number of processors that must be assigned to each task $i$ to further decrease the schedule time from $T_a$. From Proposition 4.3, if $S_a$ is not optimal, then for all dominating tasks $k$ in $S_a$, we have $a_k < q_k$, and thus $a'_k$ is a valid choice for task $k$. The next lemma states that for any feasible schedule with a lower completion time than our algorithm, the number of processors assigned to each task in $S_f$ must be at least as large as the number assigned by our approximation algorithm. Further, all dominating tasks (in $S_a$) must be assigned more processors. In other words, for any such feasible schedule $S_f$ with $T_f < T_a$, for each task $i \in [w]$, we have $f_i \geq a'_i$ (and therefore $p_i(f_i) \geq p_i(a'_i)$).

LEMMA 4.5. *Assume that there exists some feasible schedule $S_f$ such that $T_f < T_a$. Then for all $i \in [w], a'_i \leq f_i$.*

*Proof.* The completion time of a schedule is no smaller than the execution time of any of its tasks. Thus, in the schedule $S_f$, for all $i \in [w]$, $t_i(f_i) \leq T_f$; further, since $T_f < T_a$ from the assumption of the lemma, we have the relation $t_i(f_i) \leq T_f < T_a$. We next analyze the number of processors allocated to the tasks in the schedule $S_a$. Let $i$ be an arbitrary task in $[w]$. There are two cases.

*Case* 1. Task $i$ is a dominating task in $S_a$ with $T_a = t_i(a_i)$. Thus, we have the relation $t_i(f_i) \leq T_f < T_a = t_i(a_i)$, implying that $f_i \geq a_i + 1$ from Property (i). From the definition of $a'_i$, $a'_i = a_i + 1$ and therefore $a'_i \leq f_i$.

*Case* 2. Task $i$ is not a dominating task in $S_a$. If $a_i = 1$, since $p_i(1)$ is the smallest processor-collection size for task $i$, we have $f_i \geq a_i$. If $a_i > 1$, the approximation algorithm has increased the number of processors for task $i$ in some iteration from $p_i(a_i - 1)$ to $p_i(a_i)$, implying that task $i$ was a dominating task in the schedule produced by the approximation algorithm at that iteration with completion time $t_i(a_i - 1)$. From Lemma 4.4, we have $T_a \leq t_i(a_i - 1)$, and this results in the relation $t_i(f_i) \leq T_f < T_a \leq t_i(a_i - 1)$. Therefore, from Property (i), we have $f_i > a_i - 1$ and $f_i \geq a_i$.

From Cases 1 and 2, it follows that for all $i \in [w]$, $f_i \geq a'_i$. □

For the case in which $S_a$ is not optimal, Theorem 4.6 derives a lower bound on the completion time of an optimal schedule in terms of $p_i(a'_i)$ and $t_i(a'_i)$ for all $i \in [w]$.

THEOREM 4.6. *Assume that $S_a$ is not optimal. Then $T_o \geq \sum_{i=1}^{w} p_i(a'_i)t_i(a'_i)/n$.*

*Proof.* Since an optimal schedule is also a feasible schedule, from Lemma 4.5, we have for all $i \in [w], o_i \geq a'_i$. Thus from Property (ii), we have for all $i \in [w]$, $p_i(o_i)t_i(o_i) \geq p_i(a'_i)t_i(a'_i)$; and therefore, $\sum_{i=1}^{w} p_i(o_i)t_i(o_i) \geq \sum_{i=1}^{w} p_i(a'_i)t_i(a'_i)$. Since $T_o \geq \sum_{i=1}^{w} p_i(o_i)t_i(o_i)/n$ from Lemma 4.1, the theorem follows. □

For the case in which $S_a$ is not optimal and there is a dominating task in $T_a$, Lemma 4.7 derives an upper bound on the ratio of $T_a$ to $T_o$. The proof follows from Theorem 4.6.

LEMMA 4.7. *Let $k \in [w]$ such that task $k$ is a dominating task in $T_a$. Assume that $S_a$ is not optimal. Then $T_a/T_o \leq n\ t_k(a_k)/\sum_{i=1}^{w} p_i(a'_i)t_i(a'_i)$.*

In the next theorem, we derive an upper bound of $r$ on the ratio $T_a/T_o$, by exploiting the fact that the processor-time product for each task in the optimal schedule is no less than the processor-time product for each task in the schedule derived by our algorithm.

THEOREM 4.8. *The approximation algorithm derives a schedule $S_a$ with $T_a/T_o \leq r$.*

*Proof.* If $S_a$ is optimal, the theorem trivially follows, since $r \geq 1$. Assume therefore that $S_a$ is not optimal. In the final schedule derived by the approximation algorithm, we have two possible cases: (1) the schedule $S_a$ has a dominating task and thus $T_a = T_{\max} \geq T_{\text{avg}}$; or (2) the schedule $S_a$ has no dominating task and thus $T_a = T_{\text{avg}} > T_{\max}$. Based on the schedule $S_a$ generated by the algorithm (when it terminates), we can partition the set of all tasks into three sets $U_1, B_1$, and $B_2$.

- $U_1$ consists of all nondominating tasks assigned to a single processor (as defined before); $U_1 = \{i \mid a_i = 1, t_i(a_i) < T_a\}$;
- $B_1$ contains all nondominating tasks that are assigned more than one processor: $B_1 = \{i \mid i \in [w],\ t_i(a_i) < T_a,\text{ and } a_i > 1\}$;
- $B_2 = \{i \mid i \in [w], t_i(a_i) = T_a\}$, which contains all the dominating tasks in $S_a$.

Let $m$ denote the total number of processors allocated to tasks in $U_1$ and $b$ denote the total number of processors allocated to tasks in $B_1$.

Thus, we have

$$(1)\qquad \sum_{i=1}^{w} p_i(a_i')t_i(a_i') = \sum_{i\in U_1} p_i(a_i')t_i(a_i') + \sum_{i\in B_1} p_i(a_i')t_i(a_i') + \sum_{i\in B_2} p_i(a_i')t_i(a_i').$$

First consider the term $\sum_{i\in B_1} p_i(a_i')t_i(a_i')$, which denotes the processor-time product of nondominating tasks running on more than one processor. From the definition of $a_i'$, for all $i \in B_1$ we have $a_i' = a_i$. From Property (ii), for all $i \in B_1$ we have $p_i(a_i)t_i(a_i) \geq p_i(a_i-1)t_i(a_i-1)$; and, from Lemma 4.4 and as shown in the proof of Case 2 of Lemma 4.5, we have $t_i(a_i-1) \geq T_a$. Further, since $p_i(a_i-1)/p_i(a_i) \geq 1/r$ from the definition of $r$, we have $t_i(a_i) \geq p_i(a_i-1)t_i(a_i-1)/p_i(a_i) \geq t_i(a_i-1)/r \geq T_a/r$ from Property (ii). Therefore,

$$(2)\qquad \sum_{i\in B_1} p_i(a_i')t_i(a_i') = \sum_{i\in B_1} p_i(a_i)t_i(a_i) \geq \frac{T_a}{r}\sum_{i\in B_1} p_i(a_i) = \frac{T_a}{r}b.$$

We now derive bounds on $\sum_{i\in[w]} p_i(a_i')t_i(a_i')$ for each of the two terminating conditions below.

*Case* 1. $T_a = T_{\max} \geq T_{\text{avg}}$, and for some dominating task $k \in B_2$ we have $T_a = t_k(a_k)$. From the definition of $a_i'$, for all $i \in U_1$ we have $a_i' = a_i = 1$, and from Lemma 4.2, we have $\sum_{i\in U_1} p_i(a_i)t_i(a_i) > (m-1)t_k(a_k)$. Therefore, since $r \geq 1$, we have

$$(3)\qquad \sum_{i\in U_1} p_i(a_i')t_i(a_i') = \sum_{i\in U_1} p_i(a_i)t_i(a_i) > (m-1)T_a \geq (m-1)T_a/r.$$

For all $i \in B_2$, we have $a_i' = a_i+1$ and $t_i(a_i) = t_k(a_k) = T_a$. From Property (ii), we have $t_i(a_i')p_i(a_i') \geq t_i(a_i)p_i(a_i)$ and therefore $t_i(a_i') \geq t_k(a_k)/r = T_a/r$. Consequently, we have

$$\sum_{i\in B_2} t_i(a_i')p_i(a_i') \geq \frac{T_a}{r}\sum_{i\in B_2} p_i(a_i').$$

Furthermore, we must have $\sum_{i\in B_2} p_i(a_i') > (n-m-b) \geq (n-m-b+1)$; otherwise, there would have been enough processors for the algorithm to increase the number of processors assigned to each dominating task to its next allowable size (which is $p_i(a_i')$, for all $i \in B_2$).

Therefore, we now have

$$\sum_{i\in B_2} t_i(a_i')p_i(a_i') \geq \frac{t_k(a_k)}{r}(n-m-b+1). \tag{4}$$

From equations (1)–(4), we have

$$\sum_{i=1}^{w} p_i(a_i')t_i(a_i') \geq n\frac{T_a}{r} \text{ for } r \geq 1,$$

from which we have $(nT_a)/\left(\sum_{i=1}^{w} p_i(a_i')t_i(a_i')\right) \leq r$, and from Lemma 4.7, we have $T_a/T_o \leq r$.

