SIAM J. DISC. MATH.
Vol. 9, No. 1, pp. 1–6, February 1996

A CHARACTERIZATION OF NONNEGATIVE BOX-GREEDY MATRICES *

ULRICH FAIGLE†, ALAN J. HOFFMAN‡, AND WALTER KERN†

Abstract. Given an ordering of the variables according to nonincreasing coefficients of the objective function $c^T x$, the nonnegative matrix A is said to be *greedy* if, under arbitrary nonnegative constraint vectors b and h, the greedy algorithm maximizes $c^T x$ subject to $Ax \leq b$, $0 \leq x \leq h$. Extending a result of Hoffman, Kolen, and Sakarovitch for (0, 1)-matrices, we characterize greedy matrices in terms of forbidden submatrices, which yields polynomial recognition algorithms for various classes of greedy matrices. The general recognition problem for the existence of forbidden submatrices is shown to be NP-complete

Key words. greedy algorithm, linear program

AMS subject classification. 90C27

1. Introduction.

The greedy algorithm constitutes a fundamental algorithmic principle in combinatorics and has received considerable attention within the framework of linear programming. Basically, one orders the components of the objective function according to nonincreasing values and then constructs a feasible solution for the linear program greedily in that order of the variables. The problem is to find out under what conditions the greedy solution is optimal.

A well-known example is the (generalization of the) matroid greedy algorithm of Edmonds (1970), which refers to (0, 1)-constraint matrices A and is robust relative to arbitrary linear objective functions as long as the vector b in the feasibility conditions

$$Ax \leq b, \qquad x \geq 0, \tag{1.1}$$

corresponds to a submodular set function.

A different line of research on greedily solvable linear programs was initiated by Hoffman, Kolen, and Sakarovitch (1985). Here the order of the variables is *fixed*; i.e., one only considers objective functions $c^T x$, where $c^T = (c_1, \ldots, c_n)$ satisfies $c_1 \geq c_2 \geq \cdots \geq c_n$. The matrix A is then termed *(box)-greedy* if the greedy algorithm successfully solves the linear program

$$\max c^T x \quad \text{s.t. } Ax \leq b,\ 0 \leq x \leq h, \tag{1.2}$$

relative to any feasible choice of c, b, and h. Hoffman, Kolen, and Sakarovitch show that greedy (0, 1)-matrices A can be characterized by two simple forbidden submatrices.

It is our purpose here to provide a combinatorial characterization of arbitrary nonnegative greedy matrices that was announced (without proof) in Hoffman (1985). We do this in §2 by showing that it suffices to check greediness for the *quasi-diagonal* submatrices of A, i.e., for those submatrices that can be brought into essentially

* Received by the editors November 23, 1993; accepted for publication (in revised form) January 27, 1995.

† Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands (faigle@math.utwente.nl and kern@math.utwente.nl).

‡ Department of Mathematical Sciences, IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (hoffa@watson.ibm.com).

diagonal form by a suitable permutation of the rows. Checking greediness of particular quasi-diagonal submatrices turns out to be very simple.

Our characterization yields polynomial recognition algorithms for various special classes of nonnegative box-greedy matrices (cf. §3). The general problem turns out to be NP-complete.

2. Greedy matrices. Let $A \in \mathbb{R}_+^{m\times n}$ be a nonnegative real matrix and $b \in \mathbb{R}_+^m, h \in \mathbb{R}_+^n$ be nonnegative vectors. The *greedy algorithm* constructs a feasible solution $x^* = (x_1^*, \dots, x_n^*)^T \in \mathbb{R}^n$ for the system

$$Ax \le b, \qquad 0 \le x \le h, \tag{2.1}$$

as follows:

FOR $k = 1, \dots, n$

$$x_k^* = \max\{x \in \mathbb{R} | (x_1^*, \dots, x_{k-1}^*, x, 0, \dots, 0)^T \text{ feasible}\}.$$

We say that the matrix A is *(box)-greedy* if, for any choice of b and h, the greedy solution is optimal for the linear program

$$\max c^T x \quad \text{s.t. } Ax \le b,\ 0 \le x \le h, \tag{2.2}$$

whenever $c = (c_1, \dots, c_n)^T$ satisfies $c_1 \ge c_2 \ge \cdots \ge c_n \ge 0$. Note that the greedy algorithm for (2.2) depends on b and h but not on c. Our aim is to characterize greedy matrices in terms of inherent combinatorial properties without any reference to constraint vectors b and h. Choosing b and h appropriately, it is easy to see that the matrix A can only be greedy if all submatrices of A are greedy. We are interested in particular submatrices of A.

The matrix B is a *normalized submatrix* of A if there is a submatrix A' of A such that the first element in each row of A' is strictly positive and B arises from A' by dividing each row by its first element. Let B_0 be the submatrix obtained by removing the first column of B. So

$$B = [\mathbf{1}|B_0],$$

where $\mathbf{1}$ denotes the vector with all 1's of appropriate dimension.

The normalized submatrix B of A is *quasi-diagonal* if the rows of B_0 can be permuted so that a diagonal matrix results. With each quasi-diagonal normalized submatrix B we associate the *weight*

$$w(B) = \sum \frac{1}{d_j}, \tag{2.3}$$

where the d_j's are the nonzero entries of B_0. (Set $w(B) = 0$ if B_0 contains no nonzero element.)

The quasi-diagonal submatrix B is *feasible* if $w(B) \le 1$.

LEMMA. *If A is greedy, then every quasi-diagonal normalized submatrix B is feasible.*

Proof. Suppose B is normalized submatrix with $w(B) > 1$, where without loss of generality $B_0 = \operatorname{diag}(d_1, \dots, d_k)$ and $d_1 d_2 \dots d_k \ne 0$.

Then the greedy algorithm yields objective function value 1 for the linear program

$$\max \mathbf{1}^T x \quad \text{s.t. } Bx \le \mathbf{1},\ x \ge 0.$$

Since $\bar{x} = (0, d_1^{-1}, \ldots, d_k^{-1})$ is feasible with $\mathbf{1}^T x = w(B) > 1$, B is obviously not greedy. Hence also A is not greedy. □

We now state a sufficient condition.

PROPOSITION. *Assume that for every normalized submatrix B of A, the linear inequality system*

$$B_0\lambda \geq \mathbf{1}, \qquad \lambda \geq 0,$$

either is not feasible or has a solution λ with $\mathbf{1}^T\lambda \leq 1$. Then A is greedy.

Proof. Consider the feasible linear program

$$\max c^T x \quad \text{s.t. } Ax \leq b,\ 0 \leq x \leq h,$$

with $c_1 \geq \cdots \geq c_n \geq 0$. Among all optimal solutions for the linear program, let $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)^T$ be one with first component $\bar{x}_1$ as large as possible.

If $\bar{x}_1 = x_1^*$, where $x^* = (x_1^*, \ldots, x_n^*)$ is the greedy solution, we may assume by induction on the number of variables that

$$c_2\bar{x}_2 + \cdots + c_n\bar{x}_n \leq c_2x_2^* + \cdots + c_nx_n^*$$

and hence $c^Tx^* \geq c^T\bar{x}$, i.e., that x^* is optimal.

So suppose $\bar{x}_1 < x_1^*$. We will then exhibit a feasible solution $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)$ with $\tilde{x}_1 > \bar{x}_1$ and $c^T\tilde{x} \geq c^T\bar{x}$, a contradiction to the choice of $\bar{x}$.

Let A' be the submatrix of those rows A_i of A with leading coefficient $a_{i1} \neq 0$ and $A_i\bar{x} = b_i$, restricted to those columns A^j such that $j = 1$ or $\bar{x}_j > 0$. Denote by B the normalized submatrix associated with A'.

Because $\bar{x}_1 < x_1^*$, the matrix B_0 has (at least) one nonzero element in each row. Hence $B_0\lambda \geq \mathbf{1}, \lambda \geq 0$, is feasible and we may assume that $\mathbf{1}^T\lambda \leq 1$ holds.

Let $\varepsilon_1 > 0$ be such that the vector $(\bar{x}_1 + \varepsilon_1, \bar{x}_2, \ldots, \bar{x}_n)$ does not violate any constraint A_i outside A', and choose $\varepsilon_2 > 0$ such that

$$\bar{x}_j - \varepsilon_2\lambda_j \geq 0 \quad \text{for all } \bar{x}_j > 0,\ j \geq 2.$$

With $\varepsilon = \min\{\varepsilon_1, \varepsilon_2\} > 0$, we thus conclude from $B_0\lambda \geq \mathbf{1}$ that the vector $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)^T$ given by

$$\tilde{x}_j = \begin{cases} \bar{x}_1 + \varepsilon & \text{if } j = 1, \\ \bar{x}_j - \varepsilon\lambda_j & \text{if } j \geq 2 \text{ and } \bar{x}_j > 0, \\ 0 & \text{otherwise} \end{cases}$$

is feasible for the linear program. Moreover, $c_1 \geq \max\{c_j | j \geq 2\}$ and $\mathbf{1}^T\lambda \leq 1$ yield $c^T\tilde{x} \geq c^T\bar{x}$, which is the desired contradiction. □

Our main result asserts that the necessary condition in the lemma is equivalent with the sufficient condition in the proposition.

THEOREM 1. *Let A be a nonnegative matrix. Then A is greedy if and only if every quasi-diagonal normalized submatrix B of A is feasible.*

Proof. We show that A violates the necessary condition of the lemma if A violates the sufficient condition of the proposition.

Suppose that B is a (row and column) minimal normalized submatrix of A such that the linear program

$$\min \mathbf{1}^T\lambda \quad \text{s.t. } B_0\lambda \geq \mathbf{1},\ \lambda \geq 0, \tag{2.4}$$

is feasible with optimal objective function value $z_0 > 1$. We claim that B contains some infeasible quasi-diagonal submatrix.

To fix notation, let E be the index set of the rows of B_0 and $E_1, \ldots, E_k$ be the supports of the k columns of B_0. Because B_0 was chosen feasible and column-minimal with objective function value strictly greater than 1, we must have

$$E \subseteq \bigcup_{s=1}^{k} E_s$$

$$\text{and} \quad E \not\subseteq \bigcup_{s \neq t} E_s \quad \text{for } t = 1, \ldots, k.$$

So there are elements $e_1, \ldots, e_k \in E$ with $e_s \in E_s$ and $e_s \notin E_t$ for $t \neq s$. Now restrict B to the submatrix $\bar{B}$ consisting of the rows of B that correspond to $\{e_1, \ldots, e_k\}$, and observe that $\bar{B}$ is a quasi-diagonal normalized submatrix of A. If $d_1, \ldots, d_k$ are the nonzero entries of $\bar{B}_0$, the proof is finished once we can establish the relation

$$\frac{1}{d_1} + \cdots + \frac{1}{d_k} = z_0 > 1.$$

Let $\bar{\lambda} = (\bar{\lambda}_1, \ldots, \bar{\lambda}_k)^T$ be an optimal feasible solution for (2.4). Then $\bar{B}_0 \bar{\lambda} \geq \mathbf{1}$ implies

$$\bar{\lambda}_s \geq \frac{1}{d_s} \quad \text{for } s = 1, \ldots, k.$$

If $\bar{\lambda}_1 \neq d_1^{-1}$, for example, then the e_1-constraint in (2.4) is not binding for the optimal solution $\bar{\lambda}$. Hence also the linear program

$$\min \mathbf{1}^T \lambda \quad \text{s.t. } B_0' \lambda \geq \mathbf{1},\ \lambda \geq 0, \tag{2.4'}$$

where (2.4′) is obtained by dropping the e_1-constraint in (2.4), is feasible with optimal objective function value $z_0' > 1$, which contradicts the row-minimality of the matrix B.

Hence we conclude $\bar{\lambda} = (d_1^{-1}, \ldots, d_k^{-1})^T$. □

3. Recognition complexity of box-greedy matrices. For the fixed parameter $K \geq 0$, let $\mathcal{M}(K)$ be the class of all nonnegative matrices A such that the maximum ratio of every pair of nonzero elements in each row of A is bounded by K.

COROLLARY 1. *For every fixed $K \geq 0$, there is a polynomial algorithm to decide whether the matrix $A \in \mathcal{M}(K)$ is (box-)greedy.*

Because $A \in \mathcal{M}(K)$, we know that the positive entries in each quasi-diagonal normalized submatrix B lie in the interval $[1, K]$. Hence we have $w(B) > 1$, whenever B_0 has more than K nonzero elements. So it suffices to check the condition of Theorem 1 for quasi-diagonal normalized submatrices with at most $K + 1$ rows and $K + 2$ columns, which can be done in polynomial time for fixed K.

A further specialization yields the characterization of greedy (0, 1)-matrices as "totally balanced" matrices due to Hoffman, Kolen, and Sakarovitch (1985).

COROLLARY 2. *Let A be a $(0,1)$-matrix. Then A is greedy if and only if neither of the following two matrices occurs as a submatrix:*

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

Reasoning similarly as in Corollary 1, it is also easy to recognize nonnegative greedy matrices A with arbitrary coefficients if we bound the number of rows or the number of columns. Let us turn to the recognition complexity for general greedy matrices. To be more precise, consider the following decision problem.

Instance. A nonnegative matrix $A \in \mathbb{Q}^{m \times n}$.

Question. Does A contain an infeasible quasi-diagonal normalized submatrix?

THEOREM 2. *The decision problem above is NP-complete.*

Proof. Since infeasibility of a given quasi-diagonal submatrix can be checked in polynomial time, the decision problem is in NP. To prove that it is NP-complete it suffices to show that it is at least as hard as the problem of deciding whether a graph G with n nodes has a stable set of size $k+1$ (cf. Garey and Johnson (1979)).

We associate with G the $(n \times n)$-matrix $A_0 = (a_{ij})$, where

$$a_{ij} = \begin{cases} n^{j+1} & \text{if } i < j \text{ and } ij \text{ is an edge in } G, \\ k + \frac{1}{2} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Assuming $n > k \geq 2$, consider now the matrix $A = [\mathbf{1}|A_0]$. Clearly, if G contains a stable set of size $k+1$, A contains an infeasible quasi-diagonal submatrix $B = [\mathbf{1}|B_0]$, where the nonzero elements of B_0 form a subset of the diagonal of A_0.

Conversely, let $B = [\mathbf{1}|B_0]$ be a normalized quasi-diagonal submatrix of A. Suppose that G has no stable set of size $k+1$. We claim $w(B) \leq 1$. To verify the claim, we distinguish two cases.

If B is a normalized submatrix of the submatrix A_0, then every nonzero entry of B_0 has size at least n. So $w(B) \leq n \cdot (1/n) = 1$.

In the other case, B_0 is a submatrix of A_0. Because G has no stable set of size $k+1$, B_0 contains at most k elements from the diagonal of A_0. Hence

$$w(B) \leq \frac{k}{k+\frac{1}{2}} + \sum_{j=1}^{n} \frac{1}{n^{j+1}} \leq \frac{2k}{2k+1} + \frac{1}{n^2 - n} \leq 1. \qquad \square$$

4. Remarks and open problems. While Theorem 1 characterizes nonnegative matrices A that are greedy relative to any *box constraints* $0 \leq x \leq h$, it is interesting to know the structure of general "greedy" matrices. Hoffman (1992) gives a characterization of (0, 1)-matrices A for which the greedy algorithm solves the linear program

$$\max \mathbf{1}^T x \quad \text{s.t. } Ax \leq b,\ x \geq 0,$$

and its dual relative to any b. An extension of this result to arbitrary nonnegative matrices is open.

We finally mention a related problem. If x^* is the greedy solution to the nonnegative linear program

$$\max c^T x \quad \text{s.t. } Ax \leq b,$$

it is not hard to see that x^* is a basic solution. Is there a "simple" way to decide whether x^* is optimal? For example, if x^* is nondegenerate, it suffices to consider the basis B associated with x^* and to check whether $y = c^T B^{-1}$ is nonnegative, i.e., whether the vector $c^T B^{-1}$ yields a dually feasible solution (cf. Kovalev and Vasilkov (1993)).

REFERENCES

J. EDMONDS (1970), *Submodular functions, matroids and certain polyhedra*, in Combinatorial Structures and Their Applications, R. Guy et al., eds., Gordon and Breach, New York, pp. 69–87.

M. R. GAREY AND D. S. JOHNSON (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA.

A. J. HOFFMAN (1985), *On greedy algorithms that succeed*, in Surveys in Combinatorics, I. Anderson, ed., Cambridge University Press, Cambridge, pp. 97–112.

A. J. HOFFMAN (1992), *On simple combinatorial optimization problems*, Discrete Math., 106/107, pp. 285–289.

A. J. HOFFMAN, A. W. J. KOLEN, AND M. SAKAROVITCH (1985), *Totally-balanced and greedy matrices*, SIAM J. Alg. Discrete Methods, 6, pp. 721–730.

M. M. KOVALEV AND D. M. VASILKOV (1993), *The canonical order and optimization problems*, preprint, Fac. of Applied Mathematics, Belarusian State University, Minsk; Z. Oper. Res., 1996.

SIAM J. DISC. MATH.
Vol. 9, No. 1, pp. 7–18, February 1996

002

EFFICIENT GOSSIPING BY PACKETS IN NETWORKS WITH RANDOM FAULTS *

KRZYSZTOF DIKS† AND ANDRZEJ PELC‡

Abstract. Every node of a communication network has a constant size value which should be made known to all other nodes. Nodes and links fail independently with constant probabilities $p < 1$ and $q < 1$, respectively. Faults are permanent and of crash type: a faulty link does not transmit messages and a faulty node neither sends nor receives messages. In a unit of time, every node can send a packet of information to at most one neighbor and receive a packet from at most one neighbor. The size of each packet does not exceed $b(n)$, where n is the number of nodes. For every $\eta > 0$ we present an algorithm to exchange values between all fault-free nodes of an n-node network in time $O(\frac{n}{b(n)} + \log n)$, with probability exceeding $1 - n^{-\eta}$, for sufficiently large n. This order of magnitude of running time is optimal.

Key words. fault, gossiping, packet, random

AMS subject classifications. 68C05, 68C25

1. Introduction.

Gossiping is an important and well-studied problem in network communication. Every node of a communication network has a value which should be made known to all other nodes. Algorithms for gossiping (and closely related broadcasting) working fast and/or using few messages have been proposed by many researchers (see [14] for an extensive bibliography).

Recently, growing attention has been devoted to broadcasting and gossiping in the presence of faulty links and/or nodes [2]–[12], [16]–[17]. While the classic approach assumes an upper bound on the total number of faults and their worst case location [2], [11]–[12], probabilistic models, where links and/or nodes fail independently with constant probability, have recently gained prominence [3]–[10], [16]–[17]. Two alternative assumptions are usually made about faults: crash faults mean that a faulty component does not send or transmit any messages [2], [4], [6]–[12], [17], while Byzantine faults mean arbitrary, even malicious, behavior of faulty components [3], [5], [16].

In this paper we study gossiping under the assumption that nodes and links of the network fail independently with constant probabilities $p < 1$ and $q < 1$, respectively. Faults are permanent and of crash type. Since nodes can fail, the aim of gossiping becomes the exchange of values between fault-free nodes. As always in probabilistic models, we seek gossiping algorithms working with high probability: exceeding $1 - n^{-\eta}$, for n-node networks and a fixed positive constant η.

We assume that, in a unit of time, each node can send a message to at most one neighbor and receive a message from at most one neighbor. We generalize the classic model (cf. [14]) in which a node can send to a neighbor all information it currently has in a unit of time: we assume that a packet of size not exceeding $b(n)$, where n is the

* Received by the editors September 14, 1993; accepted for publication (in revised form) April 5, 1995.

†Instytut Informatyki, Universytet Warszawski, Banacha 2, 02-097 Warszawa, Poland and Département d'Informatique, Université du Québec à Hull, C. P. 1250, succursale B, Hull, Québec J8X 3X7, Canada. The research of this author was partially supported by a Natural Sciences and Engineering Research Council of Canada International Fellowship and grant KBN 2-2043-92-03.

‡Département d'Informatique, Université du Québec à Hull, C. P. 1250, succursale B, Hull, Québec J8X 3X7, Canada. The research of this author was partially supported by Natural Sciences and Engineering Research Council of Canada grant OGP 0008136.

number of nodes, can be sent in a unit of time. For simplicity we assume that node values are 0 or 1, although any constant length values would increase the execution time of the algorithm by only a constant factor, thus leaving our result unchanged. For $b(n) \geq n$ our model coincides with the classic one, while for constant $b(n)$ it is equivalent to the linear model (cf. [15]) with negligible start-up time.

Our algorithm is synchronous (a global clock is assumed) and nonadaptive. The latter means that all transmissions are scheduled in advance: before starting the execution, it is determined which nodes communicate in a given time unit; we do not assume the ability of nodes to choose message recipients depending on the success or failure of previous communications.

We assume that every pair of nodes can communicate directly, i.e., the underlying network is complete. It turns out, however, that our algorithm uses only links of a specific, much sparser network of logarithmic degree. The topology of this network will follow from our presentation, and we will indicate how our algorithm can be modified to work for a large class of underlying networks of logarithmic degree (and for all networks containing them as subgraphs).

For every $\eta > 0$, we design an algorithm to exchange values between all fault-free nodes. Our algorithm works in time $O(\frac{n}{b(n)} + \log n)$, with probability exceeding $1 - n^{-\eta}$, for a sufficiently large number n of nodes in the network. We count total execution time, including transmissions and local computations in nodes. This order of magnitude is clearly optimal under our assumptions, even without faults: logarithmic time is needed to broadcast a single bit and time $\Omega(\frac{n}{b(n)})$ is needed to read all values by one node. Our result should be compared to that from [6], where a broadcasting algorithm working in logarithmic time has been given under the same assumptions. On the other hand, in [8], linear time gossiping has been described assuming fault-free nodes and random faults of individual transmissions (as opposed to permanent link failures).

Two main techniques are used. The first concerns gossiping by packets of size 1: messages are sent along edges of expander graphs to ensure information exchange in linear time with high probability. The second is a modification of the nonadaptive broadcasting algorithm from [6] and is used to gossip in subnetworks of size not exceeding that of the packet. We describe in detail and analyze the first technique, which is the main contribution of this work, and briefly indicate the required modifications of [6] to our present setting.

The paper is organized as follows. In §2 we give some preliminary notions and results to be used later. Section 3 is devoted to the description of our algorithm for $b(n) = 1$ under the additional assumption that node and link failure probabilities are sufficiently small. In §4 we prove that our gossiping algorithm for $b(n) = 1$ works correctly with probability exceeding $1 - n^{-\eta}$, for sufficiently large n. In §5 we show how to modify our algorithm for arbitrary failure probabilities. Finally, in §6 we describe the gossiping algorithm for packets of arbitrary size.

2. Preliminaries. For any finite set X, $|X|$ denotes the number of elements of X. For a real x, $\lceil x \rceil$ denotes the smallest integer greater than or equal to x, and $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. We use $\log x$ for the logarithm with base 2. For any graph G and any set X of nodes of G, $\Gamma_G(X)$ denotes the set of all neighbors of nodes from X in graph G. If $G = (A, B, E)$ is a bipartite graph with sets of nodes A, B and set of edges E, G is called an (α, β, n, d)-expander if $|A| = |B| = n$, the degree of each node is d, and, for all $X \subset A$ such that $|X| \leq \alpha n$, we have $|\Gamma(X)| \geq \beta |X|$.

We will use the following result (cf. [1]).

PROPOSITION 2.1. *For any $\alpha < 1$ and β such that $\alpha\beta < 1$ there is an explicit construction of an (α, β, m, d)-expander with $d \leq 8\beta(1-\alpha)/(1-\alpha\beta)$.* □

The following version of the Chernoff bound has been derived in [13]. It estimates the probability that at least r successes are obtained in a series of n Bernoulli trials with success probability p. Let S denote the number of successes in such a series.

PROPOSITION 2.2. $\Pr(S \geq r) \leq 2^{-r}$, *for* $r \geq 6pn$. □

Let η be a fixed positive constant and b: $N \to N$ a function. An algorithm is called η-safe $b(n)$-gossiping if it exchanges values between all fault-free nodes of an n-node network using packets of size $b(n)$, with probability exceeding $1 - n^{-\eta}$, for sufficiently large n.

3. Gossiping with packets of size 1. In this section we describe a gossiping algorithm for packets of size 1 under the additional assumption that failure probabilities are sufficiently small. In §5 we will show how this assumption can be removed.

We first define networks that support our algorithm. Let c be a sufficiently large constant (depending on failure probabilities but not on the number of nodes) and let $m = \lceil c \log n \rceil$, where n is the number of nodes. For simplicity we assume that m divides n; otherwise the algorithm can be easily modified. We partition the set of all nodes into $k = n/m$ pairwise disjoint sets $V_0, \ldots, V_{k-1}$ of size m. These sets will be called groups. Nodes $v \in V_i$ and $w \in V_j$ are joined by a link if and only if $|i-j|$ is 1 or $k-1$ (cf. [3], [5], where similar graphs called "fat rings" have been used). We will use the convention that operations on indices of groups V_i are performed modulo k.

Let $\alpha > 0$ and $\beta > 1$ be such that $\alpha\beta < 1$. Let d be a positive integer for which an (α, β, m, d)-expander is explicitly constructible, by Proposition 2.1. Let p_0 and q_0 be positive constants such that $\varepsilon = 6p_0$ and $\delta = 6q_0$ satisfy the following conditions: $\delta < 1, \varepsilon + \alpha < 1$, and $\beta\alpha/2 - \varepsilon - \alpha/2 - \delta\alpha d > 0$. We describe our algorithm assuming that node failure probability p is less than p_0 and link failure probability q is less than q_0.

We start by presenting the idea of the algorithm. During its execution, every node stores already obtained values of other nodes in a vector A whose terms are initialized to a special symbol *. When a node is supposed to transmit a value of another node, it transmits the symbol from the appropriate term of its vector A (0, 1, or *), thus using two bits (and two time units) per symbol. Upon completion of the algorithm all nodes have the symbol * as the value of faulty nodes only, and correct values, 0 or 1, of fault-free nodes, with high probability.

The algorithm works in three stages. In stage 1 all nodes of group V_i send their values to all other nodes in this group, using consecutive nodes in V_{i+1} as intermediaries. This is done in time $O(m^2)$. It will be proved that upon completion of this stage, every fault-free node knows the values of all other fault-free nodes in its group, with high probability.

The aim of stage 2 is to transmit information on all fault-free nodes in the network to many nodes in each group, with high probability. For every $i < k$ we use an (α, β, m, d)-expander $G_i = (V_i, V_{i-1}, E_i)$. (Expanders were used in a similar context in [1].) Links in E_i are partitioned into d disjoint perfect matchings $F_i^1, \ldots, F_i^d$. For every $v \in V_i$ we denote by $a(v, r)$ the node in V_{i-1} matched with v by F_i^r. Similarly, for every $w \in V_{i-1}$ we denote by $b(w, r)$ the node in V_i matched with w by F_i^r. Stage 2 operates in $k-1$ rounds. Before round $j = 0, \ldots, k-2$, some fault-free nodes in each group are called informed—they have value 1 assigned to a special register D, and other fault-free nodes are not informed and their value of D is 0. Before

round 0, D is set to 1 for all nodes. During round j, every informed node $w \in V_{i-1}$ informs nodes $b(w,r)$, for $r = 1, \ldots, d$, in d consecutive steps (in each step nodes act in parallel). First it sends its value D and then all values of nodes in V_{i-1-j} (some of these values may be *). Each step takes time $O(m)$ and thus each round is executed in time $O(m)$ as well. A node $v \in V_i$, which had register D set to 1 before round j and obtained value $D = 1$ from a node $a(v,r)$ during this round, keeps its register D set to 1 and is considered informed after round j. Otherwise, node v sets its register D to 0 and remains uninformed until the end of stage 2. It will be proved that upon the completion of stage 2, at least $(1 - \varepsilon - \alpha/2)m$ fault-free nodes in each group have register D set to 1 and know the values of all other fault-free nodes in the network, with high probability. Stage 2 consists of $k - 1$ rounds, each taking time $O(m)$, for a total time $O(n)$.

Stage 3 is devoted to informing nodes that have not yet obtained all information after stage 2: those that have register D set to 0. Every such node $v \in V_i$ obtains information from informed nodes in V_{i+1}. This is done in three steps. First, all nodes from V_{i+1} send the value of their register D to all nodes in V_i, in time $O(m)$. In the second step, every node $v \in V_i$ requests a portion of total information from all nodes in V_{i+1} from which it obtained $D = 1$. We shall prove that, with high probability, there are at least $m_0 = (1 - \delta)(1 - \varepsilon - \alpha/2)m$ such nodes in V_{i+1}. Node v sends consecutive integers $i = 1, \ldots, m_0$ to the first m_0 of these nodes in V_{i+1} (for simplicity we assume that m_0 is an integer: it is easy to modify the algorithm in the general case); $\log m$ time units are reserved for the transmission of each integer. Since transmissions must be scheduled in a nonadaptive way, all nodes from V_i must be able to send these integers to all nodes in V_{i+1}. This can be done in time $O(m \log m)$. The third step consists of informed nodes from V_{i+1} sending the requested portion of information to nodes $v \in V_i$ which sent the request. Every node divides the set of all nodes into m_0 sets $P_1, \ldots, P_{m_0}$ of size $l_0 = n/m_0$. The partition is identical for all nodes. (Again, without loss of generality, we assume that l_0 is an integer.) If node $v \in V_i$ sent integer j to an informed node w, node w interprets this as a request for portion P_j of information. It sends values of nodes from P_j to node v in the third step, which (for all possible v) takes time $O(ml_0) = O(n)$. This concludes stage 3, which takes total time $O(n)$.

It should be noted that, although some actions in the algorithm execution are conditional, the algorithm remains nonadaptive in the sense that all transmissions to be performed in a given time unit are scheduled in advance. A node may choose the message to be sent or refrain from sending any message at a given time, but a fixed receiver for a particular node and time unit is reserved in advance.

We now proceed to a more formal description of our gossiping algorithm. For every node x, A_x denotes the content of register A in this node. We use the elementary procedure SEND(x, A, y, B, t) for adjacent nodes x, y and registers A in node x and B in node y that consists of the following actions:

1. x sends a t-bit long message A_x to y.
2. y assigns A_x to its register B if it got the message; otherwise it does nothing.

The parameter t indicates the length of the message transmitted in a particular SEND procedure and the time reserved for its execution. Since possible node values that are transmitted during algorithm execution are 0, 1, or $*$, to send them we need $t = 2$.

The algorithm is preceded by the initialization of all registers. We now explain the role of each register and specify how it is initialized. Every node $v \in V_i$ of the network uses the following registers:

- Val: used to store the original value of the node. Val is initialized to 0 or 1 and never changed.
- $A[x]$, for all nodes x: used to store the value of node x obtained during execution. $A[x]$ is initialized to $*$, for $x \neq v$, and to Val, for $x = v$.
- $B[x]$, for all nodes $x \in V_{i-1}$: used to store the value of node x from the neighboring group in the first part of stage 1. $B[x]$ is initialized to $*$.
- $C[x, y]$, for all nodes $x \in V_i, y \in V_{i+1}$: used to store the value of node x obtained from node y in the second part of stage 1. $C[x, y]$ is initialized to $*$.
- D: used to determine if the node is informed. D initialized to 1.
- $M[r]$, for $r = 1, \ldots, d$: used to receive the content of register D from a node in V_{i-1} in stage 2. $M[r]$ is initialized to 0.
- $N[r]$, for $r = 1, \ldots, m$: used to receive the content of register D from a node in V_{i+1} in stage 3. $N[r]$ is initialized to 0.
- S: used to keep the current index of the requested set P_i. S is initialized to 1.
- $Z[r]$, for $r = 1, \ldots, m$: used to receive the index of the set P_i of nodes whose values are requested in stage 3. $Z[r]$ is initialized to 0.

We now describe three subroutines used in our gossiping algorithm. The first of them corresponds to stage 1 of the algorithm: values are exchanged among nodes of the same group via the neighboring group. For $i < k$, let $H_i^1, \ldots, H_i^m$ be a fixed partition of all links between V_i and V_{i+1} into disjoint perfect matchings. For every $v \in V_i$, we denote by $h(v, r)$ the node in V_{i+1} matched with v by H_i^r. Similarly, for every $w \in V_{i+1}$, we denote by $g(w, r)$ the node in V_i matched with w by H_i^r.

procedure EXCHANGE (i)
  for $r \leftarrow 1$ **to** m **do**
    for all $v \in V_i$ **in parallel do**
      SEND(v, Val, $h(v, r), B[v], 2$)
  for $r \leftarrow 1$ **to** m **do**
    for all $v \in V_i$ **do**
      for all $w \in V_{i+1}$ **in parallel do**
        SEND($w, B[v], g(w, r), C[v, w], 2$)
        if $C[v, w]_{g(w,r)} \neq *$ **then do**
          $A[v]_{g(w,r)} \leftarrow C[v, w]_{g(w,r)}$.

The first external loop takes time $O(m)$ and the second takes time $O(m^2)$. Execution time of procedure EXCHANGE is $O(m^2)$.

The next subroutine corresponds to one round of stage 2. We keep notation from our informal description.

procedure NEXT (i, round)
  for $r \leftarrow 1$ **to** d **do**
    for all $v \in V_{i-1}$ **in parallel do**
      if $D_w = 1$ **then do**
        SEND($w, D, b(w, r), M[r], 1$)
        for all $v \in V_{i-1-\text{round}}$ **do**
          SEND($w, A[v], b(w, r), A[v], 2$)
  for all $u \in V_i$ **in parallel do**
    if ($M[r]_u = 0$ for all $r \leq d$) **then do**
      $D_u \leftarrow 0$.

The first external loop takes time $O(m)$ and the second takes constant time. Execution time of procedure NEXT is $O(m)$.

The final subroutine corresponds to stage 3: informing nodes not yet informed. The meaning of $h(v,r)$ and $g(w,r)$ is as before.

procedure FINAL(i)
  for $r \leftarrow 1$ **to** m **do**
    for all $v \in V_{i+1}$ **in parallel do**
      SEND($w, D, g(w,r), N[r], 1$)
  for $r \leftarrow 1$ **to** m **do**
    for all $v \in V_i$ **in parallel do**
      if $N[r] = 1$ **and** $S_v \leq m_0$ **then do**
        SEND($v, S, h(v,r), Z[r], \log n$)
        $S_v \leftarrow S_v + 1$
  for $r \leftarrow 1$ **to** m **do**
    for all $v \in V_{i+1}$ **in parallel do**
      if $Z[r]_w > 0$ **then do**
        for all $u \in P_{z[r]_w}$ **do**
          SEND($w, A[u], g(w,r), A[u], 2$).

The first external loop takes time $O(m)$, the second takes time $O(\log n \log m)$, and the third takes time $O(n)$. The execution time of procedure FINAL is $O(n)$.

Our algorithm can now be written as follows.

algorithm GOSSIP
  for all $i < k$ **in parallel do**
    EXCHANGE(i)
  for round $\leftarrow 0$ **to** $k-2$ **do**
    for all $i < k$ **in parallel do**
      NEXT(i, round)
    for all $i < k$ **in parallel do**
      FINAL(i).

Execution time of algorithm GOSSIP is $O(m^2 + km + n) = O(n)$.

We conclude this section by pointing out that, although our algorithm has been described for particular networks of logarithmic degree, a much wider class of networks could be used with only minor algorithm modifications. Let H be any k-node graph of bounded degree (independent of the number of nodes). Replace every node of H by a group of $\lceil c \log k \rceil$ nodes, for sufficiently large c. For adjacent nodes v and w of H, join all nodes of the group corresponding to v with all nodes of the group corresponding to w. The obtained network H^* (and every network containing H^* as a subnetwork) supports a suitably modified gossiping algorithm. Instead of operating on the cycle $V_0, \ldots, V_{k-1}$ of groups (as algorithm GOSSIP does), the modified algorithm would operate on the tree of groups obtained from a spanning tree of H. The details of modification are left to the reader.

4. Probability of correctness. In this section we prove that if failure probabilities p and q satisfy the conditions imposed at the beginning of §3, then the algorithm GOSSIP is η-safe, for a sufficiently large constant c. Fix $\eta > 0$. Let $p_0, q_0, \varepsilon, \delta, \alpha$, and β be as required in §3. We assume that the constant c (for which $m = \lceil c \log n \rceil$) is sufficiently large and depends on p, q, and η, but not on n.

LEMMA 4.1. *Assume that $p < p_0$. Let E_1 be the event that in each group there are at most εm faulty nodes. Then*

$$\Pr(E_1) > 1 - n^{-\eta/4},$$

for sufficiently large n.

Proof. The probability that in a given group V_i there are at least εm faulty nodes is at most $2^{-\varepsilon m}$, in view of Proposition 2.2. Hence,

$$\Pr(E_1) \geq 1 - k \cdot 2^{-\varepsilon m} > 1 - n^{-\eta/4},$$

for sufficiently large c and n. □

LEMMA 4.2. *Let E_2 be the event that, upon the completion of stage 1 of the algorithm, all fault-free nodes know the values of all other fault-free nodes in their group. Then*

$$\Pr(E_2) > 1 - n^{-\eta/4},$$

for sufficiently large n.

Proof. Fix distinct nodes $u, v \in V_i$. The probability that at least one path $u - w - v$, where $w \in V_{i+1}$, has both links and the middle node fault free is equal to

$$1 - (1 - (1-q)^2(1-p))^m.$$

Hence,

$$\Pr(E_2) \geq 1 - km^2(1 - (1-q)^2(1-p))^m > 1 - n^{-\eta/4},$$

for sufficiently large c and n. □

Before formulating the next lemma we need to introduce the following definition. Let $v \in V_i$. A thread of length $j = 0, 1, \ldots, k-1$, ending at node v, is a sequence $v = v_j, v_{j-1}, \ldots, v_0$, where, for every $s \leq j, v_{j-s} \in V_{i-s}$ and link $v_{j-s+1} - v_{j-s}$ is an edge of the expander G_{j-s+1}. A thread is open if all its links and nodes are fault free.

LEMMA 4.3. *Assume that $p < p_0$ and $q < q_0$. Let E_3 be the event that at least $(1 - \varepsilon - \alpha/2)m$ nodes in each group are ends of an open thread of length $k-1$. Then*

$$\Pr(E_3 | E_1) > 1 - n^{-\eta/4},$$

for sufficiently large n.

Proof. Assume that event E_1 holds. Let F_j, for $j < k$, be the event that at least $(1 - \varepsilon - \alpha/2)m$ nodes in each V_i are ends of an open thread of length j. We will prove by induction on j that $\Pr(F_j) \geq 1 - jk2^{-\delta d\lceil m\alpha/2 \rceil}$.

In view of event E_1, at least $(1 - \varepsilon)m$ nodes in each V_i are fault free and all of them are ends of an open thread of length 0. Thus $\Pr(F_0) = 1$. Suppose that $\Pr(F_j) \geq 1 - jk2^{-\delta d\lceil m\alpha/2 \rceil}$ and assume that F_j holds. Fix $i < k$. Let X_0 be the set of fault-free nodes in V_i that are not ends of an open thread of length $j+1$. Suppose that $|X_0| \geq m\alpha/2$. Take $X \subset X_0$ of size $\lceil m\alpha/2 \rceil$. For sufficiently large $m, |X| \leq \alpha m$. Consider any node $v \in X$. For every node $w \in V_{i-1}$ linked with v by an edge of the expander G_i, either w is not an end of an open thread of length j or the link $v - w$ is faulty.