*Case* 2. $T_a = T_{\text{avg}} > T_{\max}$ and there is no dominating task in the schedule. For this case, the sets $U_1$ and $B_1$ comprise the set of all tasks since $B_2$ is empty (and $U_1$ is not empty). In this schedule, there are no idle processors (since all processors in *remain* are used to schedule tasks in $U_1$) and so we have $m + b = n$.

For all $i \in U_1$, we have $a_i' = a_i = 1$, and since the algorithm terminates because $T_{\text{avg}} \geq T_{\max}$, from Lemma 4.2 we have $\sum_{i\in U_1} p_i(a_i)t_i(a_i) = mT_a$. Therefore,

$$\sum_{i\in U_1} p_i(a_i')t_i(a_i') = \sum_{i\in U_1} t_i(1) = mT_a = (n-b)T_a \geq (n-b)T_a/r. \tag{5}$$

From equations (1), (2), and (5), we have

$$\sum_{i=1}^{w} p_i(a_i')t_i(a_i') \geq \frac{nT_a}{r}.$$

From Theorem 4.6 we have $T_o \geq \sum_{i=1}^{w} p_i(a_i')t_i(a_i')/n$. Substituting into the above equation, it follows that $T_a/T_o \leq r$.

To show that the upper bound is asymptotically tight, we construct a problem instance in which $w = 2$, and $r$ an integer such that $n/r$ is an integer. Both tasks have perfect speedup and can be assigned at most $n$ processors. In addition, for task 2, there are two successive processor-collection choices of sizes $n/r$ and $n$. Task 1 has very small execution times compared to task 2. Thus, $p_1(1) = 1$, $t_1(1) = \epsilon$ ($\epsilon << 1$), $p_1(q_1) = n$, and $t_1(q_1) = \epsilon/n$. For task 2, $q_2 \geq 4$; $p_2(1) = 1$, $t_2(1) = 1$, $p_2(l) = n/r$, $t_2(l) = r/n$, $p_2(q_2) = n$, and $t_2(q_2) = 1/n$. The algorithm generates a parallel schedule where task 1 is allocated one processor, and task 2 is allocated a collection of $n/r$ processors, and the completion time $T_a$ is $r/n$. An optimal schedule is a schedule in which both tasks are allocated $n$ processors and executed in sequence with a completion time of $T_o = (\epsilon+1)/n$, which is less than $r/n$. Thus, $T_a/T_o = (r/n)/((\epsilon+1)/n)$ which for $n \to \infty, \epsilon << 1$ gives $T_a/T_o \to r$. $\square$

**5. Conclusion.** This paper presented an $r$-approximation algorithm for preemptively scheduling a set of $w$ independent tasks on a parallel-task system consisting of $n$ processors, where $r$ is the maximum factor by which we can increase the number of processors used to execute a task. Future efforts must focus on approximation algorithms for the problems of preemptive and nonpreemptive scheduling of a set of tasks with precedence constraints, which have been shown to be strongly NP-hard [3].

**Acknowledgments.** The authors are grateful to the anonymous referees for numerous suggestions that have helped improve the presentation.

## REFERENCES

[1] J. Blazewicz, M. Drabowski, and J. Welgarz, *Scheduling multiprocessor tasks to minimize schedule length*, IEEE Trans. Comput., C-35 (1986), pp. 389–393.

[2] G. I. Chen and T. H. Lai, *Preemptive scheduling of independent jobs on hypercubes*, Inform. Process. Lett., 28 (1988), pp. 201–206.

[3] J. Du and J. Y-T. Leung, *Complexity of scheduling parallel task systems*, SIAM J. Discrete Math., 2 (1989), pp. 473–487.

[4] M. R. Garey and D. S. Johnson, *Bounds for multiprocessor scheduling with resource constraints*, SIAM J. Comput., 4 (1975), pp. 187–200.

[5] ———, *Complexity results for multiprocessor scheduling under resource constraints*, SIAM J. Comput., 4 (1975), pp. 397–411.

[6] J. P. Hayes, T. N. Mudge, Q. F. Stout, and S. Colley, *Architecture of a hypercube supercomputer*, Proc. International Conference on Parallel Processing, IEEE Computer Society Press, Washington, DC, 1986, pp. 653–660.

[7] R. Krishnamurti and E. Ma, *An approximation algorithm for scheduling tasks on varying partition sizes in partitionable multiprocessor systems*, IEEE Trans. Comput., 41 (1992), pp. 1572–1579.

[8] R. McNaughton, *Scheduling with deadlines and loss functions*, Management Science, 6 (1959), pp. 1–12.

[9] B. Veltman, B. J. Lageweg, and J. K. Lenstra, *Multiprocessor scheduling with communication delays,* Parallel Comput., 16 (1990), pp. 173–182.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 670–677, November 1995

# ALTSHULER'S SPHERE $M^9_{963}$ REVISITED*

JÜRGEN BOKOWSKI† AND PETER SCHUCHERT†

**Abstract.** We show that a minimal nonpolytopal matroid polytope, the unique oriented matroid $\mathcal{M}$, characterized by requiring its Las Vergnas face lattice to be isomorphic to Altshuler's 3-sphere $M^9_{963}$, has no polar $\mathcal{M}^\Delta$, i.e., there is no oriented matroid $\mathcal{M}^\Delta$ such that its Las Vergnas lattice $\mathcal{F}_{\text{lv}}(\mathcal{M}^\Delta)$ equals the opposite (antiisomorphic or order-dual) lattice $\mathcal{F}_{\text{lv}}(\mathcal{M})^{op}$. This provides the minimal rank in which polarity for oriented matroids, i.e., $\mathcal{F}_{\text{lv}}(\mathcal{M}^\Delta) = \mathcal{F}_{\text{lv}}(\mathcal{M})^{op}$, fails to hold, thus answering a problem posed by Billera.

**Key words.** oriented matroid, matroid polytope, polarity

**AMS subject classifications.** 52C07, 52B40

## 1. Introduction.

During the last twenty years, it has been proved that the theory of convex polytopes has benefited very much from the concepts developed within the theory of oriented matroids. We often study convex polytopes within this more general framework. Many theorems and concepts that are fundamental in the theory of convex polytopes have been carried over, and they have been shown to hold in the more general case of matroid polytopes, i.e., oriented matroids with only extreme points. Moreover, very often they simplify the insight in the combinatorial structure. See [4] and [5].

On the other hand, the concept of polarity for convex polytopes does not carry over to matroid polytopes in its full generality. The convex hull of the vertices of a polytope and the intersection of the supporting half-spaces of a polytope have different generalizations in oriented matroid theory. Billera and Munson have proved the following theorem, see [3] and [4, Cor. 9.3.10].

THEOREM 1.1 (Billera and Munson, 1984). *There exist rank-12 matroid polytopes with 16 vertices that do not have polars.*

Within their proof, Billera and Munson used the Lawrence construction, which turns out to be inapplicable in cases of lower rank. Billera has asked whether a corresponding result can be obtained in cases of lower rank. We find this also as an open problem in [4, Exercise 9.12*]:

PROBLEM 1.2. *What is the smallest rank of a matroid polytope without a polar?*

In this article, we are going to solve this problem. The decisive example will be a matroid polytope $\mathcal{M}$ that has as its Las Vergnas lattice $\mathcal{F}_{\text{lv}}(\mathcal{M})$ Altshuler's sphere $M^9_{963}$, see [1]. We have defined this example in the next section, and we have listed several facts which are relevant in oriented matroid theory. Finally, in §3, we formulate our main result as Theorem 3.1. A nonpolytopal matroid polytope is an oriented matroid $\mathcal{M}$ with only extreme points such that there is no realizable polytope with face lattice isomorphic to the Las Vergnas face lattice $\mathcal{F}_{\text{lv}}(\mathcal{M})$ of $\mathcal{M}$.

## 2. The smallest uniform nonpolytopal matroid polytope $\mathcal{M}$.

### 2.1. Description of the matroid polytope $\mathcal{M}$.

We start by listing the facets of the 3-sphere denoted in Altshuler's list by $M^9_{963}$, compare [1].

* Received by the editors June 25, 1993; accepted for publication (in revised form) November 11, 1994.

† University of Darmstadt, Mathematics Department, Schloßgartenstraße 7, D-64289 Darmstadt, Germany (bokowski@mathematik.th-darmstadt.de, schuchert@mathematik.th-darmstadt.de).

1235 2137 1256 1268 2179 1289 3156 3168 1378 7189 3245 2347 2456
2467 6279 2689 4359 3478 3489 5369 6389 4567 4579 4789 5679

The sphere has 9 vertices $\{1, 2, \ldots, 9\}$ and 25 simplicial facets. It has the symmetry group $\Sigma_{\mathcal{M}}$ defined by $(1,5)(4,8)(2,6)(7,9)(3)$. We relabel the vertices in the above order to obtain the symmetry in the form $(1,2)(3,4)(5,6)(7,8)(9)$. We denote the (oriented) facets of the sphere, ordered according to the symmetry, by letters $a, b, \ldots, y$ as follows:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $a =$ | $+1256$ | | | $j =$ | $-1478$ | $k =$ | $+2378$ | $s =$ | $-3479$ | $t =$ | $+3489$ |
| $b =$ | $-1259$ | $c =$ | $+1269$ | $l =$ | $+1479$ | $m =$ | $+2389$ | $u =$ | $+3567$ | $v =$ | $-4568$ |
| $d =$ | $-1579$ | $e =$ | $-2689$ | $n =$ | $+1458$ | $o =$ | $+2367$ | $w =$ | $+3579$ | $x =$ | $+4689$ |
| $f =$ | $-1456$ | $g =$ | $+2356$ | $p =$ | $+1578$ | $q =$ | $-2678$ | $y =$ | $+5678.$ | | |
| $h =$ | $-1469$ | $i =$ | $-2359$ | $r =$ | $+3478$ | | | | | | |

Now we assume that there exists an oriented matroid $\mathcal{M}$ of rank 5, our matroid polytope $\mathcal{M}$, such that its Las Vergnas lattice $\mathcal{F}_{\text{lv}}(\mathcal{M})$ coincides with the above Altshuler sphere. We are going to determine signs of bases of $\mathcal{M}$ derived from the face-lattice structure. The sphere turned out to be a rigid one when the corresponding chirotope was determined in the investigation of the first author in 1978. The argument of Sturmfels for proving the rigidity of this matroid polytope in [4, p. 405] is false. The result of Shemer, carried over to the oriented matroid setting by Sturmfels, cannot be applied because the sphere is not neighborly (it has two missing edges : 1,3 and 2,4).