Let Y be the set of edges of the expander G_i incident to nodes from X. Thus $|Y| = d\lceil m\alpha/2 \rceil$. Let $F^*(i)$ be the event that at most $\delta d\lceil m\alpha/2 \rceil$ links from Y are faulty. By Proposition 2.2, $\Pr(F^*(i)) \geq 1 - 2^{-\delta d\lceil m\alpha/2 \rceil}$. Assume that $F^*(i)$ holds. By the property of the expander $G_i, |\Gamma_{G_i}(X)| \geq \beta\lceil \alpha m/2 \rceil$. At least $\beta\lceil \alpha m/2 \rceil - \varepsilon m - m\alpha/2$ nodes in $\Gamma_{G_i}(x)$ are ends of an open thread of length j. Each of those nodes is linked by at least one link from Y to a node in X. At most $\delta d\lceil m\alpha/2 \rceil$ of these links are faulty. For sufficiently large m we have

$$\beta\lceil \alpha m/2 \rceil - \varepsilon m - m\alpha/2 - \delta d\lceil \alpha m/2 \rceil \geq \beta\alpha m/2 - \varepsilon m - m\alpha/2 - \delta dm\alpha,$$

which is positive by the choice of our constants. Thus, at least one node $w \in V_{i-1}$ is an end of an open thread of length j linked to a node $v \in X$ by a fault-free link. Hence, v is an end of an open thread of length $j+1$, which contradicts the definition of X.

This implies that if events F_j and $F^*(i)$ hold, then $|X_0| < m\alpha/2$ and hence at least $(1-\varepsilon-\alpha/2)m$ nodes in V_i are ends of an open thread of length $j+1$. Let F^* be the intersection of events $F^*(i)$ over all $i < k$. Hence $\Pr(F^*) \geq 1 - k \cdot 2^{-\delta d\lceil m\alpha/2\rceil}$. We have

$$F_{j+1} \supset F_j \cap F^*;$$

hence, by the inductive hypothesis,

$$\Pr(F_{j+1}) \geq 1 - (j+1)k \cdot 2^{-\delta d\lceil m\alpha/2\rceil}.$$

By induction we get

$$\Pr(F_{k-1}) \geq 1 - k^2 2^{-\delta d\lceil m\alpha/2\rceil}$$

(assuming that E_1 holds).

Since $E_3 = F_{k-1}$ we have

$$\Pr(E_3|E_1) > 1 - n^{-\eta/4},$$

for sufficiently large c and n. □

LEMMA 4.4. *Assume that $p < p_0$ and $q < q_0$. Let E_4 be the event that, upon the completion of stage 2 of the algorithm, at least $(1-\varepsilon-\alpha/2)m$ fault-free nodes in each group know the values of all other fault-free nodes. Then*

$$\Pr(E_4) > 1 - 3n^{-\eta/4},$$

for sufficiently large n.

Proof. Since $E_4 \supset E_1 \cap E_2 \cap E_3$, the conclusion follows from Lemmas 4.1, 4.2, and 4.3. □

The following theorem is the main result of this section.

THEOREM 4.5. *Assume that $p < p_0$ and $q < q_0$. Then the algorithm GOSSIP is η-safe.*

Proof. Assume that event E_4 holds. For every $i < k$, let $W_i \subset V_i$ be the set of those fault-free nodes in V_i that know the values of all other fault-free nodes, after stage 2 of the algorithm. Thus $|W_i| \geq (1-\varepsilon-\alpha/2)^m$, for all $i < k$. Fix $i < k$ and consider $v \in V_i$. Let F_v be the event that at least $(1-\delta)|W_{i+1}|$ links joining v with nodes from W_{i+1} are fault free. By Proposition 2.2, $\Pr(F_v) \geq 1 - 2^{-\delta|W_{i+1}|} \geq 1 - 2^{-\delta(1-\varepsilon-\alpha/2)m}$. Let F be the intersection of events F_v over all nodes v in the network. Thus,

$$\Pr(F) \geq 1 - n2^{-\delta(1-\varepsilon-\alpha/2)m} > 1 - n^{-\eta/4},$$

for sufficiently large c and n. If events E_4 and F hold, every node $v \in V_i$ can communicate during stage 3 with at least $m_0 = (1-\delta)(1-\varepsilon-\alpha/2)m$ nodes from V_{i+1} which already had values of all fault-free nodes after stage 2, for all $i < k$. This implies that, assuming $E_4 \cap F$, all fault-free nodes will get values of all other fault-free nodes after stage 3. Since $\Pr(E_4) > 1 - 3n^{-\eta/4}$, by Lemma 4.4, and $\Pr(F) > 1 - n^{-\eta/4}$, we get $\Pr(E_4 \cap F) > 1 - n^{-\eta}$, for sufficiently large n, which shows that algorithm GOSSIP is η-safe. □

5. Modifications for arbitrary failure probabilities. In this section we show how our 1-gossiping algorithm should be modified in case of arbitrary node failure probability $p < 1$ and link failure probability $q < 1$.

Let p_0 and q_0 be as described in §3. Let c be a constant sufficiently large that our previous arguments hold for node and link failure probabilities less than p_0 and q_0, respectively. Let $p < 1$ and $q < 1$ be arbitrary node failure and link failure probabilities. Define K_0 to be the least integer satisfying the following conditions:

- $K_0/(1-6p)$ is an integer.
- $2^{-K_0 6p/(1-6p)} < p_0$.
- $2(K_0 \frac{1}{1-6p})q^{K_0} < q_0$.

Let $L_0 = K_0/(1-6p)$. A supernode is a set of L_0 nodes. A superlink joining two disjoint supernodes S_1 and S_2 is the set of all links $u - v$, for $u \in S_1, v \in S_2$. A supernode is called faulty if less than K_0 of its nodes are fault free. A superlink joining supernodes S_1 and S_2 is called faulty if there exists a node $v \in S_1$ which is not joined by a fault-free link with any fault-free node in S_2 or there exists a node $w \in S_2$ which is not joined by a fault-free link with any fault-free node in S_1. Thus, a supernode is faulty if at least $L_0 - K_0 = 6pL_0$ of its nodes are faulty. The probability of this event does not exceed $2^{-6pL_0} < p_0$. The probability that a superlink joining S_1 and S_2 is faulty does not exceed $2L_0 q^{K_0} < q_0$.

We first slightly modify the network supporting our algorithm. Let $m = \lceil c \log n \rceil$, as before. For simplicity assume that $m' = mL_0$ divides n (it is easy to adjust the argument in the general case). We partition all nodes into k' groups $V'_0, \ldots, V'_{k'-1}$ of size m'.

The algorithm is modified as follows. Stage 1 is executed as before. In stage 2 we partition each group V'_i into m disjoint supernodes and consider (α, β, m, d)-expanders G'_i on supernodes from V'_i and V'_{i-1}: nodes of G'_i are supernodes and links of G'_i are superlinks joining them; G'_i is isomorphic to the expander G_i used in §3. Procedure NEXT is now executed using G'_i instead of G_i: whenever, in the original version of NEXT, node $v \in V_{i-1}$ communicated with node $w \in V_i$, in the modified version of NEXT, corresponding supernodes v' and w' communicate; that is, every node in v' sends the respective message to every node in w'. Since the size of the supernodes is constant, execution time may increase only by a constant factor. Since supernode and superlink failure probabilities are less than p_0 and q_0, respectively, our previous arguments show that, after the modified stage 2, at least $(1 - \varepsilon - \alpha/2)m$ fault-free supernodes in each group V'_i know that the values of all fault-free nodes, with high probability. (A fault-free supernode S is said to know values of all fault-free nodes if some fault-free node $v \in S$ knows all these values.) It follows that, after the modified stage 2, at least $(1 - \varepsilon - \alpha/2)m$ fault-free nodes in each group V'_i know the values of all other fault-free nodes, with high probability. Now stage 3 can be executed as before, with one modification: instead of $m_0 = (1-\delta)(1 - \varepsilon - \alpha/2)m$ we should take $m'_0 = m_0/L_0$. This can increase execution time by only a constant factor. An argument similar to that from the proof of Theorem 4.5 can be used to show that after (the modified) stage 3, all fault-free nodes know the values of all other fault-free nodes, with probability exceeding $1 - n^{-\eta}$, for sufficiently large n. This proves the following result.

THEOREM 5.1. *For any node failure and link failure probabilities $p < 1, q < 1$, and for any constant $\eta > 0$, η-safe 1-gossiping among n nodes can be done in time $O(n)$.* □

6. Gossiping with arbitrary packets. We start with the description of gossiping when packets are large. Fix $\eta > 0$ and let c be a constant depending on η; c will have to be sufficiently large to ensure η-safe gossiping. We first construct an η-safe $b(n)$-gossiping algorithm working in time $O(\frac{n}{b(n)} + \log n)$ under the assumption that $b(n) \geq 2\lceil c \log n \rceil$. Suppose that n is large enough to satisfy $\lfloor n/\log n \rfloor \geq 2\lceil c \log n \rceil$. Let $r = \min(b(n), \lfloor n/\log n \rfloor)$. For simplicity, assume that r divides n and $\lceil c \log n \rceil$ divides r (it is easy to modify the algorithm in the general case).

Partition the set V of all nodes into $s = n/r$ pairwise disjoint sets $V_0, \ldots, V_{s-1}$ of size r and partition each set V_i into $t = r/\lceil c \log n \rceil$ subsets $V_i^k, k = 0, \ldots, t-1$ of size $\lceil c \log n \rceil$. By assumption, $t \geq 2$.

Our algorithm works in two phases. In the first phase nodes exchange their values in each set V_i independently and in the second phase information is exchanged between sets V_i.

Phase I is performed in all sets V_i in parallel. We describe it for a fixed set V_i. It is very similar to the algorithm NBA from [6]. The only difference is this: NBA consisted of three identical stages, while in our present phase I there are four identical stages, each of them as in NBA. We refer the reader to [6] for a detailed description of a stage and merely point out the role of the additional stage in the present setting. As in [6], sets V_i^k are organized in a complete binary tree of size $t \geq 2$ with V_i^0 in the root. Every node in the set V_i^j is connected with all nodes in the parent set. The role of the additional stage is to transmit the value of each fault-free node in V_i to some fault-free node in the root V_i^0, with high probability. The aim of the remaining three stages is identical to that in NBA: broadcasting from V_i^0 to all remaining nodes. Since the size r of V_i does not exceed the size $b(n)$ of a packet, phase I works in time $O(\log n)$, similarly as NBA, and an analysis analogous to that from [6] yields the following lemma.

LEMMA 6.1. *Assume $b(n) \geq 2\lceil c \log n \rceil$. For every $\eta > 0$ and every $i < s$, there exists a constant c such that $(1+\eta)$-safe $b(n)$-gossiping can be done in V_i in time $O(\log n)$.*

Since $s < n$, Lemma 6.1 implies the following corollary.

COROLLARY 6.2. *Assume $b(n) \geq 2\lceil c \log n \rceil$. For every $\eta > 0$ there exists a constant c such that after phase* I *every fault-free node in V_i knows values of all other fault-free nodes in V_i, for all $i < s$, with probability exceeding $1 - n^{-\eta}$, for sufficiently large n.* □

Before describing phase II of our algorithm we consider the following auxiliary problem. Let $W_0, \ldots, W_{s-1}$ be pairwise disjoint sets of size $\lceil c \log n \rceil$. Nodes $v \in W_i$ and $w \in W_j$ are joined by an edge if and only if $|i - j|$ is 1 or $s - 1$ (cf. groups V_i from §3). Each fault-free node in W_i initially has the same information $\alpha(i)$ consisting of a bits. Nodes can exchange packets of size b with their neighbors. The aim of b-GROUP-GOSSIPING $(W_0, \ldots, W_{s-1})$ is that every fault-free node $v \in \cup_{i=0}^{s-1} W_i$ know all information $\alpha(i)$, for $i < s$.

The above problem can be solved by slightly modifying algorithm GOSSIP from §3. Indeed, it is enough to delete its stage 1 (communication inside groups) and allow transmissions of packets of size b in a unit of time. Since initial information at each node now has size a, the algorithm GOSSIP thus modified works in time $O(s\lceil \frac{a}{b} \rceil + \log n)$.

The analysis from §4 yields the following lemma.

LEMMA 6.3. *For every $\eta > 0$ there exists a constant c such that b-GROUP-GOSSIPING $(W_0, \ldots, W_{s-1})$ can be done in time $O(s\lceil \frac{a}{b} \rceil + \log n)$, with probability*

exceeding $1 - n^{-(1+\eta)}$, *for sufficiently large* n. □

Now it is easy to describe phase II of our algorithm: $b(n)$-GROUP-GOSSIPING $(V_0^j, \ldots, V_{s-1}^j)$ is performed for all $j < t$, in parallel. In our case $\alpha(i)$, for $i < s$, is the set of all values of nodes in V_i (each value is 0, 1, or *). Thus $a \in O(r)$ and $s\lceil \frac{a}{b(n)} \rceil \in O(\frac{sr}{b(n)}) = O(\frac{n}{b(n)})$. Consequently, the algorithm works in time $O(\frac{n}{b(n)} + \log n)$.

Corollary 6.2, together with Lemma 6.3, yields the following result.

LEMMA 6.4. *Assume* $b(n) \geq 2\lceil c \log n \rceil$. *For every* $\eta > 0$ *there exists a constant* c *such that* η*-safe* $b(n)$*-gossiping can be done in time* $O(\frac{n}{b(n)} + \log n)$. □

It remains to describe the gossiping algorithm for small packets, i.e., when $b(n) < 2\lceil c \log n \rceil$. In this case we take $r = \lceil c \log n \rceil$ and partition the set V into $s = n/r$ pairwise disjoint sets $V_0, \ldots, V_{s-1}$ of size r. (Again we assume for simplicity that r divides n.) We only change phase I of the algorithm described earlier in this section, leaving phase II as before. As before, the aim of the new phase I is to exchange values inside each set V_i, independently and in parallel. We describe it for a fixed $i < s$. Let E_i be the set of edges of the complete graph on vertex set V_i and let $F_1^i, \ldots, F_{r+1}^i$ be a partition of E_i into $r + 1$ pairwise disjoint matchings. Now phase I consists of two stages, each executed in $r + 1$ steps. In the jth step of each stage every node $u \in V_i$ communicates with the node $v \in V_i$ for which the edge $\{u, v\}$ is in F_j^i. In stage 1, nodes u and v exchange their values (if both are fault free): this takes 1 unit of time. In stage 2 nodes u and v exchange all r values obtained in stage 1, subsequently updating previously stored values: * is first stored as a value of every node w and then replaced by 0 or 1 as soon as this value of w is obtained. Every step of stage 2 takes $\lceil \frac{r}{b(n)} \rceil$ time units and thus phase 1 is now executed in time $O(r + \lceil \frac{r}{b(n)} \rceil \cdot r)$, which is $O(\log^2 n)$ in view of the assumption $b(n) < 2\lceil c \log n \rceil = 2r$. This is dominated by execution time $O(\frac{n}{b(n)})$ of the second (unchanged) phase.

Since every pair of fault-free nodes in $V_i, i < s$, communicates via $r - 2$ distinct intermediary nodes, the probability of exchanging information between such a pair is $1 - (1 - (1-p)(1-q)^2)^{r-2}$. It follows that all pairs of fault-free nodes in V_i exchange information with probability exceeding $1 - r^2(1 - (1-p)(1-q)^2)^{r-2}$, which is larger than $1 - n^{-(1+\eta)}$, for sufficiently large c and n. This implies the following lemma.

LEMMA 6.5. *Assume* $b(n) < 2\lceil c \log n \rceil$. *For every* $\eta > 0$ *there exists a constant* c *such that after phase* I *every fault-free node in* V_i *knows values of all other fault-free nodes in* V_i, *for all* $i < s$, *with probability exceeding* $1 - n^{-\eta}$, *for sufficiently large* n. □

Since phase II and its analysis remain unchanged for $b(n) < 2\lceil c \log n \rceil$, Lemmas 6.4, 6.3, and 6.5 imply our main result, Theorem 6.6.

THEOREM 6.6. *For every* $\eta > 0, \eta$*-safe* $b(n)$*-gossiping can be done in time* $O(\frac{n}{b(n)} + \log n)$.

We conclude the paper by pointing out which networks support our gossiping algorithm for packets of arbitrary size. If $b(n) < 2\lceil c \log n \rceil$, these are networks similar to the network H^* described at the end of §3—with one change: all connections have to be added inside groups. If $b(n) \geq 2\lceil c \log n \rceil$, the supporting networks are somewhat more complicated. The set V of all nodes is divided into sets $V_0, \ldots, V_{s-1}$, each of size $r = \min(b(n), \lfloor n/\log n \rfloor)$. Each set V_i is partitioned into sets $V_i^0, \ldots, V_i^{t-1}$ of size $\lceil c \log n \rceil$. Sets $V_i^0, \ldots, V_i^{t-1}$ are arranged into a complete binary tree, each node $v \in V_i^j$ being connected to all nodes from the parent set. Additionally, sets $V_0^j, \ldots, V_{s-1}^j$, for every $j < t$, are arranged to span a graph H^* as described at the end of §3. (The choice of H may be different for every j.) In both cases the obtained supporting network has

logarithmic degree, and numerous possible choices of the underlying graph H allow a considerable flexibility of network topology.

REFERENCES

[1] S. ASSAF AND E. UPFAL, *Fault-tolerant sorting networks*, SIAM J. Discrete Math., 4 (1991), pp. 472–480.

[2] K. A. BERMAN AND M. HAWRYLYCZ, *Telephone problems with failures*, SIAM J. Alg. Discrete Methods, 7 (1986), pp. 13–17.

[3] P. BERMAN AND A. PELC, *Efficient broadcasting and gossiping with Byzantine link failures*, Rapport de Recherche RR 92/11-11, Université du Québec à Hull, 1992.

[4] D. BIENSTOCK, *Broadcasting with random faults*, Discrete Appl. Math., 20 (1988), pp. 1–7.

[5] D. M. BLOUGH AND A. PELC, *Optimal communication in networks with randomly distributed Byzantine faults*, Networks, 23 (1993), pp. 691–701.

[6] B. S. CHLEBUS, K. DIKS, AND A. PELC, *Sparse networks supporting efficient reliable broadcasting*, in Proc. 20th International Colloquium on Automata, Languages and Programming (ICALP 93), Lund, Sweden, Lecture Notes in Computer Science 700, 1993, pp. 388–397.

[7] ———, *Optimal broadcasting in faulty hypercubes*, Digest of Papers, 21st International Symposium on Fault-Tolerant Computing, 1991, pp. 266–273.

[8] ———, *Fast gossiping with short unreliable messages*, Discrete Appl. Math., 53 (1994), pp. 15–24.

[9] K. DIKS AND A. PELC, *Reliable gossip schemes with random link failures*, in Proc. 28th Ann. Allerton Conf. on Comm. Control and Comp., Allerton, IL, 1990, pp. 978–987.

[10] ———, *Almost safe gossiping in bounded degree networks*, SIAM J. Discrete Math., 5 (1992), pp. 338–344.

[11] L. GARGANO, *Tighter time bounds on fault-tolerant broadcasting and gossiping*, Networks, 22 (1992), pp. 469–486.

[12] R. W. HADDAD, S. ROY, AND A. A. SCHAFFER, *On gossiping with faulty telephone lines*, SIAM J. Alg. Discrete Methods, 8 (1987), pp. 439–445.

[13] T. HAGERUP AND C. RUB, *A guided tour of Chernoff bounds*, Inform. Process. Lett., 33 (1989/90), pp. 305–308.

[14] S. M. HEDETNIEMI, S. T. HEDETNIEMI, AND A. L. LIESTMAN, *A survey of gossiping and broadcasting in communication networks*, Networks, 18 (1988), pp. 319–349.

[15] S. L. JOHNSSON AND C.-T. HO, *Optimum broadcasting and personalized communication in hypercubes*, IEEE Trans. Comput., 38 (1989), pp. 1249–1268.

[16] A. PELC, *Reliable communication in networks with Byzantine link failures*, Networks, 22 (1992), pp. 441–459.

[17] E. R. SCHEINERMAN AND J. C. WIERMAN, *Optimal and near-optimal broadcast in random graphs*, Discrete Appl. Math., 25 (1989), pp. 289–297.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 19–28, February 1996

SHORT RANDOM WALKS ON GRAPHS*

GREG BARNES† AND URIEL FEIGE‡

Abstract. The short-term behavior of random walks on graphs is studied, in particular, the rate at which a random walk discovers new vertices and edges. A conjecture by Linial that the expected time to find $\mathcal{N}$ distinct vertices is $O(\mathcal{N}^3)$ is proved. In addition, upper bounds of $O(\mathcal{M}^2)$ on the expected time to traverse $\mathcal{M}$ edges and of $O(\mathcal{MN})$ on the expected time to either visit $\mathcal{N}$ vertices or traverse $\mathcal{M}$ edges (whichever comes first) are proved.

Key words. random walk, graph, Markov chain

AMS subject classification. 60J15

1. Introduction.

Consider a simple random walk on G, an undirected graph with n vertices and m edges. At each time step, if the walk is at vertex v, it moves to a vertex chosen uniformly at random from the neighbors of v. Random walks have been studied extensively and have numerous applications in theoretical computer science, including space-efficient algorithms for undirected connectivity [4], [8], derandomization [1], recycling of random bits [10], [15], approximation algorithms [6], [12], [17], efficient constructions in cryptography [14], and self-stabilizing distributed computing [11], [16].

Frequently (see, for example, Karger et al. [19] and Nisan et al. [20]), we are interested in $E[T(\mathcal{N})]$, the expected time before a simple random walk on an undirected connected graph, G, visits its $\mathcal{N}$th distinct vertex, $\mathcal{N} \leq n$. The corresponding question for edges is also interesting and arises in the work of Broder et al. [8]: how large is $E[T(\mathcal{M})]$, the expected time before a simple random walk on an undirected connected graph, G, traverses its $\mathcal{M}$th distinct edge, $\mathcal{M} \leq m$? This paper gives upper bounds on $E[T(\mathcal{N})]$ and $E[T(\mathcal{M})]$ for arbitrary graphs. While a great deal was previously known about how quickly a random walk covers the entire graph (see, for example, [2], [4], [7], [9], [18], [22], [23]), little was known about the behavior of a random walk before the vertices are covered. These bounds help fill the gaps in our knowledge of random walks, giving a picture of the rate at which a random walk explores a finite or an infinite graph.

Aleliunas et al. [4] show that the expected time to visit *all* vertices of an arbitrary graph (called the *cover time*) is $O(mn) \leq O(n^3)$. Using this bound, Linial derives a bound for general $\mathcal{N}$ of $E[T(\mathcal{N})] = O(\mathcal{N}^4)$ [19, Lemma 4.1]. Linial (in a personal communication) conjectures that the cover time bound generalizes to all $\mathcal{N}$, that is, for all $\mathcal{N} \leq n, E[T(\mathcal{N})] = O(\mathcal{N}^3)$. We prove Linial's conjecture.

THEOREM 1.1. *For any connected graph on n vertices and for any $\mathcal{N} \leq n$,*

$$E[T(\mathcal{N})] = O(\mathcal{N}^3).$$

* Received by the editors March 21, 1994; accepted for publication (in revised form) December 13, 1994. Portions of this work were performed while the authors were at the IBM T.J. Watson Research Center, Yorktown Heights, NY.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 6E3, Canada (gsbarnes@plg.uwaterloo.ca). The research of this author was supported by an NSERC International Fellowship. Portions of this research were performed while the author was at the Max-Planck-Institut für Informatik, Saarbrücken, Germany.

‡ Department of Applied Mathematics, The Weizmann Institute, Rehovot 76100, Israel (feige@wisdom.weizmann.ac.il). The research of this author was supported by a Koret Foundation fellowship.

Zuckerman [23] proves an upper bound of $O(mn)$ on the time to traverse all edges in a general graph. We are unaware of any previous nontrivial bounds for $\mathcal{M} < m$. We prove the following theorem.

THEOREM 1.2. *For any connected graph with m edges and for any $\mathcal{M} \leq m$,*

$$E[T(\mathcal{M})] = O(\mathcal{M}^2).$$

Theorem 1.2 holds even if G is not a simple graph (i.e., if we allow self-loops and parallel edges).

Let $E[T(\mathcal{M},\mathcal{N})]$ be the expected time for a simple random walk to either traverse $\mathcal{M}$ distinct edges or visit $\mathcal{N}$ distinct vertices (whichever comes first). Then the following theorem implies both of the above theorems, by considering $E[T(\mathcal{N}^2,\mathcal{N})]$ and $E[T(\mathcal{M},\mathcal{M})]$, respectively.

THEOREM 1.3. *For any connected graph with m edges and n vertices and for any $\mathcal{M}$ and $\mathcal{N}$ such that $\mathcal{M} \leq m$ or $\mathcal{N} \leq n$,*

$$E[T(\mathcal{M},\mathcal{N})] = O(\mathcal{M}\mathcal{N}).$$

In the above three theorems, the graph G need not be finite. If G is a graph with infinitely many vertices (each vertex of finite degree), then we can consider only the finite portion of G that is within a distance $\mathcal{N}$ (or $\mathcal{M}$) of the starting vertex of the random walk, and the proofs remain unchanged. For finite graphs, the following theorem serves to complete the picture of the rate at which vertices (or edges) are discovered. It provides better bounds than Theorems 1.1 and 1.2 when the number of vertices to be discovered is larger than $\sqrt{m}$ or the number of edges to be discovered is larger than n.

THEOREM 1.4. *For any simple connected graph on n vertices and m edges and for any $\mathcal{N} \leq n$,*

$$E[T(\mathcal{N})] = O(m\mathcal{N}),$$

and for any $\mathcal{M} \leq m$,

$$E[T(\mathcal{M})] = O(n\mathcal{M}).$$

Our theorems are the best possible in the sense that there exist graphs for which the bounds are tight up to constant factors (e.g., the n-cycle for Theorem 1.2). However, these bounds can be refined if additional information regarding the structure of G is given. The work of Kahn et al. [18] indicates that $d_{\min}$, the minimum degree of the vertices in the graph G, is a useful parameter to consider. They show that the expected cover time of any connected graph is $O(mn/d_{\min})$, implying a cover time of $O(n^2)$ for regular graphs. This inverse dependency on $d_{\min}$ applies also to short random walks. Preliminary results in this direction (tight up to a logarithmic factor) were presented in an earlier version of this paper [5]. The superfluous logarithmic factor in these results was subsequently removed by Feige [13], building upon proof techniques that were developed by Aldous [3]. Aldous is writing a textbook that will include a systematic account of random walks on graphs and reversible Markov chains. The current draft [3] contains results similar to ours in the regular graph setting.

While the short-term behavior of random walks is worth studying in its own right, short random walks also have immediate applications in many areas of computer science. Our results, of course, cannot be applied to all such areas. For example, much

stronger results are already known about the properties of short random walks on the special class of graphs known as *expanders* (see, e.g., Ajtai et al. [1] and Jerrum and Sinclair [17]). One might hope our results would dramatically improve the algorithms of Karger et al. [19] and Nisan et al. [20] for undirected connectivity. As mentioned above, both require an estimate of $E[T(\mathcal{N})]$ (and both used the estimate $E[T(\mathcal{N})] = O(\mathcal{N}^4)$). Unfortunately, substituting our bound only improves the constants for the algorithms since the running times of both depend on the *logarithm* of $E[T(\mathcal{N})]$ rather than $E[T(\mathcal{N})]$.

Our results may yield significant improvements for other randomized algorithms. In particular, consider randomized time–space tradeoffs for undirected $\mathcal{S}$–$\mathcal{T}$ connectivity (USTCON), as studied by Broder et al. [8]. One key property of Broder et al.'s algorithm is that a short random walk from a given edge traverses many edges. Improved bounds on $E[T(\mathcal{M})]$, then, would seem to provide an improvement to their tradeoff. Partial results in this direction were presented in an earlier version of our paper [5], and further improvements are presented by Feige [13].

2. Proofs of theorems. The proofs of the theorems are best read in order. The proof of Theorem 1.1 introduces the proof techniques that are used in all subsequent proofs. The proof of Theorem 1.2 is a simple modification of this proof technique. The proof of Theorem 1.3 introduces additional subtleties. For this reason, we have stated Theorems 1.1 and 1.2 explicitly rather than presenting them as corollaries of Theorem 1.3. The proof of Theorem 1.4 is a slight modification of the proofs of Theorems 1.1 and 1.3.

Proof of Theorem 1.1. Assume that $n > 2\mathcal{N}$. Otherwise the proof follows from Aleliunas et al.'s bound of $E[T(n)] = O(mn)$ [4]. We view the random walk as proceeding in phases. For any i, $1 \le i \le 2\mathcal{N}$, at the beginning of phase i, we identify (as described later) a set of vertices $V_i \subset V$ and a starting vertex $s_i \in V_i$, the last vertex visited in phase $i-1$. Phase i starts with the random walk at s_i and ends when the random walk exits V_i. We show that for any i, the expected number of steps taken in phase i is $O(i^2)$ and that up to phase i at least $i/2$ distinct vertices are visited. Thus, at most $2\mathcal{N}$ phases are needed to visit $\mathcal{N}$ distinct vertices, proving (by linearity of the expectation) that $E[T(\mathcal{N})] = O(\mathcal{N}^3)$.

To simplify the presentation, assume that G contains a Hamiltonian cycle. The case where G does not have a Hamiltonian cycle is only slightly more complicated and will be addressed later. Let $v_1, v_2, \ldots, v_n$ be an arrangement of the vertices of G along the Hamiltonian cycle.

At the beginning of phase i, we identify the following vertices and sets of vertices:

Starting vertex. The vertex s_i at which the walk of phase $i-1$ ended (if $i > 1$); s_1 is the starting vertex of the whole random walk.

Right vertex. The vertex r_i following s_i in the cyclic order imposed by the Hamiltonian cycle.

Visited vertices. The set $Y_i \subset V$ of vertices visited in previous steps by the walk; note that $s_i \in Y_i$.

Good vertices. The set of vertices $U_i \subseteq V \setminus Y_i$ (for two sets A and B, $A \setminus B$ denotes the set of elements in A but not in B) with the following property: let $R_{v_j,\ell}$, for $1 \le \ell \le n$, be the set of ℓ consecutive vertices $\{v_j, v_{j+1}, \ldots, v_{j+\ell-1}\}$ on the Hamiltonian cycle. (By convention, if $k > n$, then v_k is interpreted as v_{k-n}.) $v_j \in U_i$ if and only if for all $\ell \le n$, $|R_{v_j,\ell} \cap Y_i| \le \ell/2$. Thus, a vertex v_j is good in phase i if, starting at v_j and walking along the Hamiltonian cycle, at least half of the vertices discovered are new. This holds for any number of

steps that a walk might make before walking completely around the cycle.

Bad vertices. All other vertices. Let B_i denote this set.

The following lemma gives a bound on $|B_i|$.

LEMMA 2.1. *For every i, if $|Y_i| < n/2$, then $|B_i| < |Y_i|$.*

Proof. The proof is by induction on $|Y_i|$. If $|Y_i| = 1$ (and $n > 2$), then $|B_i| = 0$, as there is no consecutive set of vertices starting from an unvisited vertex such that a majority of vertices in the set are visited.

Assume the lemma is true for $|Y_i| = k$ and any $n > 2k$, and prove for $|Y_i| = k+1$ and any $n > 2k+2$. Consider a ring with n vertices and $k+1$ visited vertices ($n > 2k+2$). Then there exists a visited vertex y that is to the right of an unvisited vertex v. Remove y and v from the ring, linking the left neighbor of v to the right neighbor of y in the obvious way. Thus, n is decreased by 2 and $|Y_i|$ is decreased by 1. Now the induction hypothesis holds, and there are at most $k-1$ bad vertices. Put y and v back in the ring, and restore the values of $|Y_i|$ and n. No previously good vertex can become bad, as any consecutive set of vertices that starts at an unvisited vertex and includes y must contain v as well. Thus, the only bad vertex that could possibly have been added is v, resulting in the desired bound of at most k bad vertices. □

Let $V_i = (Y_i \cup B_i) \setminus \{r_i\}$. At the beginning of phase i the random walk is at the vertex $s_i \in V_i$. Phase i ends when the random walk exits V_i. Let T_i denote the number of steps taken in phase i.

LEMMA 2.2. $E[T_i] < (2|Y_i|)^2$.

Proof. By Lemma 2.1, and since $|V_i| \leq |Y_i| + |B_i|$, it follows that $|V_i| < 2|Y_i|$. Phase i ends when an edge leading out of V_i is taken. One such edge is the edge connecting s_i to r_i on the Hamiltonian cycle. We claim that if we remove all other edges leading out of V_i, the expected time to leave V_i remains at least $E[T_i]$.

Suppose we wish to remove a single edge e between $u \in V_i$ and $v \notin V_i$. Instead of actually removing e, create a new auxiliary vertex $w \notin V_i$ and make e connect u with w. This does not change $E[T_i]$. Let T' denote the number of steps taken to exit $V_i \cup \{w\}$, not counting steps that traverse the edge e (in either direction). Then $E[T'] \geq E[T_i]$. Finally, remove e completely, and observe that the expected time to leave V_i remains $E[T']$.

After removing all edges but $e = \{s_i, r_i\}$ leading out of V_i, we are left with only the subgraph induced by V_i and the edge e. Let m_i denote the number of edges in this subgraph. Observe that if $i > 1$, $m_i \leq |V_i|^2/2$. (If $i = 1, T_i = 1$ and the lemma trivially holds.) Since the walk of phase i starts at s_i and since there is an edge connecting s_i to r_i, the expected time to reach r_i is at most $2m_i$. (This is well known. See, e.g., Aleliunas et al. [4].) □

We have bounded the duration of each phase. It remains to bound the number of phases.

LEMMA 2.3. *For any k, in the first k phases at least $k/2$ distinct vertices are visited.*

Proof. Observe that each phase ends by either walking to the next vertex along the Hamiltonian cycle or "jumping" to a good vertex. Let S be the sequence of starting vertices visited by the walk, $s_1, s_2, \ldots, s_k$. Partition S into subsequences as follows: the first subsequence begins with s_1, and s_j begins a new subsequence if and only if phase $j-1$ did not end at r_{j-1}. Then each subsequence begins with a good vertex, and at least half the vertices visited in it are new. □

Note that given the definition of good vertices, it is possible that there may be many consecutive phases in the walk in which no new vertex is visited. However, this

can only occur if earlier in the same subsequence there were enough phases in which a new vertex was visited. Breaking the walk into subsequences, therefore, amounts to amortizing the cost of the phases in which no new vertex is found over the earlier phases in the same subsequence.

If G has a Hamiltonian cycle, this completes the proof, as

$$E[T(\mathcal{N})] \leq \sum_{i=1}^{2\mathcal{N}} E[T_i] < \sum_{i=1}^{2\mathcal{N}} (2|Y_i|)^2 < 8\mathcal{N}^3.$$

If G does not have a Hamiltonian cycle, we can use the following well-known lemma.

LEMMA 2.4. *For any connected graph, G, there is a cyclic ordering of its vertices, $w_1, w_2, \ldots, w_n$, such that the distance (the length of the shortest path in G) between any vertex and its successor is at most* 3.

Proof. Let G_{span} be a spanning tree of G. Traverse G_{span} in depth-first search fashion, using vertices of even distance from the root to advance toward the leaves and vertices of odd distance from the root to backtrack. Let $w_1, w_2, \ldots, w_n$ be the vertex ordering derived from this traversal, where w_i is the ith vertex visited by the traversal. Then w_n is a neighbor of w_1, the root of G_{span}, and for all $1 \leq i < n$, w_i is at most distance 3 from w_{i+1}. □

Using the ring obtained by Lemma 2.4 in place of a Hamiltonian cycle makes the expected time to leave V_i at most three times as large, thus only affecting the constants involved. □

Proof of Theorem 1.2. In Lemma 2.4 we show a way to arrange the vertices of a connected graph in a cyclic order. Arranging the edges in a cyclic order is even simpler. View each undirected edge as two *anti-parallel* directed edges (two directed edges are anti-parallel if they have the same endpoints, u and v, but one is directed from u to v, and the other directed from v to u). The number of directed edges entering any vertex is equal to the number of directed edges leaving it. Hence the directed graph is Eulerian and has an Eulerian cycle. This Eulerian cycle induces a cyclic ordering on the directed edges and can replace the Hamiltonian cycle used in the proof of Theorem 1.1.

Now the proof technique of Theorem 1.1 can be applied to prove Theorem 1.2, with "directed edges" replacing "vertices" in a straightforward manner. For edges, however, Lemma 2.2 can be strengthened—in phase i, the set V_i is now a set of edges and not a set of vertices, so the expected time to leave V_i is $|V_i|$ instead of $|V_i|^2$. This yields a bound of $E[T(\mathcal{M})] = O(\mathcal{M}^2)$. □

Proof of Theorem 1.3. Assume that $m \geq 2\mathcal{M}$. The proof for the case $m < 2\mathcal{M}$ is much simpler. See, for example, the first part of the proof of Theorem 1.4.

As in the proof of Theorem 1.1, view the random walk as proceeding in phases. For any $i \geq 1$, at the beginning of phase i, we identify (as described later) a set of vertices $V_i \subset V$, a set of edges $E_i \subset E$, and a starting vertex $s_i \in V_i$—the last vertex visited in phase $i-1$. Phase i starts with the random walk at s_i. Phase i ends when the random walk exits the subgraph $G_i = (V_i, E_i)$. Phases where s_i has many yet unvisited outgoing edges will also end if the walk returns to s_i. The whole walk ends when either $\mathcal{N}$ vertices or $\mathcal{M}$ edges are visited. This does not necessarily correspond to any fixed number of phases, making the analysis (of the expected number of steps taken by the walk) subtler than the analysis in the proof of Theorem 1.1, where the completion of $2\mathcal{N}$ phases guaranteed the end of the walk.

View each undirected edge as two anti-parallel directed edges. Observe that traversing $2\mathcal{M} - 1$ distinct directed edges guarantees at least $\mathcal{M}$ distinct original

(undirected) edges are traversed. The set of outgoing edges from vertex v is denoted by $\mathrm{Out}(v)$. Let $d(v) = |\mathrm{Out}(v)|$.

Let $v_1, v_2, \ldots, v_n$ be an arrangement of the vertices of G along the ring obtained by Lemma 2.4. At the beginning of phase i, we identify the following vertices and sets of vertices and edges.

Starting vertex. The vertex s_i at which the walk of phase $i-1$ ended (if $i > 1$); s_1 is the starting vertex of the whole random walk.

Right vertex. The vertex r_i following s_i in the ring of vertices.

Traversed edges. The set $F_i \subset E$ of edges traversed in previous steps by the walk.

Visited vertices. The set $Y_i \subseteq V$ of vertices visited in previous steps by the walk; note that $s_i \in Y_i$.

Exhausted vertices. The set $X_i \subseteq Y_i$ of vertices, x, such that at least half of the outedges from x have been traversed in previous steps by the walk.

Good vertices. The set of vertices U_i with the following property: As in the proof of Theorem 1.1, let $R_{v_j,\ell}$, for $1 \leq \ell \leq n$, be the set of ℓ consecutive vertices $\{v_j, v_{j+1}, \ldots, v_{j+\ell-1}\}$ on the ring of vertices. Define $g_i(v)$ to be $\max(0, \lceil d(v)/2 \rceil - |F_i \cap \mathrm{Out}(v)|)$, that is, the number of untraversed outedges of v that would have to be traversed for v to become exhausted. For a vertex v_j, $v_j \in U_i$ if and only if for all $\ell, 1 \leq \ell \leq n$,

$$\sum_{v_k \in R_{v_j,\ell}} g_i(v_k) \geq (2\mathcal{M}/\mathcal{N})|R_{v_j,\ell} \cap X_i|.$$

Note that a visited vertex can be good, but an exhausted vertex cannot. Note also that s_1 is a good vertex in phase 1.