We list all (consistently oriented) partial hyperline sequences defined by subfacets in Table 1, below; compare for this notation [5]. See also the later geometric explanation.

TABLE 1

| subfacet | hyperline sequence | subfacet | hyperline sequence |
|---|---|---|---|
| $a \cap b$ | $(+1,2,5 \mid \{6\},\ldots,\{9\})$ | $a \cap c$ | $(-1,2,6 \mid \{5\},\ldots,\{9\})$ |
| $b \cap c$ | $(+1,2,9 \mid \{5\},\ldots,\{6\})$ | | |
| $f \cap n$ | $(-1,4,5 \mid \{6\},\ldots,\{8\})$ | $g \cap o$ | $(-2,3,6 \mid \{5\},\ldots,\{7\})$ |
| $f \cap h$ | $(+1,4,6 \mid \{5\},\ldots,\{9\})$ | $g \cap i$ | $(+2,3,5 \mid \{6\},\ldots,\{9\})$ |
| $j \cap l$ | $(-1,4,7 \mid \{8\},\ldots,\{9\})$ | $k \cap m$ | $(-2,3,8 \mid \{7\},\ldots,\{9\})$ |
| $n \cap j$ | $(-1,4,8 \mid \{5\},\ldots,\{7\})$ | $o \cap k$ | $(-2,3,7 \mid \{6\},\ldots,\{8\})$ |
| $h \cap l$ | $(+1,4,9 \mid \{6\},\ldots,\{7\})$ | $i \cap m$ | $(+2,3,9 \mid \{5\},\ldots,\{8\})$ |
| $a \cap f$ | $(+1,5,6 \mid \{2\},\ldots,\{4\})$ | $a \cap g$ | $(-2,5,6 \mid \{1\},\ldots,\{3\})$ |
| $p \cap d$ | $(+1,5,7 \mid \{8\},\ldots,\{9\})$ | $q \cap e$ | $(+2,6,8 \mid \{7\},\ldots,\{9\})$ |
| $n \cap p$ | $(+1,5,8 \mid \{4\},\ldots,\{7\})$ | $o \cap q$ | $(+2,6,7 \mid \{3\},\ldots,\{8\})$ |
| $b \cap d$ | $(-1,5,9 \mid \{2\},\ldots,\{7\})$ | $c \cap e$ | $(-2,6,9 \mid \{1\},\ldots,\{8\})$ |
| $c \cap h$ | $(+1,6,9 \mid \{2\},\ldots,\{4\})$ | $b \cap i$ | $(+2,5,9 \mid \{1\},\ldots,\{3\})$ |
| $j \cap p$ | $(-1,7,8 \mid \{4\},\ldots,\{5\})$ | $k \cap q$ | $(+2,7,8 \mid \{3\},\ldots,\{6\})$ |
| $l \cap d$ | $(+1,7,9 \mid \{4\},\ldots,\{5\})$ | $m \cap e$ | $(+2,8,9 \mid \{3\},\ldots,\{6\})$ |
| $r \cap s$ | $(+3,4,7 \mid \{8\},\ldots,\{9\})$ | $r \cap t$ | $(-3,4,8 \mid \{7\},\ldots,\{9\})$ |
| $s \cap t$ | $(+3,4,9 \mid \{7\},\ldots,\{8\})$ | | |
| $g \cap u$ | $(-3,5,6 \mid \{2\},\ldots,\{7\})$ | $f \cap v$ | $(+4,5,6 \mid \{1\},\ldots,\{8\})$ |
| $u \cap w$ | $(-3,5,7 \mid \{6\},\ldots,\{9\})$ | $v \cap x$ | $(-4,6,8 \mid \{5\},\ldots,\{9\})$ |
| $i \cap w$ | $(+3,5,9 \mid \{2\},\ldots,\{7\})$ | $h \cap x$ | $(+4,6,9 \mid \{1\},\ldots,\{8\})$ |
| $o \cap u$ | $(-3,6,7 \mid \{2\},\ldots,\{5\})$ | $n \cap v$ | $(-4,5,8 \mid \{1\},\ldots,\{6\})$ |
| $k \cap r$ | $(-3,7,8 \mid \{2\},\ldots,\{4\})$ | $j \cap r$ | $(+4,7,8 \mid \{1\},\ldots,\{3\})$ |
| $s \cap w$ | $(-3,7,9 \mid \{4\},\ldots,\{5\})$ | $t \cap x$ | $(-4,8,9 \mid \{3\},\ldots,\{6\})$ |
| $m \cap t$ | $(-3,8,9 \mid \{2\},\ldots,\{4\})$ | $l \cap s$ | $(-4,7,9 \mid \{1\},\ldots,\{3\})$ |
| $u \cap y$ | $(-5,6,7 \mid \{3\},\ldots,\{8\})$ | $v \cap y$ | $(+5,6,8 \mid \{4\},\ldots,\{7\})$ |
| $p \cap y$ | $(-5,7,8 \mid \{1\},\ldots,\{6\})$ | $q \cap y$ | $(+6,7,8 \mid \{2\},\ldots,\{5\})$ |
| $d \cap w$ | $(+5,7,9 \mid \{1\},\ldots,\{3\})$ | $e \cap x$ | $(+6,8,9 \mid \{2\},\ldots,\{4\})$ |

The following signs of bases of the chirotope are determined directly from the above hyperlines (we have used the alternating rules for brackets).

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 12356 | + | 12456 | + | 13478 | + | 23478 | + | 14689 | + | 23579 | + |
| 12359 | − | 12469 | + | 13479 | − | 23489 | + | 14789 | − | 23789 | + |
| 12367 | + | 12458 | − | 13489 | + | 23479 | − | 15678 | + | 25678 | + |
| 12369 | + | 12459 | − | 13567 | + | 24568 | − | 15679 | − | 25689 | + |
| 12378 | + | 12478 | + | 13578 | − | 24678 | + | 15789 | + | 26789 | − |
| 12389 | + | 12479 | − | 13579 | + | 24689 | + | 34567 | − | 34568 | − |
| 12567 | + | 12568 | + | 14567 | − | 23568 | + | 34578 | + | 34678 | + |
| 12569 | + | | | 14568 | − | 23567 | + | 34579 | − | 34689 | + |
| 12578 | − | 12678 | − | 14569 | − | 23569 | + | 34589 | + | 34679 | − |
| 12579 | + | 12689 | − | 14578 | − | 23678 | + | 34789 | + | | |
| 12589 | + | 12679 | − | 14579 | + | 23689 | + | 35678 | + | 45678 | + |
| 13456 | + | 23456 | + | 14589 | + | 23679 | + | 35679 | + | 45689 | − |
| 13458 | − | 23467 | + | 14678 | − | 23578 | + | 35789 | − | 46789 | + |
| 13469 | + | 23459 | − | 14679 | + | 23589 | + | 56789 | + | | |

For the remaining signs we look at rank-2 contractions of this potential partial chirotope of $\mathcal{M}$, and we try to determine other signs of oriented bases by using the chirotope axioms. The following sequence of chirotope axioms determines all remaining signs of bases of the chirotope $\mathcal{M}$ (and its symmetric images).

| | | | |
|---|---|---|---|
| $\{578\mid 2634\} \Rightarrow [24578] := +,$ | $([13678] := -)$ | $\{234\mid 1789\} \Rightarrow [12349] := +$ | |
| $\{258\mid 4736\} \Rightarrow [23458] := +,$ | $([13467] := -)$ | $\{137\mid 2469\} \Rightarrow [12379] := +,$ | $([12489] := -)$ |
| $\{358\mid 2469\} \Rightarrow [35689] := +,$ | $([45679] := -)$ | $\{379\mid 1248\} \Rightarrow [13789] := +,$ | $([24789] := -)$ |
| $\{589\mid 3627\} \Rightarrow [25789] := -,$ | $([16789] := +)$ | $\{789\mid 1325\} \Rightarrow [12789] := +$ | |
| $\{789\mid 2536\} \Rightarrow [36789] := -,$ | $([45789] := +)$ | $\{128\mid 7946\} \Rightarrow [12468] := -,$ | $([12357] := +)$ |
| $\{689\mid 3712\} \Rightarrow [13689] := -,$ | $([24579] := -)$ | $\{148\mid 2635\} \Rightarrow [13468] := -,$ | $([23457] := +)$ |
| $\{369\mid 1827\} \Rightarrow [13679] := -,$ | $([24589] := -)$ | $\{134\mid 6857\} \Rightarrow [13457] := -,$ | $([23468] := +)$ |
| $\{679\mid 1324\} \Rightarrow [24679] := -,$ | $([13589] := -)$ | $\{457\mid 1326\} \Rightarrow [12457] := -,$ | $([12368] := +)$ |
| $\{679\mid 2415\} \Rightarrow [25679] := +,$ | $([15689] := -)$ | $\{245\mid 1736\} \Rightarrow [12345] := -,$ | $([12346] := -)$ |
| $\{567\mid 2934\} \Rightarrow [24567] := -,$ | $([13568] := +)$ | $\{345\mid 1289\} \Rightarrow [13459] := +,$ | $([23469] := -)$ |
| $\{467\mid 2513\} \Rightarrow [12467] := -,$ | $([12358] := +)$ | $\{459\mid 1367\} \Rightarrow [34569] := +$ | |
| $\{247\mid 1638\} \Rightarrow [12347] := -,$ | $([12348] := -)$ | $\{469\mid 3512\} \Rightarrow [24569] := +,$ | $([13569] := -)$ |