Bad vertices. The set of vertices B_i that are not good.

Good edges. The set of untraversed edges $D_i \subseteq E \setminus F_i$ that exit from good vertices.

Bad edges. The set of edges C_i that are neither good nor traversed.

Informally, the definition of good vertices above is similar to the definition of good vertices in the proof of Theorem 1.1. In that proof, each subsequence (see the proof of Lemma 2.3) begins at a good vertex and progresses along the ring. The definition of a good vertex ensures that the number of previously unvisited vertices visited during the subsequence is at least a constant fraction of the number of phases in the subsequence. In this proof, a subsequence again starts at a good vertex, but the walk does not progress along the ring until the current vertex is exhausted. This definition of good vertices ensures that the number of previously untraversed edges that are traversed during a subsequence is at least $2\mathcal{M}/\mathcal{N}$ times the number of exhausted vertices that are starting vertices in the phases in the subsequence. This property is used in Lemma 2.9 to bound the number of phases that begin at exhausted vertices.

The following lemma gives a bound on $|C_i|$.

LEMMA 2.5. *For every i, $|C_i| \leq |F_i| + 4\mathcal{M}(1 + |X_i|/\mathcal{N})$. In particular, for $|F_i| < 2\mathcal{M}$, $|C_i| < 10\mathcal{M}$.*

Proof. Consider the ring of vertices at the beginning of phase i. For any vertex v, let $g_i'(v) = \max(0, d(v) - 2|F_i \cap \mathrm{Out}(v)|)$, and mark $g_i'(v)$ of v's untraversed outedges. Since $2|F_i| + \sum_v g_i'(v) \geq 2m$, the number of untraversed edges in G that are not marked is at most $|F_i|$. We will show by induction on $|X_i|$ that the number of marked edges that are bad is no more than $4\mathcal{M}(1 + |X_i|/\mathcal{N})$. Hence the total number of bad edges is as claimed in the lemma.

To bound the number of marked edges that are bad, we prove the following lemma. The lemma is more general than is necessary, since it considers not only configurations of marked edges and exhausted vertices that could be a result of random walks on graphs but also configurations that could not.

LEMMA 2.6. *Consider a ring of n vertices, and let $k < \mathcal{N} \leq n$. Choose k of the vertices in an arbitrary way and mark them as* exhausted. *Distribute an arbitrary number of tokens on the unexhausted vertices of the ring in an arbitrary way. A vertex v is* bad *if for some $\ell \leq n$, the number of tokens encountered by taking ℓ steps to the right (including the tokens on v itself) is less than $4\mathcal{M}/\mathcal{N}$ times the number of exhausted vertices encountered by such a walk. A token is* bad *if it is placed on a bad vertex. Then for any $n, \mathcal{N}, \mathcal{M}$, and k, the number of bad tokens is at most $4\mathcal{M}(1 + k/\mathcal{N})$.*

Proof. The lemma is proved by induction on k. If $k = 0$, then for all values of n, $\mathcal{N}$, and $\mathcal{M}$, there are no bad tokens, and the lemma holds.

Assume the lemma is true for $k = j$, for all values of n, $\mathcal{N}$, and $\mathcal{M}$, where $k < \mathcal{N} \leq n$. Prove for $k = j + 1$ and arbitrary n, $\mathcal{N}$, and $\mathcal{M}$, where $k < \mathcal{N} \leq n$.

Consider a walk backward along the ring from some exhausted vertex, v. After a certain number of steps along the ring, the walk will have encountered $4\mathcal{M}/\mathcal{N}$ tokens. Let y_v be the vertex where this walk from v first reaches its $(4\mathcal{M}/\mathcal{N})$th token. Because there are at least $4\mathcal{M}$ tokens (otherwise, the lemma trivially holds) and at most $\mathcal{N}$ exhausted vertices, there must be some exhausted vertex v such that the walk from v to y_v visits no exhausted vertex besides v.

Let v be such a vertex and $\mathcal{T}_v$ be the first $4\mathcal{M}/\mathcal{N}$ tokens encountered by this walk (this may include only some of the tokens placed on y_v). Remove from the ring the tokens $\mathcal{T}_v$, and make v not exhausted. Thus, k is decreased by 1. Now the induction hypothesis holds, so there are at most $4\mathcal{M}(1 + (k-1)/\mathcal{N})$ bad tokens.

Add the tokens in $\mathcal{T}_v$ back to the ring, mark v as exhausted, and restore the value of k. The tokens in $\mathcal{T}_v$ may be bad, but no token t that is not in $\mathcal{T}_v$ and was not previously bad can become bad, as any walk from t that includes v must include all the tokens between y_v and v as well. So the number of bad tokens increases by at most $4\mathcal{M}/\mathcal{N}$, proving the lemma. □

Observe that for any subset V_s of the vertices, $\sum_{v \in V_s} g_i(v) \geq (\sum_{v \in V_s} g'_i(v))/2$. By considering marked edges as the tokens of Lemma 2.6 and bad marked edges as the bad tokens, the proof of Lemma 2.5 follows. □

The definition of the subgraph $G_i = (V_i, E_i)$ and the stopping condition for phase i depends on whether s_i is exhausted or not. At the beginning of phase i, the random walk is at the vertex $s_i \in V_i$.

If s_i is not exhausted, $V_i = B_i \cup \{s_i\}$, E_i are all the edges with both endpoints in V_i, along with the edges out of s_i and the edges into s_i; and phase i ends when the random walk returns to s_i or exits G_i by visiting a vertex in U_i.

If s_i is exhausted, $V_i = B_i \setminus \{r_i\}$; E_i are all the edges with both endpoints in V_i, along with the edges along a shortest path from s_i to r_i and the edges anti-parallel to the edges in this path; and phase i ends when the random walk exits G_i by visiting a vertex in $U_i \cup \{r_i\}$.

Let T_i denote the number of steps taken in phase i.

LEMMA 2.7. *If phase i begins at an unexhausted vertex, v, $E[T_i] < 12\mathcal{M}/d(v) + 2$.*

Proof. The number of edges in G_i is no more than the number of outedges from vertices in V_i plus the edges into s_i. An outedge from a bad vertex is either bad or traversed, so $|E_i| \leq |F_i| + |C_i| + 2d(v)$. Therefore, by Lemma 2.5, if $|F_i| < 2\mathcal{M}$, then

$|E_i| < 12\mathcal{M} + 2d(v)$.

It is well known that on an undirected graph with m_i edges the expected time for a random walk that starts at vertex v to return to v is $2m_i/d(v)$. G_i is equivalent to an undirected graph with $|E_i|/2$ edges, since if a directed edge e is in E_i, the edge anti-parallel to e is in E_i as well. The degree of v in G_i is the same as its degree in G; therefore, the expected length of a phase that begins at an unexhausted vertex, v, is no more than $12\mathcal{M}/d(v) + 2$. □

LEMMA 2.8. *If phase i begins at an exhausted vertex, $E[T_i] < 36\mathcal{M} + 15$.*

Proof. The only edges in E_i that may not be outedges from bad or exhausted vertices are the edges in the path from s_i to r_i and the edges anti-parallel to these edges. By the construction of the ring of vertices, this path is of length 3 or less, and the first outedge in the path from s_i is from an exhausted vertex, so there are at most five such edges, and $|E_i| \leq |F_i| + |C_i| + 5 < 12\mathcal{M} + 5$. Using a proof similar to the proof of Lemma 2.2, the expected length of such a phase is less than the distance from s_i to r_i times $|E_i|$. □

We have bounded the duration of each phase. It remains to bound the number of phases. Call phases that start at unexhausted vertices *short phases* and phases that start at exhausted vertices *long phases.* The walk ends when $|Y_i| \geq \mathcal{N}$ or $|F_i| \geq 2\mathcal{M}-1$. Since we are considering directed edges, this will ensure that either $\mathcal{N}$ vertices were visited or $\mathcal{M}$ undirected edges were traversed. To analyze the expected number of phases of the walk, consider the following two stopping conditions.

1. At least $\mathcal{N}$ distinct vertices were the starting vertices of phases. Clearly, this implies that $\mathcal{N}$ vertices were visited.

2. There were at least $2\mathcal{N}$ long phases. This implies that at least $2\mathcal{M}$ edges have been visited, by the following lemma.

LEMMA 2.9. *If $m \geq 2\mathcal{M}$, $2\mathcal{N}$ long phases occur, and no more than $\mathcal{N}$ distinct vertices are the starting vertices of these long phases, then at least $2\mathcal{M}$ edges are traversed.*

Proof. Similar to the proof of Lemma 2.3, observe that each phase ends by either walking to the next vertex in the ring, returning to the starting vertex, or "jumping" to a good vertex. Let S be the sequence of starting vertices visited by the walk, $s_1, s_2, \ldots, s_k$. Partition S into subsequences as follows. The first subsequence begins with s_1, and s_j begins a new subsequence if and only if phase $j-1$ was long and did not end at r_{j-1} or phase $j-1$ was short and did not end at s_{j-1}. Then each subsequence begins with a good vertex (and a short phase) and continually exhausts its current vertex and steps to the next vertex in the ring. Because $m \geq 2\mathcal{M}$, a subsequence that begins at s_j cannot step completely around the ring and return to s_j. We use the following property: For a subsequence S_j beginning with phase k, at least $2q(\mathcal{M}/\mathcal{N})$ edges are traversed for the first time in the phases of S_j before the qth vertex in X_k appears as a starting vertex in S_j. This must be true by the definition of a good vertex. With each vertex v of X_k we can therefore associate $2\mathcal{M}/\mathcal{N}$ untraversed edges that must be traversed if v is to start a phase within the subsequence, associating each untraversed edge with at most one vertex of X_k.

Now consider the $2\mathcal{N}$ long phases. They start at no more than $\mathcal{N}$ distinct vertices, so at least $\mathcal{N}$ of them start at a vertex that was the starting vertex of a previous long phase. Let phase i be such a phase, and assume that it is part of a subsequence that begins with phase k. Then s_i must already have been exhausted when phase k was begun, implying that we can identify $2\mathcal{M}/\mathcal{N}$ edges that were first traversed between phases k and i and associate them with phase i. Altogether, from all long phases, we

can identify at least $\mathcal{N} \cdot 2\mathcal{M}/\mathcal{N} = 2\mathcal{M}$ distinct traversed edges. □

By Lemma 2.9, there are at most $2\mathcal{N}$ long phases and each long phase takes expected time no more than $36\mathcal{M} + 15$, so the long phases contribute a total of $O(\mathcal{N}\mathcal{M})$ to the expected number of steps in the walk. To analyze the contribution of the short phases, let v_i denote the ith distinct vertex that was discovered by the walk and $E[v_i]$ denote the expected number of short phases that start at v_i. For each such phase, the probability that the first step of the walk traverses a yet untraversed outedge from v_i is at least $1/2$ since the majority of edges leading out of v_i are untraversed. Therefore $E[v_i]$ is no more than $d(v_i)$. If $d(v_i) > 2\mathcal{M}$, similar reasoning shows that $E[v_i]$ is no more than $2\mathcal{M}$ since the walk can stop after $\mathcal{M}$ distinct outedges of v_i are traversed. The expected number of steps in a short phase is no more than $12\mathcal{M}/d(v_i) + 2$, so it follows by Wald's Equation (see Ross [21, page 38], for example) that the expected number of steps spent on short phases that begin at v_i is no more than $(12\mathcal{M}/d(v_i) + 2) \cdot \min(d(v_i), 2\mathcal{M}) \leq 16\mathcal{M}$. The short phases therefore contribute a total of $O(\mathcal{N}\mathcal{M})$ to the expected number of steps in the walk. □

Proof of Theorem 1.4. To show that $E[T(\mathcal{N})] = O(m\mathcal{N})$, we distinguish between two cases. The case $\mathcal{N} \geq n/2$ is handled by Aleliunas et al. [4], who show that $E[T(n)] = O(mn)$. For the case $\mathcal{N} < n/2$, consider the proof of Theorem 1.1. The maximum distance between consecutive vertices on the ring is 3, so the expected time of each phase is at most $6m$, and the proof follows.

To show that $E[T(\mathcal{M})] = O(n\mathcal{M})$, we again distinguish between two cases. The case $\mathcal{M} > m/2$ is handled by Zuckerman [23], who shows that $E[T(m)] = O(mn)$. For the case $\mathcal{M} \leq m/2$, consider the proof of Theorem 1.3 with $\mathcal{N} = n$. Observe that by Lemma 2.9, in order to visit $\mathcal{M}$ distinct edges, it suffices to have $2n$ phases that start at exhausted vertices. Hence the expected number of steps spent on phases that start at exhausted vertices is $O(\mathcal{M}n)$. Likewise, since the graph has only n vertices, the expected number of steps spent on phases that start at vertices that are not exhausted is also $O(\mathcal{M}n)$ (the fact that the random walk may not stop at the time that all vertices of G are visited does not affect this argument). The proof follows. □

Acknowledgments. We thank Nati Linial, Prabhakar Raghavan, and Larry Ruzzo for many helpful discussions and for directing us to relevant literature.

REFERENCES

[1] M. Ajtai, J. Komlós, and E. Szemerédi, *Deterministic simulation in LOGSPACE*, in Proc. 19th Annual ACM Symposium on Theory of Computing, New York, May 1987, pp. 132–140.

[2] D. J. Aldous, *Lower bounds for covering times for reversible Markov chains and random walks on graphs*, J. Theoret. Probab., 2 (1989), pp. 91–100.

[3] ———, *Reversible Markov chains and random walks on graphs*, draft, University of California, Berkeley, 1993.

[4] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. W. Rackoff, *Random walks, universal traversal sequences, and the complexity of maze problems*, in Proc. 20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, October 1979, IEEE, pp. 218–223.

[5] G. Barnes and U. Feige, *Short random walks on graphs*, in Proc. 25th Annual ACM Symposium on Theory of Computing, San Diego, CA, May 1993, pp. 728–737.

[6] A. Z. Broder, *How hard is it to marry at random? (On the approximation of the permanent)*, in Proc. 18th Annual ACM Symposium on Theory of Computing, Berkeley, CA, May 1986,

pp. 50–58. Errata: Proc. 20th Annual ACM Symposium on Theory of Computing, Chicago, IL, May 1988, p. 551.

[7] A. Z. Broder and A. R. Karlin, *Bounds on the cover time*, J. Theoret. Probab., 2 (1989), pp. 101–120.

[8] A. Z. Broder, A. R. Karlin, P. Raghavan, and E. Upfal, *Trading space for time in undirected s-t connectivity*, in Proc. 21st Annual ACM Symposium on Theory of Computing, Seattle, WA, May 1989, pp. 543–549.

[9] A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari, *The electrical resistance of a graph captures its commute and cover times*, in Proc. 21st Annual ACM Symposium on Theory of Computing, Seattle, WA, May 1989, pp. 574–586.

[10] A. Cohen and A. Wigderson, *Dispersers, deterministic amplification, and weak random sources*, in Proc. 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, NC, October 1989, IEEE, pp. 14–19.

[11] D. Coppersmith, P. Tetali, and P. Winkler, *Collisions among random walks on a graph*, SIAM J. Discrete Math., 6 (1994), pp. 363–374.

[12] M. Dyer, A. M. Frieze, and R. Kannan, *A random polynomial time algorithm for approximating the volume of convex bodies*, in Proc. 21st Annual ACM Symposium on Theory of Computing, Seattle, WA, May 1989, pp. 375–381.

[13] U. Feige, *A randomized time-space tradeoff of $\tilde{O}(m\hat{R})$ for USTCON*, in Proc. 34th Annual Symposium on Foundations of Computer Science, Palo Alto, CA, November 1993, IEEE, pp. 238–246.

[14] O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesan, and D. I. Zuckerman, *Security preserving amplification of hardness*, in Proc. 31st Annual Symposium on Foundations of Computer Science, St. Louis, MO, October 1990, IEEE, pp. 318–326.

[15] R. Impagliazzo and D. I. Zuckerman, *How to recycle random bits*, in Proc. 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, NC, October 1989, IEEE, pp. 248–253.

[16] A. Israeli and M. Jalfon, *Token management schemes and random walks yield self-stabilizing mutual exclusion*, in Proc. 9th Annual ACM Symposium on Principles of Distributed Computing, Quebec City, Quebec, Canada, August 1990, pp. 119–131.

[17] M. Jerrum and A. Sinclair, *Conductance and the rapid mixing property for Markov chains: The approximation of the permanent resolved*, in Proc. 20th Annual ACM Symposium on Theory of Computing, Chicago, IL, May 1988, pp. 235–244.

[18] J. D. Kahn, N. Linial, N. Nisan, and M. E. Saks, *On the cover time of random walks on graphs*, J. Theoret. Probab., 2 (1989), pp. 121–128.

[19] D. R. Karger, N. Nisan, and M. Parnas, *Fast connected components algorithms for the EREW PRAM*, in Proc. 1992 ACM Symposium on Parallel Algorithms and Architectures, San Diego, CA, June–July 1992, pp. 373–381.

[20] N. Nisan, E. Szemerédi, and A. Wigderson, *Undirected connectivity in $O(\log^{1.5} n)$ space*, in Proc. 33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, PA, October 1992, IEEE, pp. 24–29.

[21] S. M. Ross, *Applied Probability Models with Optimization Applications*, Holden-Day, San Francisco, CA, 1970.

[22] D. I. Zuckerman, *A technique for lower bounding the cover time*, in Proc. 22nd Annual ACM Symposium on Theory of Computing, Baltimore, MD, May 1990, pp. 254–259.

[23] ———, *On the time to traverse all edges of a graph*, Inform. Process. Lett., 38 (1991), pp. 335–337.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 29–36, February 1996

004

THE BIASED COIN PROBLEM*

RAVI B. BOPPANA† AND BABU O. NARAYANAN‡

Abstract. A *slightly random source* (with *bias* ϵ) is a sequence $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ of random bits such that the conditional probability that $\mathbf{x}_i = 1$, given the outcomes of the first $i-1$ bits, is always between $\frac{1}{2} - \epsilon$ and $\frac{1}{2} + \epsilon$. Given a subset S of $\{0,1\}^n$, define its *ϵ-biased probability* to be the minimum of $\Pr[\mathbf{X} \in S]$ over all slightly random sources $\mathbf{X}$ with bias ϵ. It is shown that, for every fixed $\epsilon < \frac{1}{2}$ and almost every subset S of $\{0,1\}^n$, the ϵ-biased probability of S is bounded away from 0.

Key words. pseudo-randomness, slightly random sources, randomized algorithms

AMS subject classifications. 68Q25, 60D05

1. Introduction.

1.1. Statement of problem. Randomized algorithms and cryptographic protocols usually assume access to a source of perfectly random bits. Unfortunately, physical sources of randomness (see Murry [10]) are not perfectly random. Can we nevertheless use such biased sources for random purposes? The purpose of this paper is to analyze the expected performance of one such source, that is, the slightly random source due to Santha and Vazirani [11].

A *slightly random source* (with *bias* $0 \le \epsilon \le \frac{1}{2}$) is a sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ of random bits such that the conditional probability that $\mathbf{x}_i = 1$, given the outcomes of the first $i-1$ bits, is always between $\frac{1}{2} - \epsilon$ and $\frac{1}{2} + \epsilon$. Intuitively, we are flipping a bunch of coins, but our adversary, who knows the complete history of previous coin flips, gets to choose the bias of each coin.

In the application to randomized algorithms, there is a "witness set" S that we are trying to hit, where S is some subset of $\{0,1\}^n$ (the set of all binary sequences of length n). Define the *ϵ-biased probability* $\Pr_\epsilon(S)$ of S by

$$\Pr_\epsilon(S) = \min_{\mathbf{x}} \Pr[\mathbf{x} \in S],$$

where $\mathbf{x}$ ranges over all slightly random sources with bias ϵ. For example, $\Pr_0(S) = |S|/2^n$, whereas $\Pr_{1/2}(S) = 0$ (unless $S = \{0,1\}^n$). Intuitively, the ϵ-biased probability measures the minimum odds of hitting S when our adversary, who knows S, gets to choose the source.

Is the ϵ-biased probability of a witness set S always within a constant factor of $|S|/2^n$? Unfortunately, the answer is no, as observed by Alon and Rabin [2]. As a

* Received by the editors June 13, 1994; accepted for publication (in revised form) January 10, 1995. A preliminary version of this paper appeared in *Proc. 25th Annual ACM Symposium on the Theory of Computing*, 1993, pp. 252–257 [6].

† Computer Science Department, New York University, New York, NY 10012 (boppana@cs.nyu.edu). The research of this author was supported in part by National Science Foundation Presidential Young Investigator Award CCR–9196230.

‡ Department of Mathematics, University of Illinois, Urbana, IL 61801 (bnarayan@math.uiuc.edu). Part of this research was performed while this author was at the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Rutgers University, Piscataway, NJ 08855 and at the Institute for Mathematics and its Applications, University of Minnesota, Minneapolis, MN 55455.

counterexample, consider the majority set (for an odd integer n)

$$\mathrm{MAJ} = \left\{ x \in \{0,1\}^n : \sum_{i=1}^{n} x_i > \frac{n}{2} \right\},$$

whose unbiased probability is $\frac{1}{2}$, but whose ϵ-biased probability is exponentially small for every fixed $\epsilon > 0$. Consequently it is impossible to obtain a strong bound for *every* witness set.

Instead, lowering our sights a little, let us aim for a bound that applies to *almost every* witness set. (Here S ranges uniformly over all subsets of $\{0,1\}^n$, and "almost every" means a $1 - o(1)$ fraction as n tends to infinity.) Alon and Rabin [2] showed that for $\epsilon < \frac{1}{2}(\sqrt{2}-1) \approx 0.207$, the ϵ-biased probability of almost every witness set is bounded away from 0. They posed the question of whether the same conclusion holds for *every* $\epsilon < \frac{1}{2}$.

1.2. Statement of results. Our main result is an affirmative answer to the Alon–Rabin question. More precisely, we show that for every $\epsilon < \frac{1}{2}$ there is a constant $c_\epsilon > 0$ such that almost every witness set has ϵ-biased probability at least c_ϵ.

The constant c_ϵ necessarily tends to 0 as ϵ increases to $\frac{1}{2}$. Our proof shows that c_ϵ is at least $p^{O(\log(1/p))}$, where $p = \frac{1}{2} - \epsilon$. We do not know if this lower bound can be improved.

Our work actually applies to a more general situation than the one described above. First, the witness set need not have a uniform distribution; all that matters is that the events "$x \in S$" (for x in $\{0,1\}^n$) be mutually independent. Second, the source need not output merely bits; any finite alphabet will do (example: dice). For simplicity, we omit these generalizations.

Our techniques also apply to the problems of collective coin-flipping and leader election in the perfect-information model [4], [5]. In parallel and distributed computing, a set of processors often has to produce the same random bit in order to perform some task. We would like to design a coin-flipping protocol that works even against many faulty processors. A protocol is said to be *t-resilient* if the probability of producing "heads" is bounded away from 0 and 1, given that the number of faulty processors is at most t. Under the perfect-information model, Alon and Naor [1] showed the existence of protocols that are ϵn-resilient for every $\epsilon < \frac{1}{3}$. Combining techniques from their paper and the present paper, we show in [7] that the Alon–Naor protocols are actually ϵn-resilient for every $\epsilon < \frac{1}{2}$.

1.3. Significance of work. The slightly random source is quite general and models many real-world sources. It allows many kinds of correlation among the random bits. It permits the adversary to know the witness set, to know the complete history of previous coins, and to be computationally powerful. Our result holds for any ϵ less than $\frac{1}{2}$, which means that the coins may be arbitrarily biased.

Our proof technique is interesting as well. Alon and Rabin [2] used a second-moment method to analyze the underlying random process. Unfortunately, this method provably fails for ϵ larger than $\frac{1}{2}(\sqrt{2}-1) \approx 0.207$. Instead, we analyze a higher moment of the random process. Because higher moments have fewer properties than does the second moment, many technical complications arise. Nevertheless, through the judicious use of classical inequalities, we are able to make this higher moment method succeed.

1.4. Relation to previous work. Santha and Vazirani [11] invented the model of slightly random sources. They showed how to approximate perfect randomness by using more than $\Omega(\log n)$ independent slightly random sources. Later, Vazirani [13] showed how to approximate true randomness using just two independent slightly random sources. In both cases, more than one source is required, and the sources are assumed to be statistically independent. In our problem, only one source is permitted.

Vazirani and Vazirani [14] showed how to simulate the complexity class BPP using just one slightly random source. Chor and Goldreich [8] showed how to simulate BPP using a weaker source, called a "probability-bounded" source. Zuckerman [15] showed how to simulate BPP using an even weaker source, called a "δ-source." In all three cases, they transform the bits of the imperfect source to create a polynomial number of n-bit strings, most of which will be witnesses with high probability. In our problem, no transformation is permitted; we must use the random bits directly. This restriction has the advantage that the resulting algorithm, when it works, will be more efficient.

Shamir [12] found an alternative proof of the Alon–Rabin result based on an interesting martingale technique. Unfortunately, and coincidentally, his proof also works only when $\epsilon < \frac{1}{2}(\sqrt{2}-1) \approx 0.207$.

Ben-Or, Linial, and Saks [5] have written a delightful survey on imperfect random sources and the collective coin-flipping problem.

2. Proof of main result. In this section, we prove that the ϵ-biased probability of almost every witness set is bounded away from 0, for every $0 \le \epsilon < \frac{1}{2}$. From now on, fix such an ϵ.

We begin with an interesting function that is intimately connected to slightly random sources. Given a vector $x = (x_1, x_2)$ in $\Re^2$, define its *biased mean* by

$$B(x) = \min(p_1 x_1 + p_2 x_2, p_2 x_1 + p_1 x_2), \tag{1}$$

where $p_1 = \frac{1}{2} - \epsilon$ and $p_2 = \frac{1}{2} + \epsilon$. There are three other, equivalent, ways to define biased mean:

$$B(x) = p_1 \max(x_1, x_2) + p_2 \min(x_1, x_2), \tag{2}$$

$$B(x) = \tfrac{1}{2}(x_1 + x_2) - \epsilon |x_1 - x_2|, \tag{3}$$

and

$$B(x) = \min_{p_1 \le p \le p_2} [p x_1 + (1-p) x_2]. \tag{4}$$

Notice that biased mean is an increasing function of its two arguments, which is easy to see from (1), (2), or (4).

As an application of biased mean, we show how to calculate recursively the ϵ-biased probability of a witness set $S \subseteq \{0,1\}^n$ (for $n \ge 1$). Let S_0 and S_1 be the $(n-1)$-dimensional *slices* of S defined as

$$S_0 = \{x \in \{0,1\}^{n-1} : 0x \in S\},$$
$$S_1 = \{x \in \{0,1\}^{n-1} : 1x \in S\}.$$

Then it is not hard to see that

$$\Pr_\epsilon(S) = B(\Pr_\epsilon(S_0), \Pr_\epsilon(S_1)). \tag{5}$$

The proof uses (4) together with the definitions of biased probability and slightly random source.

We next define a related transformation of probability distributions. Let D be a probability distribution on the real numbers. Define $\beta(D)$, the *biased-mean transformation* of D, to be the probability distribution of the random variable $B(x_1, x_2)$, where x_1 and x_2 are independent random variables with distribution D.

We are particularly interested in iterating this biased-mean transformation. Let D_0 be the uniform distribution on the two-element space $\{0, 1\}$. For $i \geq 1$, define $D_i = \beta(D_{i-1})$. Then it is not hard to see that the random variable $\Pr_\epsilon(S)$, where S is a random subset of $\{0,1\}^n$, has distribution D_n. The proof is by induction on n, using the recursive formula (5). This result provides a useful bridge between biased probability and the distribution D_n. Using this bridge, all we need to show is that the distribution D_n is concentrated away from 0, as n tends to infinity.

We next define a generalization of the variance of a probability distribution. Given a real probability distribution D and a nonnegative real number d, define the dth *moment* of D by

$$\mathrm{M}_d(D) = \mathrm{E}(|x - y|^d),$$

where x and y are independent random variables with distribution D. For example, it is easy to see that $\mathrm{M}_2(D) = 2\ \mathrm{Var}(D)$, where Var represents variance. There are some known relations between the various moments. For instance, the power-mean inequality [9] implies the inequality

$$\mathrm{M}_c(D)^{1/c} \leq \mathrm{M}_d(D)^{1/d}, \tag{6}$$

whenever $0 < c \leq d$.

We next describe our approach to analyzing the distribution D_n. Taking expected values of (3) shows that, for every real probability distribution D,

$$\mathrm{E}(\beta(D)) = \mathrm{E}(D) - \epsilon \mathrm{M}_1(D), \tag{7}$$

and hence (for $i \geq 1$)

$$\mathrm{E}(D_i) = \mathrm{E}(D_{i-1}) - \epsilon \mathrm{M}_1(D_{i-1}). \tag{8}$$

Therefore, to show that the expectation of D_n is large, we will show that the moments $\mathrm{M}_1(D_i)$ are small. Alon and Rabin [2] used the second-moment method. They showed that $\mathrm{M}_2(D_i)$ geometrically contracts as i increases, for small ϵ. This implies an exponentially small bound for $\mathrm{M}_1(D_i)$, thereby providing the result. Unfortunately, the contraction holds only when $\epsilon < \frac{1}{2}(\sqrt{2} - 1) \approx 0.207$.

Instead, we shall show that for every $\epsilon < \frac{1}{2}$ and every sufficiently large number d, the dth moment $\mathrm{M}_d(D_i)$ geometrically contracts as i increases.

Given a vector x in $\Re^2$, define its *biased norm* by

$$||x|| = \max(|p_1 x_1 + p_2 x_2|, |p_2 x_1 + p_1 x_2|).$$

Our first lemma says that biased mean is a Lipschitz continuous function.

LEMMA 2.1. *If x and y are two vectors in $\Re^2$, then*

$$|B(x) - B(y)| \leq ||x - y||.$$

Proof. Let $\delta = ||x - y||$. The definition of δ implies the two inequalities

$$p_1x_1 + p_2x_2 \leq p_1y_1 + p_2y_2 + \delta,$$
$$p_2x_1 + p_1x_2 \leq p_2y_1 + p_1y_2 + \delta.$$

By the definition of B, the minimum of the two left-hand expressions is $B(x)$; the minimum of the two right-hand expressions is $B(y) + \delta$. Hence $B(x) \leq B(y) + \delta$. By symmetry, it follows that $B(y) \leq B(x) + \delta$. Combining these last two inequalities leads to the desired result. □

The next lemma bounds the biased norm by more classical norms.

LEMMA 2.2. *If x is a vector in $\Re^2$ and $d > 1$ is a real number, then*

$$||(x_1, x_2)||^d + ||(-x_1, x_2)||^d \leq c\,(|x_1|^d + |x_2|^d),$$

where $c = \left(p_1^{d/d-1} + p_2^{d/d-1}\right)^{d-1} + p_2^d$.

Proof. Assume that x_1 and x_2 are nonnegative. (The other three cases are similar.) The first term on the left is dealt with using Hölder's inequality [9]. That inequality tells us that

$$|p_1x_1 + p_2x_2|^d \leq \left(p_1^{d/d-1} + p_2^{d/d-1}\right)^{d-1}(|x_1|^d + |x_2|^d).$$

The same bound holds for $|p_2x_1 + p_1x_2|^d$ and, hence, for $||(x_1, x_2)||^d$.

The second term on the left is even easier to deal with. We have

$$\begin{aligned} |-p_1x_1 + p_2x_2|^d &\leq \max(p_1|x_1|, p_2|x_2|)^d \\ &\leq p_1^d|x_1|^d + p_2^d|x_2|^d \\ &\leq p_2^d(|x_1|^d + |x_2|^d). \end{aligned}$$

The same bound holds for $|-p_2x_1 + p_1x_2|^d$ and, hence, for $||(-x_1, x_2)||^d$. Adding the bounds for the two terms on the left completes the proof. □

We obtain the following corollary by bounding the change in biased mean.

COROLLARY 2.3. *If x and y are two vectors in $\Re^2$ and $d > 1$ is a real number, then*

$$|B(x_1, x_2) - B(y_1, y_2)|^d + |B(y_1, x_2) - B(x_1, y_2)|^d \leq c\left(|x_1 - y_1|^d + |x_2 - y_2|^d\right),$$

where $c = \left(p_1^{d/d-1} + p_2^{d/d-1}\right)^{d-1} + p_2^d$.

Proof. Apply Lemma 2.1 to both terms on the left, and then apply Lemma 2.2. □

Next we introduce randomness into the picture.

COROLLARY 2.4. *Let x and y be two random vectors in $\Re^2$ such that x_1, x_2, y_1, and y_2 are independent and such that x_1 and y_1 are identically distributed. Then for every real number $d > 1$,*

$$\mathrm{E}(|B(x) - B(y)|^d) \leq \frac{c}{2}\mathrm{E}(|x_1 - y_1|^d + |x_2 - y_2|^d),$$

where $c = \left(p_1^{d/d-1} + p_2^{d/d-1}\right)^{d-1} + p_2^d$.

Proof. Take expected values of both sides of the inequality in Corollary 2.3. The two terms on the left-hand side have the same expected value, because the tuple

(x_1, x_2, y_1, y_2) has the same distribution as the tuple (y_1, x_2, x_1, y_2). Dividing by 2 gives the result. □

We finally obtain the contraction theorem that we have been seeking, namely, that β is a contracting transformation.

THEOREM 2.5. *If D is a real probability distribution and $d > 1$ is a real number, then*

$$\mathrm{M}_d(\beta(D)) \leq c\mathrm{M}_d(D),$$

where $c = (p_1^{d/d-1} + p_2^{d/d-1})^{d-1} + p_2^d$.

Proof. Let x_1, x_2, y_1, and y_2 be independent random variables with distribution D. Then Corollary 2.4 applies. The left side of the inequality in Corollary 2.4 is simply $\mathrm{M}_d(\beta(D))$ because $B(x)$ and $B(y)$ are independent random variables having distribution $\beta(D)$. The right side has two weighted copies of $\mathrm{M}_d(D)$, for a total of $c\mathrm{M}_d(D)$. The two sides complete the proof. □

Theorem 2.5 is a contraction result: The constant c can be made less than 1 by choosing d sufficiently large. This is because as d tends to infinity the value of c tends to $p_1^{p_1} p_2^{p_2} < 1$. In fact, the choice $d = \frac{1}{p_1} \log \frac{1}{p_1}$ makes $c < 1$.

Next, we will attempt to show that the expectation $\mathrm{E}(D_n)$ at the root is bounded away from 0.

THEOREM 2.6. *For every $0 \leq \epsilon < 1/2$, the final expectation $\mathrm{E}(D_n)$ is at least $\mathrm{E}(D_0) - \frac{\epsilon(\mathrm{M}_d(D_0))^{1/d}}{1-c^{1/d}}$, where $c = (p_1^{d/d-1} + p_2^{d/d-1})^{d-1} + p_2^d$.*

Proof. Applying Theorem 2.5 iteratively shows that $\mathrm{M}_d(D_i) \leq c^i \mathrm{M}_d(D_0)$. By applying Jensen's inequality [9] and Theorem 2.5 and summing up the resulting geometric series, we get

$$\begin{aligned}
\mathrm{E}(D_n) &= \mathrm{E}(D_0) - \epsilon \sum_{i=0} n - 1\mathrm{M}_1(D_i) \\
&\geq \mathrm{E}(D_0) - \epsilon \sum_{i=0} n - 1(\mathrm{M}_d(D_i))^{1/d} \\
&\geq \mathrm{E}(D_0) - \epsilon \sum_{i=0} n - 1(c^i \mathrm{M}_d(D_0))^{1/d} \\
&\geq \mathrm{E}(D_0) - \frac{\epsilon(\mathrm{M}_d(D_0))^{1/d}}{1 - c^{1/d}}. \qquad \square
\end{aligned}$$

We still have not shown that $\mathrm{E}(D_n)$ is bounded away from 0 for every $\epsilon < \frac{1}{2}$. That is because as ϵ approaches $\frac{1}{2}$, the value of d approaches infinity, causing $1 - c^{1/d}$ to approach 0, which renders the bound of Theorem 2.6 useless. Instead, we will start with another initial distribution for which it is clear that the final expectation is bounded away from 0, and then we will use a majorizing argument to show the same for our original initial distribution. The same idea was used by Alon and Rabin [2].

A random variable X is said to *stochastically dominate* a random variable Y if $\Pr[X \geq t]$ is at least $\Pr[Y \geq t]$ for every t. Denote this by $X \prec Y$. We need the following well-known property of stochastic dominance.

LEMMA 2.7 (Stoyan Theorem 2.2.4). *Let the random variables X_1, X_2, Y_1, Y_2 be such that $X_1 \prec Y_1$ and $X_2 \prec Y_2$. Then, for all increasing functions $\Phi : \Re^2 \to \Re$, $\Phi(X_1, X_2) \prec \Phi(Y_1, Y_2)$.* In particular, observe that $\mathrm{E}(X_1) \leq E(Y_1)$.

THEOREM 2.8. *For every $0 \leq \epsilon < \frac{1}{2}$, the final expectation $\mathrm{E}(D_n)$ is bounded away from 0.*

Proof. Given a nonnegative integer k to be chosen later, consider the new initial distribution F_0:

$$y = \begin{cases} 0 & \text{with probability } 2^{-2^k}, \\ p_1^k & \text{otherwise.} \end{cases}$$

For $i \geq 1$, define $F_i = \beta F_{i-1}$, where β is the biased-mean transformation. The new initial expectation $\mathrm{E}(F_0)$ is at least $p_1^k(1 - 2^{-2^k}) \sim p_1^k$. The new initial moment $\mathrm{M}_d(F_0)$ is at most $2^{1-2^k} p_1^{kd}$. By Theorem 2.6, the final expectation is

$$\begin{aligned} \mathrm{E}(F_n) &\geq \mathrm{E}(F_0) - \frac{\epsilon(\mathrm{M}_d(F_0))^{1/d}}{1 - c^{1/d}} \\ &\geq p_1^k \left(1 - 2^{-2^k}\right) - \frac{\epsilon \left(2^{1-2^k} p_1^{kd}\right)^{1/d}}{1 - c^{1/d}} \\ &\geq p_1^k \left[1 - \frac{1}{2^{2^k}} - \frac{2\epsilon}{(1 - c^{1/d})2^{2^k}}\right]. \end{aligned}$$

Therefore, given $\epsilon < \frac{1}{2}$, we can choose $k = k(\epsilon, c, d) = k(\epsilon)$ sufficiently large and independent of n so that the final expectation is bounded away from 0, if our initial distribution were F_0.

Compare the random variables D_k and F_0. It is easy to see that the smallest nonzero value D_k can attain is p_1^k. Also, $\Pr(D_k = 0) = 2^{-2^k}$. From these, it is immediate that $F_0 \prec D_k$. Now, note that the biased-mean function $B(x_1, x_2)$ is increasing in x_1 and x_2. Applying Lemma 2.7, we get $\beta(F_i) \prec \beta(D_{k+i})$. Iterating, $F_n \prec D_{n+k}$. Therefore, $\mathrm{E}(D_n) \geq \mathrm{E}(F_{n-k})$; this gives the result. □

Not only is the expected value of $\Pr_\epsilon(S)$ bounded away from 0, but in fact almost every S will have $\Pr_\epsilon(S)$ bounded away from 0.