We still have to show that the chirotope axioms are fulfilled for the set of signed bases. Starting with these signs, we determine and complete all hyperline sequences of $\mathcal{M}$. In order to explain once more the idea of these hyperline sequences, which become abstract in the oriented matroid case, we assume the realizable case, and we assume all 9 points to lie in general position. Any 3 ordered points affinely span an oriented 2-flat. We consider the sequence of oriented hyperplanes containing this oriented 2-flat with outer normal vector rotating in the corresponding (oriented) orthogonal complement of the oriented 2-flat. This induces a cyclic order of the remaining 6 points. As an oriented hyperline sequence, we write down the oriented 2-flat together with the induced cyclic order of the remaining points. The induced cyclic order of the points must be compatible with the facet structure. It has turned out in our case that there is only one chirotope compatible with the boundary structure, i.e., the sphere is rigid.

We now present the chirotope as a complete list of all (consistent) oriented hyperline sequences of $\mathcal{M}$.

| | |
|---|---|
| ( 1,2,3 \| 4 -5 9 -6 -7 -8) | ( 1,2,4 \| 3 7 8 5 -9 6) |
| ( 1,2,5 \| 3 **9** **-6** -4 -8 -7) | ( 1,2,6 \| 3 **5** **-9** -4 -8 -7) |
| ( 1,2,7 \| 3 -8 -4 5 -9 6) | ( 1,2,8 \| 3 7 -4 5 -9 6) |
| ( 1,2,9 \| 3 7 8 4 **6** **-5**) | |
| ( 1,3,4 \| 2 -7 9 -8 -5 -6) | ( 2,3,4 \| 1 5 6 7 -9 8) |
| ( 1,3,5 \| 2 6 4 8 -9 7) | ( 2,4,6 \| 1 5 3 7 -9 8) |
| ( 1,3,6 \| 2 9 -5 4 8 7) | ( 2,4,5 \| 1 9 -6 3 7 8) |
| ( 1,3,7 \| 2 8 4 -5 9 -6) | ( 2,4,8 \| 1 7 3 -6 9 -5) |
| ( 1,3,8 \| 2 -7 9 4 -5 -6) | ( 2,4,7 \| 1 -8 9 3 -6 -5) |
| ( 1,3,9 \| 2 -7 5 -4 -8 -6) | ( 2,4,9 \| 1 -8 6 -3 -7 -5) |
| ( 1,4,5 \| 2 6 -8 -7 -3 -9) | ( 2,3,6 \| 1 5 -7 -8 -4 -9) |
| ( 1,4,6 \| 2 9 -5 -8 -7 -3) | ( 2,3,5 \| 1 9 -6 -7 -8 -4) |
| ( 1,4,7 \| 2 6 5 8 -9 -3) | ( 2,3,8 \| 1 5 6 7 -9 -4) |
| ( 1,4,8 \| 2 6 5 -7 -3 -9) | ( 2,3,7 \| 1 5 6 -8 -4 -9) |
| ( 1,4,9 \| 2 5 8 3 7 -6) | ( 2,3,9 \| 1 6 7 4 8 -5) |
| ( 1,5,6 \| **2** **9** 3 7 8 4) | ( 2,5,6 \| **1** -3 -7 -8 -4 **-9**) |
| ( 1,5,7 \| 2 3 9 -8 -4 -6) | ( 2,6,8 \| 1 4 9 -7 -3 -5) |
| ( 1,5,8 \| 2 9 3 7 -4 -6) | ( 2,6,7 \| 1 9 4 8 -3 -5) |
| ( 1,5,9 \| **2** -7 -3 -8 -4 **-6**) | ( 2,6,9 \| **1** -8 -4 -7 -3 **-5**) |
| ( 1,6,7 \| 2 3 -8 -4 5 -9) | ( 2,5,8 \| 1 4 -7 -3 6 -9) |
| ( 1,6,8 \| 2 3 7 -4 5 -9) | ( 2,5,7 \| 1 4 8 -3 6 -9) |
| ( 1,6,9 \| **2** **5** 3 7 8 4) | ( 2,5,9 \| **1** **6** 4 8 7 3) |
| ( 1,7,8 \| 2 3 9 4 -5 -6) | ( 2,7,8 \| 1 5 6 -3 -9 -4) |
| ( 1,7,9 \| 2 3 5 -4 -8 -6) | ( 2,8,9 \| 1 4 6 -3 -7 -5) |
| ( 1,8,9 \| 2 5 -4 3 7 -6) | ( 2,7,9 \| 1 6 -3 4 8 -5) |
| ( 3,4,5 \| 1 -6 -2 7 -9 8) | ( 3,4,6 \| 1 5 -2 7 -9 8) |
| ( 3,4,7 \| 1 9 -8 -2 -6 -5) | ( 3,4,8 \| 1 7 -9 -2 -6 -5) |
| ( 3,4,9 \| 1 5 6 2 8 -7) | |
| ( 3,5,6 \| 1 9 2 -7 -8 -4) | ( 4,5,6 \| 1 9 2 3 7 8) |
| ( 3,5,7 \| 1 4 8 2 6 -9) | ( 4,6,8 \| 1 5 -9 -2 -3 -7) |
| ( 3,5,8 \| 1 4 9 -7 2 6) | ( 4,6,7 \| 1 5 -2 -3 -9 8) |
| ( 3,5,9 \| 1 7 -2 -6 -8 -4) | ( 4,6,9 \| 1 5 7 3 2 8) |
| ( 3,6,7 \| 1 9 4 8 2 -5) | ( 4,5,8 \| 1 -6 -2 -9 -3 -7) |
| ( 3,6,8 \| 1 4 9 -7 2 -5) | ( 4,5,7 \| 1 -6 -2 -3 -9 8) |
| ( 3,6,9 \| 1 5 -8 -4 -7 2) | ( 4,5,9 \| 1 -2 -6 7 3 8) |
| ( 3,7,8 \| 1 5 6 2 -4 -9) | ( 4,7,8 \| 1 5 6 2 9 3) |
| ( 3,7,9 \| 1 6 2 8 4 -5) | ( 4,8,9 \| 1 7 3 -6 -2 -5) |
| ( 3,8,9 \| 1 5 6 2 -4 -7) | ( 4,7,9 \| 1 -3 -8 -2 -6 -5) |
| ( 5,6,7 \| 1 9 2 3 -8 -4) | ( 5,6,8 \| 1 9 2 3 7 -4) |
| ( 5,6,9 \| **1** -3 -7 -8 -4 **2**) | |
| ( 5,7,8 \| 1 -6 -2 -3 -9 -4) | ( 6,7,8 \| 1 5 -2 -3 -9 -4) |
| ( 5,7,9 \| 1 4 8 6 2 3) | ( 6,8,9 \| 1 4 -2 -3 -7 -5) |
| ( 5,8,9 \| 1 4 -3 -7 6 2) | ( 6,7,9 \| 1 -2 -3 4 8 -5) |
| ( 7,8,9 \| 1 5 6 2 -4 3) | |

By checking that the sign of each basis (= bracket) derived from different hyperline sequences is always the same, we have verified the chirotope axioms for $\mathcal{M}$. We do this for one example ([12569]) out of all $\binom{9}{5} = 126$ brackets. $\text{sign}[1,2,5,9,-6] = \text{sign}[1,2,6,5,-9] = \text{sign}[1,2,9,6,-5] = \cdots = \text{sign}[5,9,6,1,2]$.