THEOREM 2.9. *For every $\epsilon < \frac{1}{2}$, almost every set $S \subseteq \{0,1\}^n$ has $\Pr_\epsilon(S)$ bounded away from 0.*

Proof. We know that the variance $\mathrm{Var}(D_n) = \frac{1}{2}\mathrm{M}_2(D_n)$. Applying the Power-Mean inequality [9] followed by iterating Theorem 2.5, we get $\mathrm{Var}(D_n) \leq \frac{1}{2}(c^n M_d(D_0))^{2/d}$. Therefore, there exist positive constants γ, δ so that $\mathrm{Var}(D_n) \leq \delta 2^{-\gamma n}$. By Chebyshev's inequality [3], $\Pr(|D_n - \mathrm{E}(D_n)| > t) \leq \frac{\mathrm{Var}(D_n)}{t^2}$. Thus, there is a choice of $\gamma' = \gamma'(\gamma, \delta)$, so that with probability $1 - o(1)$, $|D_n - \mathrm{E}(D_n)| \leq 2^{-\gamma' n}$. In other words, for almost every set S, $\Pr_\epsilon(S) = \mathrm{E}(D_n) + o(1)$. Applying Theorem 2.8, the proof is complete. □

Acknowledgments. We thank Noga Alon and Joel Spencer for some valuable discussions.

REFERENCES

[1] N. ALON AND M. NAOR, *Coin-flipping games immune against linear-sized coalitions*, SIAM J. Comput., 22 (1993), pp. 403–417.

[2] N. ALON AND M. O. RABIN, *Biased coins and randomized algorithms*, in Advances in Computing Research 5, S. Micali, ed., JAI Press Inc., Greenwich, CT, 1989, pp. 499–507.

[3] N. ALON AND J. SPENCER, *The Probabilistic Method*, John Wiley & Sons Inc., New York, 1992.

[4] M. BEN-OR AND N. LINIAL, *Collective coin flipping*, in Advances in Computing Research 5, S. Micali, ed., JAI Press Inc., Greenwich, CT, 1989, pp. 91–116.
[5] M. BEN-OR, N. LINIAL, AND M. SAKS, *Collective coin flipping and other models of imperfect randomness*, in Proc. 7th Hungarian Conference on Combinatorics, Colloq. Math. Soc. János Bolyai 52, North-Holland, Amsterdam, 1987, pp. 77–112.
[6] R. BOPPANA AND B. NARAYANAN, *The biased coin problem*, in Proc. 25th Annual ACM Symposium on the Theory of Computing, 1993, pp. 252–257.
[7] ———, *Perfect-information leader election with optimal resilience*, manuscript.
[8] B. CHOR AND O. GOLDREICH, *Unbiased bits from sources of weak randomness and probabilistic communication complexity*, SIAM J. Comput., 17 (1988), pp. 230–261.
[9] G. H. HARDY, J. E. LITTLEWOOD, AND G. PÓLYA, *Inequalities*, 2nd ed., Cambridge University Press, Cambridge, 1952.
[10] H. F. MURRY, *A general approach for generating natural random variables*, IEEE Trans. Comput., C–19 (1970), pp. 1210–1213.
[11] M. SANTHA AND U. V. VAZIRANI, *Generating quasi-random sequences from semi-random sources*, J. Comput. System Sci., 33 (1986), pp. 75–87.
[12] E. SHAMIR, *A Slightly Random Source Confronts a Random Witness-Set*, Technical Report CS–87–9d, Leibniz Center for Research in Computer Science, Hebrew University, Jerusalem, 1988.
[13] U. V. VAZIRANI, *Strong communication complexity or generating quasi-random sequences from two communicating semi-random sources*, Combinatorica, 7 (1987), pp. 375–392.
[14] U. V. VAZIRANI AND V. V. VAZIRANI, *Random polynomial time is equal to slightly-random polynomial time*, in Proc. 26th Annual IEEE Symposium on Foundations of Computer Science, 1985, pp. 417–428.
[15] D. ZUCKERMAN, *Simulating BPP using a general weak random source*, in Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science, 1991, pp. 79–89.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 37–54, February 1996

005

SPANNERS OF HYPERCUBE-DERIVED NETWORKS*

MARIE-CLAUDE HEYDEMANN†, JOSEPH G. PETERS‡, AND DOMINIQUE SOTTEAU†

Abstract. A spanning subgraph G' of a simple undirected graph G is a *t-spanner* of G if every pair of vertices that are adjacent in G are at distance at most t in G'. The parameter t is called the *dilation* of the spanner. Spanners with small dilations have many applications, such as their use as low-cost approximations of communication networks with only small degradations in performance. In this paper, we derive spanners with small dilations for four closely related bounded-degree approximations of hypercubes: butterflies, cube-connected cycles, binary de Bruijn graphs, and shuffle-exchange graphs. We give both direct constructions and methods for deriving spanners for one class of graphs from spanners for another class. We prove that most of our spanners are minimum in the sense that spanners with fewer edges have larger dilations.

Key words. networks, spanning subgraphs, butterflies, cube-connected cycles, shuffle-exchange graphs, de Bruijn graphs

AMS subject classifications. 68M10, 05C12

1. Introduction.

Given a simple undirected graph G, a subgraph G' of G is a *t-spanner* of G if G' is a spanning subgraph of G, and every pair of vertices that are adjacent in G are at distance no greater than t in G'. From this definition, it is immediate that the distance between any pair of vertices in G' is at most t times greater than the distance in G. The factor t is called the *dilation* of the spanner. (Some authors use the term *stretch factor* instead of *dilation* [6], [14].) It is also immediate from the definition of spanner that any graph is a 1-spanner of itself and that a t-spanner of a graph is also a t'-spanner of the graph for every $t' > t$. Combining these two observations, any graph is a t-spanner of itself for every $t \geq 1$. We will use the terms *proper spanner* and *proper t-spanner* to refer to spanners that are proper subgraphs of the graphs that they span.

If G' has the smallest possible number of edges of any t-spanner of G, then G' is a *minimum t-spanner* of G. A minimum t-spanner is the least expensive approximation of a network in terms of number of communication links, for which the degradation in performance can be guaranteed not to exceed a factor of t. If t is small, then a minimum t-spanner can be a practical alternative to a communication network that is too expensive or difficult to build.

Spanners have many applications in addition to their use as approximations of communication networks. They have been used to transform synchronous distributed algorithms into asynchronous algorithms [2], [18], to solve motion planning problems [7], to construct small routing tables [19], to broadcast efficiently [3], and even to solve genetics problems [4].

Spanners are often difficult to find, and most problems involving spanners are *NP*-hard [17], [5], [6]. To date, much of the research on spanners has concentrated on the

* Received by the editors April 25, 1994; accepted for publication (in revised form) January 10, 1995. Part of this research was performed while the second author was visiting Université de Paris-Sud and while the third author was visiting McGill University and Simon Fraser University.

† Laboratoire de Recherche en Informatique, Unité de Recherche Associée, 410 du Centre National de la Recherche Scientifique, Bât. 490, Université de Paris-Sud, 91405 Orsay, France (mch@lri.fr and sotteau@lri.fr). The research of these authors was partially supported by project PRC C3 of the Centre National de la Recherche Scientifique.

‡ School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. The research of this author was partially supported by the Natural Sciences and Engineering Research Council of Canada (peters@cs.sfu.ca).

construction of spanners of graphs with small diameters and large average or maximum degrees such as hypercubes [13], [18] and various types of complete graphs [7], [8], [23], and graphs with small degrees and large diameters such as grids [14] and pyramids [21]. In this paper, we study spanners of graphs that have small constant degrees and small diameters. In particular, we derive minimum t-spanners for small t for four bounded-degree approximations of hypercubes: butterflies, cube-connected cycles, binary de Bruijn graphs, and shuffle-exchange graphs. We give both direct constructions of spanners and methods for deriving spanners for one class of graphs from spanners for another class.

The study of spanners is closely related to the study of *graph embeddings*. In a graph embedding problem, the goal is to assign the vertices of a *guest graph* to the vertices of a *host graph* in such a way that some parameter, such as *dilation* or *congestion*, is minimized. A spanner is, in fact, a graph embedding in which the host graph is restricted to being a spanning subgraph of the guest graph. We refer the reader to [16] for a survey of the many graph embedding results.

In the next section of this paper, we define the four classes of graphs, discuss some of their properties, and describe mappings and relationships among the classes. The four classes are quite closely related to each other [1], [12], and we will use the mappings later in the paper to derive spanners for one class from spanners for other classes. Some care is needed when using mappings to derive spanners in this way. It is possible for a graph G_3 to be a t-spanner of G_2 and G_2 to be a t'-spanner of G_1 while G_3 is neither a t-spanner nor a t'-spanner of G_1. Figure 1 illustrates this situation.

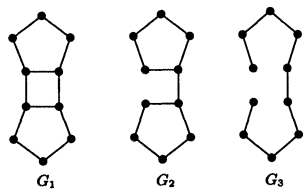


FIG. 1. *G_3 is a 4-spanner of G_2; G_2 is a 3-spanner of G_1; G_3 is a 9-spanner of G_1.*

In §3, we construct spanners for cube-connected cycles and butterfly graphs. Dilations 3 and 7 are the only possible small dilations for large graphs from these classes. We give a constructive characterization of all minimum 3-spanners of butterfly graphs and show that a cube-connected cycles graph of the same order is a minimum 3-spanner with the additional desirable property of being 3-regular. We briefly describe an algorithm to find any 3-regular minimum 3-spanner of a butterfly graph. We then give constructive characterizations of all minimum 7-spanners of butterfly graphs and cube-connected cycles graphs (for all but the first few graphs of each class). We show that some minimum 7-spanners of cube-connected cycles graphs are also minimum 7-spanners of butterfly graphs while others are not, and we explain why this is the case. Small dilation spanners of small cube-connected cycles and butterfly graphs are briefly discussed. Our results regarding small dilation spanners of cube-connected cycles and butterfly graphs are summarized in a table at the end of the section.

In §4, we present our results for shuffle-exchange and binary de Bruijn graphs. We concentrate on dilations 3 and 7, which are the most interesting for these two classes of graphs. We give a constructive characterization of all minimum 3-spanners of binary de Bruijn graphs and show how to obtain a minimum 3-spanner of maximum degree 3 from a shuffle-exchange graph of the same order. We also show that every minimum 3-spanner of a binary de Bruijn graph can be obtained directly from a minimum 3-

spanner of a butterfly graph, and we briefly describe an algorithm to find the minimum 3-spanners with maximum degree 3 of a binary de Bruijn graph. We then show how to obtain 7-spanners of shuffle-exchange graphs from 7-spanners of cube-connected cycles graphs and 7-spanners of binary de Bruijn graphs from 7-spanners of butterfly graphs and 7-spanners of cube-connected cycles graphs. We conclude the section with a table summarizing results regarding small dilation spanners of shuffle-exchange and binary de Bruijn graphs.

2. Definitions, properties, and relationships. Binary strings are used as vertex labels or components of vertex labels for all of the classes of graphs studied in this paper. We use $x(i)$ or $x_0x_1\cdots\overline{x_i}\cdots x_{n-1}$ to denote the binary string obtained from $x = x_0x_1\cdots x_{n-1}$ by complementing the bit in position i. Similarly, $x(i,j)$ and $x_0x_1\cdots\overline{x_i}\cdots\overline{x_j}\cdots x_{n-1}$ mean x with bits i and j complemented, and so on. The *parity* of a vertex is either even or odd, depending on the number of 1 bits in the binary representation of its label. We use i^j to denote a string of j i's. We will abbreviate "the vertex with label x" to "vertex x."

The four classes of graphs that we consider in this paper are bounded-degree approximations of *binary hypercubes.* The n-dimensional binary hypercube $H(n)$ has 2^n vertices which are labelled with the binary strings of length n. Two vertices of $H(n)$ are adjacent if their labels differ in exactly one bit position. The edge $[x, x(i)]$ connecting vertices x and $x(i)$ in $H(n)$ is called a *dimension* i edge and x and $x(i)$ are *dimension* i neighbours. We will only consider graphs with $n \geq 3$ dimensions in this paper; the spanners of the four bounded-degree approximations of $H(2)$ are not very interesting and some of our general techniques fail on such small graphs.

The binary *shuffle-exchange graph* of dimension n, denoted $SE(n)$, has 2^n vertices with the same labels as the vertices of $H(n)$. Each vertex $x_0x_1\cdots x_{n-1}$ is connected to its *shuffle neighbour* $x_1x_2\cdots x_{n-1}x_0$ and its *unshuffle neighbour* $x_{n-1}x_0\cdots x_{n-2}$ by *shuffle edges*, and to its *exchange neighbour* $x_0x_1\cdots\overline{x_{n-1}}$ by an *exchange edge.* The shuffle edges of the vertices 0^n and 1^n are removed since they are self-loops. If n is even, then one of the two parallel shuffle edges between $0101\cdots01$ and $1010\cdots10$ is also removed.

The *binary de Bruijn graph* of dimension n, denoted $UB(n)$, has the same vertex set as $H(n)$ and $SE(n)$. As in $SE(n)$, each vertex $x_0x_1\cdots x_{n-1}$ is connected to its shuffle neighbour $x_1x_2\cdots x_{n-1}x_0$ and its unshuffle neighbour $x_{n-1}x_0\cdots x_{n-2}$ by shuffle edges. *Shuffle-exchange edges* connect $x_0x_1\cdots x_{n-1}$ to its *shuffle-exchange neighbour* $x_1x_2\cdots x_{n-1}\overline{x_0}$ and its *unshuffle-exchange neighbour* $x_{n-1}x_0\cdots\overline{x_{n-2}}$. As in $SE(n)$, the shuffle edges at vertices 0^n and 1^n are self-loops and are removed. One of the two parallel shuffle or shuffle-exchange edges between $0101\cdots$ and $1010\cdots$ is also removed.

The *cube-connected cycles* graph of dimension n, denoted $CCC(n)$, is derived from $H(n)$ by replacing each vertex x of $H(n)$ by a cycle of length n. We call such a cycle a *fundamental cycle* of $CCC(n)$. Each of the n vertices on the fundamental cycle that replaces x inherits one of the n edges that were incident to x in $H(n)$. The vertices of a *fundamental cycle* are labelled with pairs (i,x), $0 \leq i \leq n-1$, where i is called the *level* of the vertex.[1] *Level* i of $CCC(n)$ is the set of all vertices with level i. The edges that connect (i,x) to its neighbours $(i+1,x)$ and $(i-1,x)$ on its fundamental cycle are called *C-edges.* The edge that (i,x) inherits from the hypercube vertex x is called an *H-edge.* The H-edge of (i,x) connects it to vertex $(i,x(i))$. We

[1] In this paper, all arithmetic on indices and levels is assumed to be mod$n$.

will abuse terminology slightly by saying that the fundamental cycles containing (i, x) and $(i, x(i))$ are *adjacent* in dimension i.

The *butterfly* graph of dimension n, $BF(n)$, is derived in a similar way to $CCC(n)$. $BF(n)$ has the same vertex set as $CCC(n)$ and the same fundamental cycles and C-edges. The difference is that $BF(n)$ has two H-edges corresponding to each former hypercube edge instead of one. The H-edges of vertex (i, x) in $BF(n)$ are $[(i, x), (i + 1, x(i))]$ and $[(i, x), (i - 1, x(i - 1))]$. Note that there are two H-edges between two fundamental cycles that are adjacent in dimension i: $[(i, x), (i + 1, x(i))]$ and $[(i + 1, x), (i, x(i))]$.

Observation 2.1 below follows immediately from the definitions of $CCC(n)$ and $BF(n)$. Table 1 lists some of the characteristics of the graph classes that we have defined. The diameter of $CCC(n)$ shown in the table is from [15] and is for $n > 3$. The diameter of $CCC(3)$ is 6.

TABLE 1
Some parameters of hypercube-derived graphs.

	vertices	edges	degree	diameter
$H(n)$	2^n	$n \cdot 2^{n-1}$	regular n	n
$SE(n)$	2^n	$\frac{3}{2} \cdot 2^n - 3 + (n \bmod 2)$	maximum 3	$2n - 1$
$UB(n)$	2^n	$2 \cdot 2^n - 3$	maximum 4	n
$CCC(n)$	$n2^n$	$\frac{3}{2} \cdot n2^n$	regular 3	$2n + \lfloor \frac{n}{2} \rfloor - 2$, $n > 3$
$BF(n)$	$n2^n$	$2 \cdot n2^n$	regular 4	$n + \lfloor \frac{n}{2} \rfloor$

Observation 2.1. The graph $H(n)$ results when each fundamental cycle of $BF(n)$ or $CCC(n)$ is contracted into one vertex and parallel edges are eliminated.

Figure 2 illustrates the close relationships among $H(n)$, $CCC(n)$, and $BF(n)$. In the figure, the horizontal edges are the C-edges of $CCC(3)$ and $BF(3)$. The numbers across the top of the diagram are the levels. To simplify the figure, the fundamental cycles of $CCC(3)$ and $BF(3)$ have been "cut" at level 0, and the level 0 vertices of each graph have been duplicated. In the right part of the figure, $CCC(3)$ is obtained from $BF(3)$ by deleting the dashed edges.

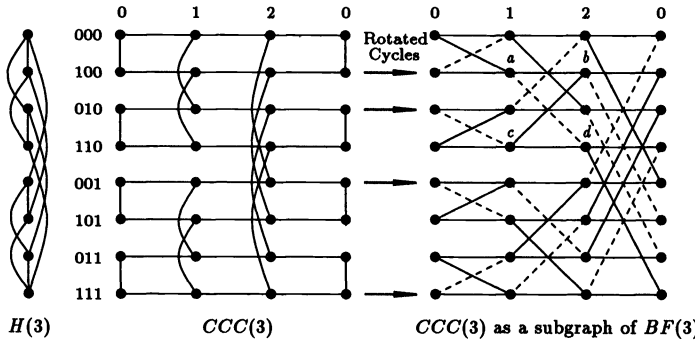


FIG. 2. *Relationships among hypercubes, cube-connected cycles, and butterflies.*

The following two propositions describe mappings between classes of graphs. We will use these mappings later in the paper to derive spanners. The homomorphisms ϕ and ψ used in Proposition 2.3 were studied in an algebraic context in [1].

PROPOSITION 2.2 (see [10]). *$SE(n)$ is a spanning subgraph of $UB(n)$ and $CCC(n)$ is a spanning subgraph of $BF(n)$.*

Proof. The mapping of vertices of $SE(n)$ onto vertices of $UB(n)$, which shows that $SE(n)$ is a spanning subgraph of $UB(n)$, is $f(x) = x$ if x has odd parity and $f(x_0 \cdots x_{n-1}) = x_{n-1}x_0 \cdots x_{n-2}$ if $x = x_0 \cdots x_{n-1}$ has even parity. The mapping of vertices of $CCC(n)$ onto vertices of $BF(n)$, which shows that $CCC(n)$ is a spanning subgraph of $BF(n)$, is $g(i,x) = (i,x)$ if x has even parity and $g(i,x) = (i+1,x)$ if x has odd parity. □

Intuitively, the mapping g of vertices of $CCC(n)$ onto vertices of $BF(n)$ rotates the labels of fundamental cycles corresponding to hypercube vertices with odd parity by one position (i.e., one level) so that a vertex at level i moves to level $i+1$. Figure 2 shows these rotations of labels. It is easy to see that g induces a one-to-one mapping of the C-edges of $CCC(n)$ onto the C-edges of $BF(n)$ and of the H-edges of $CCC(n)$ onto half of the H-edges of $BF(n)$.

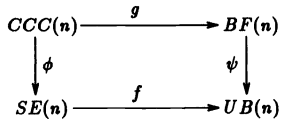


FIG. 3. *Mappings among hypercube-derived graphs.*

PROPOSITION 2.3. *There is a homomorphism (of graphs) ϕ from $CCC(n)$ onto $SE(n)$, and there is a homomorphism ψ from $BF(n)$ onto $UB(n)$, such that $\psi \circ g = f \circ \phi$, where f and g are the mappings of vertices of $SE(n)$ onto $UB(n)$, and $CCC(n)$ onto $BF(n)$, respectively, from the proof of Proposition* 2.2.