The fact that the boldface pairs are always adjacent with respect to its cyclic order tells us that we have considered a mutation. For other aspects in oriented matroid

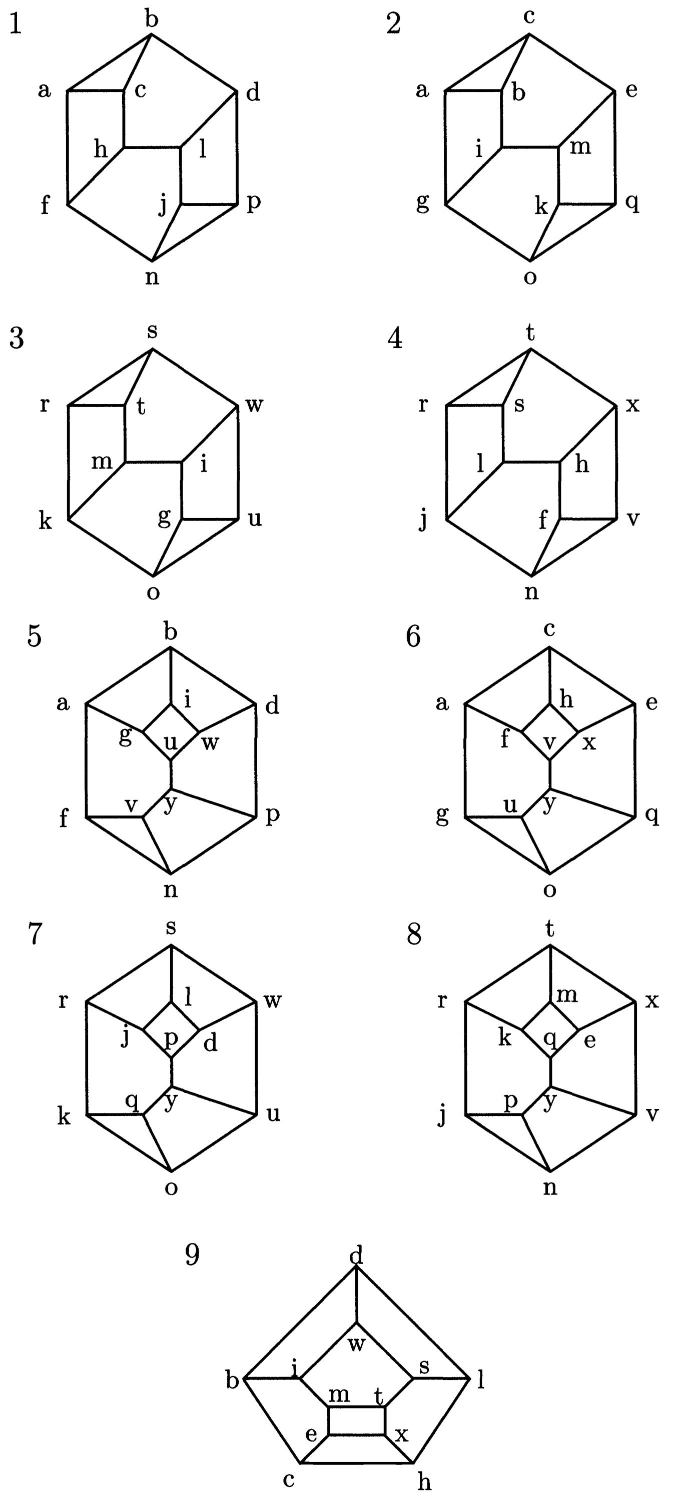


FIG. 1. *Schlegel diagrams of the facets of the decisive tope in the Folkman–Lawrence representation of $\mathcal{M}$.*

theory, we list all 8 mutations of $\mathcal{M}$: [12569]; [13579]; [14568]; [14578]; [23567]; [23678]; [24689]; and [34789]. The fact that we have only 8 mutations allows us to replace the short proof for nonrealizability, using the general method of providing a biquadratic final polynomial (as given in [1]), with the more complicated proof of [8] as mentioned in [7].

**2.2. Properties of $\mathcal{M}$.** Our sphere $M^9_{963}$ is rigid. This was shown in the last section. The matroid polytope $\mathcal{M}$ is symmetric because all signs of the bases are determined by the sphere, and the sphere itself is symmetric. The fact that $\mathcal{M}$ is not polytopal was shown in [1].

This oriented matroid $\mathcal{M}$ has a topological representation (its Folkman–Lawrence representation), and we consider therein the maximal tope (topological 4-ball bounded by 9 topological 2-spheres forming the facets of this tope) with the face lattice opposite to the face lattice of Altshuler's sphere.

Figure 1 depicts the 9 Schlegel diagrams of these facets of this tope in the Folkman–Lawrence representation of $\mathcal{M}$, which also defines once more Altshuler's sphere $M^9_{963}$ by order duality. These Schlegel diagrams will be used later in Theorem 3.1.

Starting with this example of a matroid polytope, one can construct an infinite family of nonpolytopal matroid polytopes by applying duality and successive lexicographic extensions; see Proposition 9.5.5 in [4].

The matroid polytope $\mathcal{M}$ is a minimal nonpolytopal one with respect to the rank. It is also minimal with respect to the number of vertices among all uniform matroid polytopes. This result was checked in 1981 by the first author (unpublished) but it has been documented in [2], a source which was not cited in [4].

The oriented matroid of rank 4 with 8 elements derived from our matroid polytope $\mathcal{M}$ by contracting point 9, or $AB(9)/3$ when using the original Altshuler labeling, has interesting properties. It defines the *only* reorientation class of oriented matroids with 7 mutations among all reorientation classes of oriented matroids of rank 4 with 8 points [6]. For a symmetric Folkman–Lawrence representation of this example, see also the Bielefeld model as described in [5]. See also [4] for additional applications.

We deal with an additional decisive property in the next section.

## 3. The matroid polytope $\mathcal{M}$ has no polar.

We are going to show that the minimal nonpolytopal matroid polytope from the last section, the unique oriented matroid $\mathcal{M}$ characterized by requiring its Las Vergnas face lattice to be isomorphic to Altshuler's 3-sphere $M^9_{963}$, has no polar $\mathcal{M}^\Delta$; i.e., there is no oriented matroid $\mathcal{M}^\Delta$ such that its Las Vergnas lattice $\mathcal{F}_{\mathrm{lv}}(\mathcal{M}^\Delta)$ equals the opposite (antiisomorphic or order-dual) lattice $\mathcal{F}_{\mathrm{lv}}(\mathcal{M})^{op}$.

This provides the minimal rank in which polarity for oriented matroids, i.e., $\mathcal{F}_{\mathrm{lv}}(\mathcal{M}^\Delta) = \mathcal{F}_{\mathrm{lv}}(\mathcal{M})^{op}$, fails to hold, thus answering a problem posed by Billera.

THEOREM 3.1. *The smallest rank of a matroid polytope without a polar is* 5.

*Proof.* In principle, one can proceed as we did in the case of $\mathcal{M}$. But now we have 25 vertices, and the complexity of the algorithms involved has grown tremendously. In this case, our oriented matroid (we call it $\mathcal{M}^\Delta$) would be no longer uniform. Simple calculations show that the problem of using all chirotope axioms in this case is far beyond what can easily be computed in reasonable time. Nevertheless, the attempt was made to proceed in a fashion similar to the case of $\mathcal{M}$, by using more sophisticated implementations. Fortunately, it turned out that, in our final proof, everything can easily be checked without using a computer.

The proof uses only chirotope axioms, i.e., sign conditions derived from Grassmann–Plücker relations. $\mathcal{M}^\Delta$ denotes our matroid polytope, which we assume to exist. We consider the rank-2 minors of $\mathcal{M}^\Delta$ with respect to the following subfacets: $(1\cap 5)$; $(1\cap 9)$; $(2\cap 6)$; $(3\cap 7)$; $(4\cap 7)$; $(4\cap 8)$; and $(6\cap 7)$.

We use the notation for hyperline sequences as before (now adjusted to the nonuniform case); see Table 2 below.

TABLE 2

| subfacet | ( hyperline \| | left | mid | right ) |
|---|---|---|---|---|
| $1\cap 5$ | (+a,b,d,p,n,f \| | {c,h,j,l}, | ...ekmoqrstx ..., | {g,i,u,v,w,y} ) |
| $1\cap 9$ | ( +b,c,h,l,d \| | {a,f,j,n,p}, | ...gkoqruvy ..., | {e,i,m,s,t,w,x} ) |
| $2\cap 6$ | (+a,c,e,q,o,g \| | {b,i,k,m}, | ...djlnprstw ..., | {f,h,u,v,x,y} ) |
| $3\cap 7$ | (+k,r,s,w,u,o \| | {g,i,m,t}, | ...abcefhnvx ..., | {d,j,l,p,q,y} ) |
| $4\cap 7$ | ( +j,l,s,r \| | {f,h,n,t,v,x}, | ...abcegim ..., | {d,k,o,p,q,u,w,y}) |
| $4\cap 8$ | ( +j,r,t,x,v,n \| | {f,h,l,s}, | ...abcdgiouw ..., | {e,k,m,p,q,y} ) |
| $6\cap 7$ | ( +o,u,y,q \| | {a,c,e,f,g,h,v,x}, | ...bimnt ..., | {d,j,k,l,p,r,s,w} ) |

The 7 hyperlines are connected (or consistently oriented), which can be checked by testing the signs of the following brackets in corresponding different lines above: $\text{sign}[bdace] = -$; $\text{sign}[bdlsr] = -$; $\text{sign}[bauoq] = -$; and $\text{sign}[njskr] = -$.

We use the contraction $\mathcal{M}^\Delta/\{b,d\}$ of our assumed chirotope $\mathcal{M}^\Delta$, and we use the chirotope axioms to determine more signs of brackets as follows. We have indicated on top of each bracket why we know its sign from earlier observations. The sign itself is given below.

$$\{j|asxr\} = \underbrace{\overbrace{[jas]}^{1\cap5}}_{-}\cdot\underbrace{\overbrace{[jxr]}^{\text{use }4\cap8\text{ and }[bdjnr],\text{by }1\cap5}}_{-} - \underbrace{\overbrace{[jax]}^{1\cap5}}_{-}\cdot\underbrace{\overbrace{[jsr]}^{4\cap7}}_{+} + \underbrace{\overbrace{[jar]}^{1\cap5}}_{-}\cdot\underbrace{[jsx]}_{\Rightarrow+}$$

$$\{s|cjxq\} = \underbrace{\overbrace{[scj]}^{1\cap9}}_{-}\cdot\underbrace{[sxq]}_{\Rightarrow+} - \underbrace{\overbrace{[scx]}^{9}}_{0}\cdot\underbrace{[sjq]}_{?} + \underbrace{\overbrace{[scq]}^{1\cap9}}_{-}\cdot\underbrace{\overbrace{[sjx]}^{\{j|arsx\}}}_{-}$$

$$\{u|asoq\} = \underbrace{\overbrace{[uas]}^{1\cap5}}_{+}\cdot\underbrace{\overbrace{[uoq]}^{6\cap7}}_{-} - \underbrace{\overbrace{[uao]}^{1\cap5}}_{+}\cdot\underbrace{[usq]}_{\Rightarrow-} + \underbrace{\overbrace{[uaq]}^{1\cap5}}_{+}\cdot\underbrace{\overbrace{[uso]}^{3\cap7}}_{-}$$

$$\{q|acuo\} = \underbrace{\overbrace{[qac]}^{1\cap2}}_{+}\cdot\underbrace{\overbrace{[quo]}^{6\cap7}}_{-} - \underbrace{\overbrace{[qau]}^{1\cap5}}_{-}\cdot\underbrace{\overbrace{[qco]}^{2\cap6}}_{-} + \underbrace{\overbrace{[qao]}^{2\cap6}}_{-}\cdot\underbrace{[qcu]}_{\Rightarrow-}$$

$$\{q|aueo\} = \underbrace{\overbrace{[qau]}^{1\cap5}}_{-}\cdot\underbrace{\overbrace{[qeo]}^{2\cap6}}_{-} - \underbrace{\overbrace{[qae]}^{2\cap6}}_{+}\cdot\underbrace{\overbrace{[quo]}^{6\cap7}}_{-} + \underbrace{\overbrace{[qao]}^{2\cap6}}_{-}\cdot\underbrace{[que]}_{\Rightarrow+}$$

$$\{q|cusx\} = \underbrace{\overbrace{[qcu]}^{\{q|acou\}}}_{-}\cdot\underbrace{\overbrace{[qsx]}^{\{s|cjqx\}}}_{+} - \underbrace{\overbrace{[qcs]}^{1\cap9}}_{+}\cdot\underbrace{[qux]}_{\Rightarrow-} + \underbrace{\overbrace{[qcx]}^{1\cap9}}_{+}\cdot\underbrace{\overbrace{[qus]}^{\{u|aoqs\}}}_{-}$$

$$\{q|cuex\} = \underbrace{\overbrace{[qcu]}^{\{q|acou\}}}_{-} \cdot \underbrace{[qex]}_{\Rightarrow +} - \underbrace{\overbrace{[qce]}^{1\cap 9}}_{+} \cdot \underbrace{\overbrace{[qux]}^{\{q|csux\}}}_{-} + \underbrace{\overbrace{[qcx]}^{1\cap 9}}_{+} \cdot \underbrace{\overbrace{[que]}^{\{q|aeou\}}}_{+}$$

$$\{q|ayeo\} = \underbrace{\overbrace{[qay]}^{1\cap 5}}_{-} \cdot \underbrace{\overbrace{[qeo]}^{2\cap 6}}_{-} - \underbrace{\overbrace{[qae]}^{2\cap 6}}_{+} \cdot \underbrace{\overbrace{[qyo]}^{6\cap 7}}_{-} + \underbrace{\overbrace{[qao]}^{2\cap 6}}_{-} \cdot \underbrace{[qye]}_{\Rightarrow +}$$

After plugging in the sign of the bracket $[bdqye]$ in $\mathcal{M}^\Delta$ or in the contraction $\mathcal{M}^\Delta/\{b\}$, we find a contradiction. In other words, $\mathcal{M}^\Delta$ does not exist.

$$\{beq|dayx\} = \underbrace{\overbrace{[beqda]}^{2\cap 6}}_{+} \cdot \underbrace{\overbrace{[beqyx]}^{6\cap 8}}_{0} - \underbrace{\overbrace{[beqdy]}^{\{bdq|aeoy\}}}_{+} \cdot \underbrace{\overbrace{[beqax]}^{2\cap 6}}_{-} + \underbrace{\overbrace{[beqdx]}^{\{bdq|ceux\}}}_{-} \cdot \underbrace{\overbrace{[beqay]}^{2\cap 6}}_{-}$$

Our given matroid polytope $\mathcal{M}$ has no polar $\mathcal{M}^\Delta$. The rank of $\mathcal{M}$ is minimal because of Steinitz's theorem. □

**3.1. Remark.** There is no matroid polytope with 8 vertices in rank 5 without a polar. A matroid polytope $\mathcal{M}$ with 8 vertices in rank 5 has a dual oriented matroid $\mathcal{M}^*$ in rank 3 with 8 elements which is realizable. Therefore, $\mathcal{M}$ is realizable as well, and its face lattice must be polytopal. This shows that all 3-spheres (including the Barnette sphere and the Brückner sphere) do not lead to nonpolytopal matroid polytopes.

The minimal number of vertices for a nonpolytopal matroid polytope must be 9. Because our $\mathcal{M}$ is the only uniform nonpolytopal matroid polytope with 9 vertices, compare 2.3, we can speak of $\mathcal{M}$ as *the* minimal (with respect to both the rank and the number of vertices) uniform matroid polytope without a polar.

**Acknowledgment.** We wish to thank Christoph Schulz for many stimulating discussions.

## REFERENCES

[1] A. ALTSHULER, J. BOKOWSKI, AND L. STEINBERG, *The classification of simplicial 3-spheres with nine vertices into polytopes and nonpolytopes*, Discrete Math., 31 (1980), pp. 115–124.
[2] C. ANTONIN, *Ein algorithmusansatz für realisierungsfragen im $E^d$ getestet an kombinatorischen 3-Sphären*, Staatsexamensarbeit, Bochum, 1982.
[3] L. J. BILLERA AND B. S. MUNSON, *Polarity and inner products in oriented matroids*, European J. Combin., 5 (1984), pp. 293–308.
[4] A. BJÖRNER, M. LAS VERGNAS, B. STURMFELS, N. WHITE, AND G. M. ZIEGLER, *Oriented Matroids*, Cambridge University Press, Cambridge, 1993.
[5] J. BOKOWSKI, *Handbook of Convex Geometry*, Elsevier, North–Holland, Amsterdam, 1993.
[6] J. BOKOWSKI AND J. RICHTER-GEBERT, *Reduction theorems for oriented matroids*, manuscript, 1990.
[7] J.-P. ROUDNEFF AND B. STURMFELS, *Simplicial cells in arrangements and mutations of oriented matroids*, Geom. Dedicata, 27 (1988), pp. 153–170.
[8] R. W. SHANNON, *Simplicial cells in arrangements of hyperplanes*, Geom. Dedicata, 8 (1979), pp. 179–187.

SIAM J. DISC. MATH.
Vol. 8, No. 4, pp. 678–683, November 1995

# EQUIDISTRIBUTION IN ALL DIMENSIONS OF WORST-CASE POINT SETS FOR THE TRAVELING SALESMAN PROBLEM*

TIMOTHY LAW SNYDER† AND J. MICHAEL STEELE‡

**Abstract.** Given a set $S$ of $n$ points in the unit square $[0,1]^d$, an optimal traveling salesman tour of $S$ is a tour of $S$ that is of minimum length. A *worst-case point set* for the traveling salesman problem in the unit square is a point set $S^{(n)}$ whose optimal traveling salesman tour achieves the maximum possible length among all point sets $S \subset [0,1]^d$, where $|S| = n$. An open problem is to determine the structure of $S^{(n)}$. We show that for any rectangular parallelepiped $R$ contained in $[0,1]^d$, the number of points in $S^{(n)} \cap R$ is asymptotic to $n$ times the volume of $R$. Analogous results are proved for the minimum spanning tree, minimum-weight matching, and rectilinear Steiner minimum tree. These equidistribution theorems are the first results concerning the structure of worst-case point sets like $S^{(n)}$.

**Key words.** equidistribution, worst-case, nonlinear growth, traveling salesman, rectilinear Steiner tree, minimum spanning tree, minimum-weight matching

**AMS subject classifications.** 68R10, 05C45, 90C35, 68U05

## 1. Introduction.

In this note we show that for many problems of Euclidean combinatorial optimization, the maximal value of the objective function is attained by point sets that are asymptotically equidistributed. To facilitate exposition, we focus at first on the traveling salesman problem (TSP) for a finite set $S$ of points in the $d$-dimensional unit cube $[0,1]^d$. Let $\tau(S)$ denote the set of tours that span $S$. The optimal TSP*cost* of $S$ is the value given by

$$\text{TSP}(S) = \min_{T\in\tau(S)} \sum_{e\in T} |e|, \tag{1.1}$$

where $|e|$ denotes the Euclidean length of the edge $e$.

For each dimension $d \geq 2$, there are constants $c_d$ such that

$$\text{TSP}(S) \leq c_d |S|^{(d-1)/d}, \tag{1.2}$$

where $|S|$ denotes the cardinality of $S$. Considerable effort has been devoted to determining good bounds on $c_d$; the earliest bounds are due to Few [2], and the current records are held by Karloff [5] and Goddyn [3]. Simply by considering the rectangular lattice, one can see there are also constants $c'_d > 0$ such that, for all $n \geq 2$,

$$\max_{\substack{S\subset[0,1]^d \\ |S|=n}} \text{TSP}(S) \geq c'_d n^{(d-1)/d}. \tag{1.3}$$

*Received by the editors February 4, 1994; accepted for publication (in revised form) November 29, 1994.

†Department of Computer Science, Georgetown University, Washington, DC 20057. The research of this author was supported in part by a Georgetown University 1991 Summer Research Award, a Georgetown University 1992 Junior Faculty Research Fellowship, and the Georgetown College John F. Kennedy, Jr. Faculty Research Fund.

‡Department of Statistics, The Wharton School, University of Pennsylvania, Philadelphia, Pennsylvania 19104. The research of this author was supported in part by National Science Foundation grant DMS92-11634 and Army Research Office grant DAAL03-91-G-0110.

If we let $\rho_{\mathrm{TSP}}(n) = \max\{\, \mathrm{TSP}(S) : S \subset [0,1]^d, |S| = n \,\}$, then the usual considerations of continuity and compactness show that there are $n$-sets $S$ for which $\mathrm{TSP}(S) = \rho_{\mathrm{TSP}}(n)$ (cf. [7], p. 115); these are the *worst-case point sets* referred to in our title. We suppress $\rho_{\mathrm{TSP}}$'s dependence on $d$ to keep notation simple.

The main result obtained here is that worst-case point sets are asymptotically equidistributed in the sense made explicit in the following theorem.

THEOREM 1. *If $\{S^{(n)} : 2 \le n < \infty\}$ is a sequence of worst-case TSP point sets with $S^{(n)} \subset [0,1]^d$, $d \ge 2$, and $|S^{(n)}| = n$, then for any rectangular parallelepiped $R \subset [0,1]^d$, we have*

$$\lim_{n\to\infty} \frac{1}{n}|S^{(n)} \cap R| = \mathrm{vol}_d(R). \tag{1.4}$$

While Theorem 1 is certainly intuitive, the proof we provide requires more than first principles; it relies essentially on the result of Steele and Snyder [10] that there exist constants $\beta_d > 0$ such that

$$\lim_{n\to\infty} \frac{\rho_{\mathrm{TSP}}(n)}{n^{(d-1)/d}} = \beta_d. \tag{1.5}$$

The exact asymptotic result (1.5) was motivated by the classical result of Beardwood, Halton, and Hammersley [1] for the case of random point sets, and it seems to provide just the refinement of bounds like (1.2) and (1.3) that is needed to obtain equidistribution limit theorems.

We note that a proof of Theorem 1 in dimension two using techniques different from the ones we use here is given in [9]. We also note that Theorem 1 has a close connection to some results and a conjecture of Supowit, Reingold, and Plaisted [11]. This connection will be explained more fully in §4, after we have developed some notation.

In the next section, we prove Theorem 1; §3 deals with problems other than the TSP.

**2. Proof of Theorem 1.** For any fixed integer $m \ge 2$, we partition $[0,1]^d$ into $m^d$ subcubes $Q_i$, where $1 \le i \le m^d$, each of side length $1/m$. For any rectangle $R$ and any $\epsilon > 0$, there is an $m$ and sets $A$ and $B$ such that $\cup_A Q_i \subset R \subset \cup_B Q_i$ and $\mathrm{vol}_d(\cup_{i\in B-A} Q_i) \le \epsilon\, \mathrm{vol}_d(R)$; hence, to prove Theorem 1, it suffices to consider equidistribution with respect to the $Q_i$. Specifically, it suffices to show that for each $m \ge 2$ and $1 \le i \le m^d$, we have

$$\lim_{n\to\infty} \frac{|Q_i \cap S^{(n)}|}{n} = \frac{1}{m^d}. \tag{2.1}$$

Our proof of (2.1) depends on the equality case of Hölder's inequality, which tells us that for $1 < p < \infty$ and $u_i, v_i \ge 0$, we have $\sum_{i=1}^k u_i v_i \le \left(\sum_{i=1}^k u_i^p\right)^{1/p} \left(\sum_{i=1}^k v_i^{p/(p-1)}\right)^{(p-1)/p}$. Setting $v_i = 1$ for $1 \le i \le k$, we have $\sum_{i=1}^k u_i = \sum_{i=1}^k u_i \cdot 1 \le (\sum_{i=1}^k u_i^p)^{1/p} (\sum_{i=1}^k 1^{p/(p-1)})^{(p-1)/p} = (\sum_{i=1}^k u_i^p)^{1/p} k^{(p-1)/p}$. The fact that is important for us is that one can have equality in this bound if and only if $u_1 = u_2 = \cdots = u_k$ ([4], pp. 21–26).

Let $s(n,i) = |Q_i \cap S^{(n)}|$; i.e., $s(n,i)$ is the number of points of a worst-case point set $S^{(n)}$ that appear in the the $i$th subcube. We first establish a limit result concerning the $s(n,i)$ that measures their aggregate size in a way that works usefully with Hölder's inequality.

LEMMA 1. *For all $m \geq 2$, we have*

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{m^d} s(n,i)^{(d-1)/d}}{n^{(d-1)/d}} = m. \tag{2.2}$$

*Proof.* First, write (1.5) as

$$\rho_{\mathrm{TSP}}(n) = \beta_d n^{(d-1)/d} + r(n), \quad \text{where } r(n) = o(n^{(d-1)/d}). \tag{2.3}$$

Let $W$ denote a closed walk on $S^{(n)} = \{x_1, x_2, \ldots, x_n\}$; i.e, $W$ is a sequence of edges $(x_{i_1}, x_{i_2}), (x_{i_2}, x_{i_3}), \ldots, (x_{i_{k-1}}, x_{i_k}), (x_{i_k}, x_{i_1})$ that visits each point of $S^{(n)}$ at least once and begins and ends at the same point. Even if $W$ visits some points more than once and traverses some edges more than once, $W$ is feasible for the traveling salesman problem on $S^{(n)}$, so $\mathrm{TSP}(S^{(n)}) \leq \sum_{e\in W} |e|$.

We now construct a particular $W$ on $S^{(n)}$ in the tradition of [6] and [11]. In each subcube $Q_i$ for which $S^{(n)} \cap Q_i \neq \emptyset$, construct an optimal traveling salesman tour $T_i$ of $S^{(n)} \cap Q_i$. This creates a set of at most $m^d$ within-subsquare tours. We then select a point $x_i^\star$ from each $T_i$ and let $T^\star$ be an optimal traveling salesman tour of $\{x_1^\star, x_2^\star, \ldots, x_{m^d}^\star\}$. The closed walk $W$ is then formed by visiting subsquares in the order specified by $T^\star$, visiting all members of subsquare $Q_i$ by traversing $T_i$ whenever $T^\star$ reaches $x_i^\star$.

To assess the length of $W$, we first note that $T^\star$ is a TSP tour of $m^d$ points, so by (1.2), $\sum_{e\in T^\star} |e| \leq c_d m^{d-1}$. This gives

$$\begin{aligned} \rho_{\mathrm{TSP}}(n) &= \mathrm{TSP}(S^{(n)}) \\ &\leq \sum_{e\in W} |e| \\ &= \sum_{i=1}^{m^d} \mathrm{TSP}(S^{(n)} \cap Q_i) + \sum_{e\in T^\star} |e| \\ &\leq \sum_{i=1}^{m^d} \mathrm{TSP}(S^{(n)} \cap Q_i) + c_d m^{d-1}. \end{aligned} \tag{2.4}$$

We now use (2.3) in (2.4) along with the fact that $\mathrm{TSP}(S^{(n)} \cap Q_i)$ is at most $\rho_{\mathrm{TSP}}(s(n,i))$ scaled by the subcube size $1/m$ to get

$$\begin{aligned} \rho_{\mathrm{TSP}}(n) &= \beta_d n^{(d-1)/d} + r(n) \\ &\leq \sum_{i=1}^{m^d} \frac{\rho_{\mathrm{TSP}}(s(n,i))}{m} + c_d m^{d-1} \\ &\leq \frac{1}{m}\sum_{i=1}^{m^d} \beta_d s(n,i)^{(d-1)/d} + \frac{1}{m}\sum_{i=1}^{m^d} r(s(n,i)) + c_d m^{d-1}, \end{aligned} \tag{2.5}$$

where, for all $1 \leq i \leq m^d$, the value $|r(s(n,i))| \leq \max_{k\leq n}\{r(k)\} = o(n^{(d-1)/d})$. Since $m$ is fixed, we cancel $\beta_d$ in (2.5) to find

$$\sum_{i=1}^{m^d} s(n,i)^{(d-1)/d} \geq m n^{(d-1)/d} + h(n), \tag{2.6}$$

where $h(n) = o(n^{(d-1)/d})$. Dividing by $n^{(d-1)/d}$ and letting $n \to \infty$ thus proves half of the lemma. To obtain the other half, just apply Hölder's inequality with $p = d/(d-1)$ to $\sum_{i=1}^{m^d} s(n,i)^{(d-1)/d}$ and use $\sum_{i=1}^{m^d} s(n,i) = n$ to find that $\sum_{i=1}^{m^d} s(n,i)^{(d-1)/d} \leq mn^{(d-1)/d}$. □

We are now in position to prove Theorem 1. First, we recall the subsequence convergence principle which says that if $(a_k)$ is any sequence of real numbers with the property that for any integers $n_1 < n_2 < \cdots < n_k < \cdots$ there is a further subsequence $n'_1 < n'_2 < \cdots < n'_k < \cdots$ such that $a_{n'_k} \to \alpha$ as $k \to \infty$, then in fact one must have $a_k \to \alpha$ as $k \to \infty$. One easy way to see the validity of this principle is to note that if $a_k$ does not converge to $\alpha$, then there is some $\alpha' \neq \alpha$, $-\infty \leq \alpha' \leq \infty$, and some subsequence of $(a_k)$ that converges to $\alpha'$.

Now let $(n_k)$ be a given increasing sequence of integers. Since $0 \leq s(n,i)/n \leq 1$ for all $n$ and $i$, we can find a subsequence $(n'_k)$ of the $(n_k)$ and $m^d$ constants $0 \leq \alpha_i \leq 1$ such that, for all $1 \leq i \leq m^d$, we have

$$\lim_{k\to\infty} s(n'_k, i)/n = \alpha_i. \tag{2.7}$$

Now, since $\sum_{i=1}^{m^d} s(n,i) = n$, we have from (2.7) that

$$\sum_{i=1}^{m^d} \alpha_i = 1. \tag{2.8}$$

Similarly, by (2.2) and (2.7), we have

$$\sum_{i=1}^{m^d} \alpha_i^{(d-1)/d} = m. \tag{2.9}$$

Now, equation (2.9) and Hölder's inequality applied with $u_i = \alpha_i^{(d-1)/d}$ and $p = d/(d-1)$ give us

$$m = \sum_{i=1}^{m^d} \alpha_i^{(d-1)/d} \leq \Big(\sum_{i=1}^{m^d} \alpha_i\Big)^{(d-1)/d} \Big(\sum_{i=1}^{m^d} 1^d\Big)^{1/d}. \tag{2.10}$$

But, by (2.8), we see that equality holds in (2.10), and thus $\alpha_1^{(d-1)/d} = \alpha_2^{(d-1)/d} = \cdots = \alpha_{m^d}^{(d-1)/d}$, so applying (2.8) again, we see that $\alpha_i = 1/m^d$ for all $i$. By the subsequence convergence principle noted after Lemma 1, we therefore have for all $1 \leq i \leq m^d$ that $s(n,i)/n \to 1/m^d$ as $n \to \infty$, and the proof is complete. □

## 3. Equidistribution in related problems.

The method just used for the TSP can be applied to the minimum spanning tree, the minimum-length matching, and the rectilinear minimum Steiner tree. If $\mathrm{L} = \mathrm{L}(S)$ denotes the optimal cost associated with any of these, then we can define $\rho_{\mathrm{L}}(n) = \sup_{\substack{S\subset[0,1]^d \\ |S|=n}} \mathrm{L}(S)$ and let $S_{\mathrm{L}}^{(n)}$ be such that $\mathrm{L}(S_{\mathrm{L}}^{(n)}) = \rho_{\mathrm{L}}(n)$. To show that $S_{\mathrm{L}}^{(n)}$ is asymptotically equidistributed boils down to checking that L satisfies two conditions:

1. $\rho_{\mathrm{L}}(n) = \beta_{\mathrm{L},d} n^{(d-1)/d} + o(n^{(d-1)/d})$, where $\beta_{\mathrm{L},d} > 0$ is constant; and
2. $\rho_{\mathrm{L}}(n) \leq m^{-1} \sum_{i=1}^{m^d} \rho_{\mathrm{L}}(s_{\mathrm{L}}(n,i)) + o(n^{(d-1)/d})$, where $s_{\mathrm{L}}(n,i) = |S_{\mathrm{L}}^{(n)} \cap Q_i|$.

Condition 1 has been proved for the minimum spanning tree, minimum-length matching, and rectilinear Steiner tree problems (cf., [10] and [8]), and condition 2 can be verified for these problems by the method used in the proof of Lemma 1.

For example, if $\mathrm{L}(S) = \mathrm{MST}(S)$ denotes the total length of a minimum spanning tree of $S$, we first form a minimum spanning tree $\mathrm{MST}(S_{\mathrm{MST}}^{(n)} \cap Q_i)$ on each $S_{\mathrm{MST}}^{(n)} \cap Q_i$. These trees can then be interconnected at total cost $o(n^{(d-1)/d})$ by adding $m^d - 1$ edges, each costing no more than $c/m$, where $c$ is constant. This forms a heuristic tree on $S_{\mathrm{MST}}^{(n)}$. Since the lengths $\mathrm{MST}(S_{\mathrm{MST}}^{(n)} \cap Q_i)$ are no greater than the worst-case (within-subcube) lengths $\rho_{\mathrm{MST}}(s_{\mathrm{MST}}(n,i))/m$, condition 2 follows.

Checking these conditions for each of the problems yields the following theorem.

THEOREM 2. *If* $\{ S_{\mathrm{L}}^{(n)} : 1 \le n < \infty \}$ *is a sequence of worst-case point sets for the function* L, *where* L *is the minimum spanning tree, the minimum-length matching, or the rectilinear minimum Steiner tree, then, for any rectangular parallelepiped* $R \subset [0,1]^d$,

$$\lim_{n\to\infty} \frac{1}{n} \left| S_{\mathrm{L}}^{(n)} \cap R \right| = \mathrm{Vol}_d(R). \tag{3.1}$$

**4. Concluding remarks.** The asymptotic equidistribution of worst-case point sets for the problems we have considered offers some support to the conjecture of [11] that worst-case point sets are approximated by lattices as $n \to \infty$. It is still a major open problem to resolve this conjecture.

Theorem 1 has a rather subtle relationship to some results of Supowit, Reingold, and Plaisted [11]; we explain here how these results relate to ours. In addition to improving current bounds on the constants $c_2$ and $c_2'$ in (1.2) and (1.3), their analysis of the worst-case TSP in $\mathbb{R}^2$ decomposed $[0,1]^2$ into $m^2$ labeled subsquares of side length $1/m$, then constructed a heuristic algorithm similar to that of [6]. Supowit, Reingold, and Plaisted noted that the worst-case performance of the heuristic is attained on point sets that are equidistributed, and they used this observation to prove that the leading constant of the worst-case length of their heuristic tour is identical to the worst-case TSP constant $\beta_2$ in (1.5). This observation does not produce an equidistribution result for worst-case point sets, but it is suggestive of a result like Theorem 1. Still, a rigorous proof of asymptotic equidistribution of a worst-case TSP point set required a much different path.

There are other open problems that are motivated by our results. For the Euclidean Steiner problem, the limit result for condition 1 in §3 has yet to be established. We believe such a result holds, and it would imply that a worst-case point set for the Euclidean Steiner problem is asymptotically equidistributed. It is also likely that the Steiner points in the Euclidean and rectilinear cases are asymptotically equidistributed.

Another problem concerns the greedy matching. Though condition 1 in §3 holds for this problem, the methods we use to verify condition 2 do not work, since they require a minimality condition. Hence, since the greedy matching is not a minimum-length matching, showing equidistribution for a worst-case point set for the greedy matching problem remains an open problem.

## REFERENCES

[1] J. BEARDWOOD, J. H. HALTON, AND J. HAMMERSLEY, *The shortest path through many points*, Proc. Cambridge Philosophical Society, 55 (1959), pp. 299–327.

[2] L. FEW, *The shortest path and the shortest road through n points in a region,* Mathematika, 2 (1955), pp. 141–144.

[3] L. GODDYN, *Quantizers and the worst-case Euclidean traveling salesman problem,* J. Combin. Theory, Ser. B, 50 (1990), pp. 65–81.

[4] G. H. HARDY, J. E. LITTLEWOOD, AND G. PÓLYA, *Inequalities*, Cambridge University Press, Cambridge, 1964.

[5] H. J. KARLOFF, *How long can a Euclidean traveling salesman tour be?*, SIAM J. Discrete Math., 2 (1989), pp. 91–99.

[6] R. M. KARP, *The probabilistic analysis of some combinatorial search algorithms,* in Algorithms and Complexity: New Directions and Recent Results, J. F. Traub, ed., Academic Press, New York, 1976, pp. 1–19.

[7] S. MORAN, *On the length of optimal TSP circuits in sets of bounded diameter,* J. Combin. Theory, Ser. B, 37 (1984), pp. 113–141.

[8] T. L. SNYDER, *Worst-case minimal rectilinear Steiner trees in all dimensions,* Discrete Comput. Geom., 8 (1992), pp. 73–92.

[9] T. L. SNYDER AND J. M. STEELE, *Equidistribution of point sets for the traveling salesman and related problems,* Proc. Fourth Annual ACM–SIAM Symposium on Discrete Algorithms, Association for Computing Machinery, New York, NY, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1993, pp. 462–466.

[10] J. M. STEELE AND T. L. SNYDER, *Worst-case growth rates of some classical problems of combinatorial optimization,* SIAM J. Comput., 18 (1989), pp. 278–287.

[11] K. J. SUPOWIT, E. M. REINGOLD, AND D. A. PLAISTED, *The traveling salesman problem and minimum matchings in the unit square,* SIAM J. Comput., 12 (1983), pp. 144–156.