Proof. The homomorphism $\phi(i, x_0x_1 \cdots x_{n-1}) = x_{i+1}x_{i+2} \cdots x_i$ of the vertices of $CCC(n)$ onto the vertices of $SE(n)$ maps each edge of $CCC(n)$ onto an edge of $SE(n)$. Similarly, the homomorphism $\psi(i, x_0x_1 \cdots x_{n-1}) = x_ix_{i+1} \cdots x_{i-1}$ of the vertices of $BF(n)$ onto the vertices of $UB(n)$ maps each edge of $BF(n)$ onto an edge of $UB(n)$. It is now mechanical to verify the equality $\psi \circ g = f \circ \phi$. □

The relationships of Propositions 2.2 and 2.3 are shown in Fig. 3 and later in Fig. 6 (and Fig. 7). In the left part of Fig. 6, the labels next to the vertices of $CCC(3)$ in the interior of the graph are (the labels of) the vertices of $SE(3)$ to which they are mapped by ϕ. Similarly, the labels in the interior of $BF(3)$ in the right part of the figure are the vertices of $UB(3)$ to which ψ maps the vertices of $BF(3)$.

We conclude this section with a discussion of some properties of cycles in $UB(n)$, $BF(n)$, and $CCC(n)$. Some of these properties are based on results from [11] and others are obtained using homomorphism ψ and results from [1].

Each vertex (i,x) of $BF(n)$ belongs to two 4-cycles which we call *basic 4-cycles*. One basic 4-cycle has vertices (i,x), $(i+1,x)$, $(i,x(i))$, and $(i+1,x(i))$. The other basic 4-cycle has vertices (i,x), $(i-1,x)$, $(i,x(i-1))$, and $(i-1,x(i-1))$. Note that each basic 4-cycle is an alternating sequence of C- and H-edges. The basic 4-cycles of $BF(3)$ are clearly visible in Fig. 2. It is easily verified that each edge of $BF(n)$ belongs to a unique basic 4-cycle. Thus, a butterfly graph can be viewed as the union of basic 4-cycles. We state this well-known result as an observation.

Observation 2.4. For $n \geq 3$, $BF(n)$ is the edge-disjoint union of $n \cdot 2^{n-1}$ basic 4-cycles.

The next proposition describes a graph $D(n)$ that is derived from the basic 4-cycles of $BF(n)$. We will use this result to establish structural properties of $BF(n)$. The proof of Proposition 2.5 is based on a similar proof in [11] for the *fast Fourier transform* graphs.

PROPOSITION 2.5. *For $n \geq 3$, let $D(n)$ be the graph obtained from $BF(n)$ as follows. The vertices of $D(n)$ correspond to the basic 4-cycles of $BF(n)$. There is an edge between two vertices in $D(n)$ if the corresponding basic 4-cycles in $BF(n)$ have a common vertex. Then $D(n)$ is an edge-disjoint union of 4-cycles.*

Proof. Consider the basic 4-cycle of $BF(n)$, which has vertices (i,x), $(i+1,x)$, $(i,x(i))$, and $(i+1,x(i))$ where $x = x_0 \cdots x_{n-1}$. Call this cycle X. X corresponds to a single vertex in $D(n)$ which we will label $(i, x_0 \cdots x_{i-1}x_{i+1} \cdots x_{n-1})$, or $(i, x\langle i\rangle)$ for short (with indices taken mod$n$ as usual). Each vertex of X in $BF(n)$ is on a second basic 4-cycle in $BF(n)$, and each of these four basic 4-cycles corresponds to one of the neighbours of vertex $(i, x\langle i\rangle)$ in $D(n)$. Using the same labelling convention that we used for the vertex corresponding to X, the four neighbours of $(i, x\langle i\rangle)$ in $D(n)$ are $(i-1, x\langle i-1\rangle)$, $(i+1, x\langle i+1\rangle)$, $(i-1, x_0 \cdots x_{i-2}\overline{x_i} \cdots x_{n-1}) = (i-1, x(i)\langle i-1\rangle)$, and $(i+1, x(i)\langle i+1\rangle)$. It can be verified that each vertex $(i, x\langle i\rangle)$ of $D(n)$ is on two *basic 4-cycles of* $D(n)$ and that each edge of $D(n)$ is on one basic 4-cycle of $D(n)$. The vertices on one basic 4-cycle with $(i, x\langle i\rangle)$ are $(i+1, x\langle i+1\rangle)$, $(i, x(i+1)\langle i\rangle)$, and $(i+1, x(i)\langle i+1\rangle)$. The vertices on the other basic 4-cycle with $(i, x\langle i\rangle)$ are $(i-1, x\langle i-1\rangle)$, $(i, x(i-1)\langle i\rangle)$, and $(i-1, x(i)\langle i-1\rangle)$. □

The graph $D(n)$ of Proposition 2.5 and $BF(n-1)$ have similar structures in that each graph is the edge-disjoint union of 4-cycles. However, the number of vertices in $D(n)$ is $n \cdot 2^{n-1}$, the same as the number of basic 4-cycles of $BF(n)$, whereas $BF(n-1)$ has only $(n-1) \cdot 2^{n-1}$ vertices. The two graphs have the same number (2^{n-1}) of fundamental cycles, but the fundamental cycles of $D(n)$ have length n while the fundamental cycles of $BF(n-1)$ have length $n-1$.

In the proof of Proposition 2.5, each basic 4-cycle of $BF(n)$ corresponds to a single vertex in the graph $D(n)$, and each vertex of $D(n)$ is on two basic 4-cycles of $D(n)$. Thus, each basic 4-cycle of $BF(n)$ is on two *4-cycles of basic 4-cycles* in $BF(n)$. This property of $BF(n)$ is most easily seen by looking at the neighbourhoods of the basic 4-cycles labelled *abcd* in Figs. 2 and 4. (In both figures, the edges that are in $BF(n)$ but not in $CCC(n)$ are shown as thin dashed lines.) We will use this property several times in the next section, so we state it as a proposition.

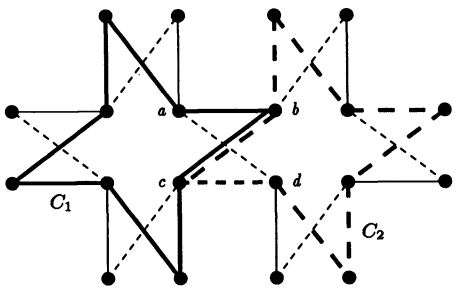


FIG. 4. *The basic 4-cycle abcd is on two 4-cycles of basic 4-cycles.*

PROPOSITION 2.6. *Each basic 4-cycle of $BF(n)$ is on two 4-cycles of basic 4-cycles.*

A binary de Bruijn graph can also be viewed as a union of basic cycles. Each vertex x of $UB(n)$ except 0^n and 1^n belongs to two basic cycles. 0^n and 1^n each belong to one basic cycle. All but two of the basic cycles are 4-cycles and are alternating sequences of shuffle and shuffle-exchange edges. The basic cycles containing 0^n and 1^n are 3-

cycles. In §4, we will use the following propositions and corollaries to derive spanners for $UB(n)$ from spanners of $BF(n)$. Proposition 2.7 is from [1]. We have included a proof because some of the details of the proof will be used in §4. Corollary 2.9 is from [11], and Proposition 2.10 follows directly from a result in [11] for binary de Bruijn digraphs.

PROPOSITION 2.7 (see [1]). *$UB(n)$ can be obtained from $BF(n)$ by taking the image under ψ of any two adjacent levels i and $i+1$ of $BF(n)$ and then removing the loops at 0^n and 1^n and one of the two duplicate edges $[0101\cdots, 1010\cdots]$.*

Proof. The homomorphism $\psi(i, x_0x_1\cdots x_{n-1}) = x_ix_{i+1}\cdots x_{i-1}$ from Proposition 2.3 defines a bijection between the vertices of any level i of $BF(n)$ and the vertices of $UB(n)$. For any given i, ψ maps each of the 2^n basic 4-cycles of $BF(n)$ with edges between levels i and $i+1$ onto a basic 4-cycle (or basic 3-cycle) of $UB(n)$. More precisely, the basic 4-cycle (i,x), $(i+1,x)$, $(i,x(i))$, $(i+1,x(i))$ of $BF(n)$ with $x = x_0x_1\cdots x_{n-1}$ is mapped to the basic 4-cycle $x_ix_{i+1}\cdots x_{i-1}$, $x_{i+1}\cdots x_{i-1}x_i$, $\overline{x_i}x_{i+1}\cdots x_{i-1}$, $x_{i+1}\cdots x_{i-1}\overline{x_i}$ of $UB(n)$ unless $x = 0^n$, $x(i) = 0^n$, $x = 1^n$, or $x(i) = 1^n$. If $x = 0^n$ or $x(i) = 0^n$, then the image in $UB(n)$ is the basic 3-cycle 0^n, $0^{n-1}1$, 10^{n-1} because the vertices $(i, 0^n)$ and $(i+1, 0^n)$ are both mapped to the vertex 0^n of $UB(n)$. Similarly, if $x = 1^n$ or $x(i) = 1^n$, then the image in $UB(n)$ is the basic 3-cycle 1^n, $1^{n-1}0$, 01^{n-1}. Also, each vertex y of $UB(n)$ is the image under ψ of one level i vertex and of one level $i+1$ vertex of $BF(n)$. The level i vertex has two neighbours in level $i+1$, and these neighbours are mapped to the shuffle and shuffle-exchange neighbours of y. The level $i+1$ vertex has two neighbours in level i, and these neighbours are mapped to the unshuffle and unshuffle-exchange neighbours of y. Thus, every edge between levels i and $i+1$ of $BF(n)$ is mapped to a different edge of $UB(n)$, except edges $[(i,0^n),(i+1,0^n)]$ and $[(i,1^n),(i+1,1^n)]$, which map onto loops at vertices 0^n and 1^n, respectively, and the two edges which map onto $[0101\cdots, 1010\cdots]$. If n is even, then these two edges are $[(i, 0101\cdots), (i+1, 0101\cdots)]$ and $[(i, 1010\cdots), (i+1, 1010\cdots)]$. If n is odd, the two edges are $[(i, \cdots 0100101\cdots), (i+1, \cdots 0101101\cdots)]$ and $[(i, \cdots 1011010\cdots), (i+1, \cdots 1010010\cdots)]$, where the bit positions shown are $i-3$ through $i+3$ (i.e., the two adjacent 0's in $(i, \cdots 0100101\cdots)$ are in positions $i-1$ and i). □

COROLLARY 2.8. *Any two edge-disjoint basic 4-cycles of $BF(n)$ are mapped by ψ onto either the same basic 4- or 3-cycle of $UB(n)$ or onto edge-disjoint basic 4- or 3-cycles of $UB(n)$.*

Proof. It is immediate from Proposition 2.7 that any two 4-cycles of $BF(n)$ using edges between the same two adjacent levels are mapped onto edge-disjoint basic 4- or 3-cycles of $UB(n)$. In addition, n edge-disjoint basic 4-cycles of $BF(n)$ (one between each pair of adjacent levels) are mapped by ψ onto the same basic 4- or 3-cycle of $UB(n)$. □

COROLLARY 2.9 (see [11]). *For $n \geq 3$, $UB(n)$ is the union of $2^{n-1}-2$ basic 4-cycles and two basic 3-cycles, and this union is edge-disjoint except for edge $[0101\cdots, 1010\cdots]$.*

Proof. The number of basic 4-cycles between two levels i and $i+1$ of vertices of $BF(n)$ is 2^{n-1}. The basic 4-cycles containing 0^n and 1^n are mapped by ψ onto basic 3-cycles of $UB(n)$. All other basic 4-cycles are mapped by ψ on edge-disjoint 4-cycles of $UB(n)$. □

PROPOSITION 2.10 (see [11]). *For $n \geq 3$, $UB(n-1)$ is the graph obtained from $UB(n)$ as follows. The vertices of $UB(n-1)$ correspond to the basic 3-cycles and basic 4-cycles of $UB(n)$. There is an edge in $UB(n-1)$ between two vertices if the*

two corresponding basic cycles in $UB(n)$ have a common vertex.

The following result for binary de Bruijn graphs is the analogue of Proposition 2.6. It can be proved directly from Corollary 2.9 and Proposition 2.10 in the same way that Proposition 2.6 was proved directly from Observation 2.4 and Proposition 2.5. Alternatively, the close relationship between $UB(n)$ and $BF(n)$ is emphasized by noting that Proposition 2.11 follows directly from Proposition 2.6 by Proposition 2.7.

PROPOSITION 2.11. *Each basic* 4*-cycle of* $UB(n)$ *is on two* 4- *or* 3*-cycles of basic* 4- *or* 3*-cycles. Each basic* 3*-cycle of* $UB(n)$ *is on one* 3*-cycle of basic* 4- *or* 3*-cycles.*

PROPOSITION 2.12. *If* $n \geq 6$, *every nonfundamental cycle of* $CCC(n)$ *has length* 8 *or length at least* 12. *If* $n \geq 8$, *every nonfundamental cycle of* $BF(n)$ *has length* 4 *or* 8 *or length at least* 12.

Proof. Any nonfundamental cycle of $CCC(n)$ or $BF(n)$ uses an even number of H-edges in any dimension.

(a) Let X be a nonfundamental cycle in $CCC(n)$ with H-edges from r different dimensions. Since no two H-edges are adjacent in $CCC(n)$, there are at least as many C-edges as H-edges in X, so X has at least $4r$ edges. In $CCC(n)$, $r \geq 2$, so $|X| \geq 8$. If $r = 3$, then $|X| \geq 12$. If $r = 2$, but X uses more than two edges in one or both of its dimensions, then it has at least six H-edges and $|X| \geq 12$. Thus, any nonfundamental cycle X of $CCC(n)$ with fewer than 12 edges has $r = 2$ and uses exactly two H-edges in each of two dimensions, say dimensions i and j with $j > i$. But then X must go through the eight vertices (l, x), $(l, x(i))$, $(l, x(i, j))$, and $(l, x(j))$ for some x and $l = i$ and $l = j$. Thus X contains four paths of C-edges, one path in each of the fundamental cycles corresponding to the vertices x, $x(i)$, $x(i, j)$, and $x(j)$ of the hypercube, and each path has length $p = j - i$ or $n - p$. The possible lengths of X are $4p + 4$, $3p + (n - p) + 4 = n + 2p + 4$, and $2p + 2(n - p) + 4 = 2n + 4$ for $1 \leq p \leq n - 1$. None of these gives a cycle length of 9, 10, or 11 when $n \geq 6$.

(b) Let X be a nonfundamental cycle in $BF(n)$ with H-edges from r different dimensions. In $BF(n)$, $r \geq 1$ and H-edges can appear consecutively in X if the H-edges are from consecutive dimensions. If $r = 1$, then X can have length 4, $n + 2$, or $2n$. If $r = 2$ and the dimensions of the H-edges in X do not alternate, then the possible cycle lengths are $n + 4$, $2n + 2$, and $3n$. If $r = 2$ and the two dimensions of the H-edges, say i and j, alternate, then X contains four paths of C-edges, one path in each of the fundamental cycles corresponding to the vertices x, $x(i)$, $x(i, j)$, and $x(j)$ of the hypercube, and each path has length $p = j - i$ or $n - p$ (assuming $j > i$). It is not difficult to check that the possible lengths for X in this case are $4p + 4$, $n + 2p + 4$, $2n + 4$ as in $CCC(n)$ (since $CCC(n)$ is a subgraph of $BF(n)$) plus $n + 2p + 2$, $n + 2p + 6$, $2n$, $2n + 2$, and $2n + 8$ for $1 \leq p \leq n - 1$. If $n \geq 8$, then the only possible lengths for X are 4, 8, and 12 or greater. If $r \geq 3$ and $n \geq 8$, it is not hard to verify that no new cycle lengths less than 12 are possible. □

Remark 2.13. It is immediate from the previous proof that the shortest nonfundamental cycles in $CCC(n)$ are alternating sequences of H- and C-edges of length 8. Each such cycle is a sequence of vertices of the form (i, x), $(i, x(i))$, $(i + 1, x(i))$, $(i + 1, x(i, i + 1))$, $(i, x(i, i + 1))$, $(i, x(i + 1))$, $(i + 1, x(i + 1))$, $(i + 1, x)$, (i, x). The H-edges of this cycle are from the two *adjacent dimensions* i and $i + 1$. It is easy to verify that this is the only possible way to select the dimensions of the H-edges and that each C-edge is on exactly one such 8-cycle. Each H-edge $[(i, x), (i, x(i))]$ is on exactly two 8-cycles; one 8-cycle uses the adjacent dimensions i and $i + 1$ and the other uses adjacent dimensions i and $i - 1$. There are $n2^n$ C-edges in $CCC(n)$, so there are $n2^{n-2}$ nonfundamental 8-cycles, 2^{n-2} for each possible choice of adjacent

dimensions i and $i+1$. The four C-edges on an 8-cycle that uses dimensions i and $i+1$ are of the form $[(i, x_0 \cdots x_{i-1}\alpha\beta x_{i+2} \cdots x_n), (i+1, x_0 \cdots x_{i-1}\alpha\beta x_{i+2} \cdots x_n)]$ with $\alpha\beta \in \{00, 01, 10, 11\}$. □

The lengths of some of the cycles of $CCC(n)$ and $BF(n)$ are also discussed in [20].

3. Spanners of $CCC(n)$ and $BF(n)$. From Proposition 2.12, we know that the shortest cycles of $BF(n)$ containing H-edges are the basic 4-cycles. Thus, removing H-edges results in a t-spanner with $t \geq 3$. The only cycles of $BF(n)$ that do not contain H-edges are the fundamental cycles, so proper 2-spanners are only possible for $BF(3)$.

PROPOSITION 3.1. *A minimum 2-spanner of $BF(3)$ has 40 edges and is obtained by removing one edge from each of the eight fundamental 3-cycles. There is no proper 2-spanner of $BF(n)$ for any $n > 3$.*

THEOREM 3.2. *A minimum 3-spanner of $BF(n)$, $n > 4$, has $\frac{3}{2} \cdot n2^n$ edges and is obtained by removing one edge from each basic 4-cycle.*

Proof. If $n > 4$, then the only 4-cycles in $BF(n)$ are the basic 4-cycles. By Observation 2.4, $BF(n)$ is the edge-disjoint union of $n \cdot 2^{n-1}$ basic 4-cycles. Since each edge is on exactly one 4-cycle, deleting one edge from each 4-cycle gives a 3-spanner for $BF(n)$ with $\frac{3}{2} \cdot n2^n$ edges. Deleting a second edge from any 4-cycle would destroy the only remaining path of length 3 between the endpoints of the first deleted edge, so no 3-spanner of $BF(n)$ can have fewer than $\frac{3}{2} \cdot n2^n$ edges. Furthermore, the only way that a minimum 3-spanner of $BF(n)$ can be obtained is by deleting exactly one edge from each basic 4-cycle. □

PROPOSITION 3.3. *For $3 \leq n \leq 7$ and $\max(3, n-1) \leq t \leq 6$, there is a minimum t-spanner of $BF(n)$ with $\frac{3}{2} \cdot n2^n - 2^{n-1}$ edges.*

Proof. $BF(n)$ is the edge-disjoint union of $n \cdot 2^{n-1}$ basic 4-cycles, so for any $t \geq 3$ a t-spanner can be obtained by deleting one edge from each basic 4-cycle. However, for $3 \leq n \leq 7$, this spanner is not a minimum t-spanner if $\max(3, n-1) \leq t \leq 6$; the 2^n fundamental cycles in $BF(n)$ have length n and all of them could remain intact after the deletion of $n \cdot 2^{n-1}$ edges. If more than one edge is deleted from a basic 4-cycle of $BF(n)$, then none of the edges deleted from the cycle can be an H-edge. This is because an H-edge is on a unique cycle; if it is deleted, then the other three edges on its 4-cycle are needed to span it. Thus, the only way to remove more than one edge from a basic 4-cycle is to remove the two C-edges. When both C-edges are removed from a basic 4-cycle, all other edges of the two fundamental cycles that contained the deleted C-edges must be retained to span the deleted C-edges with a path of length $n-1$. Since there are 2^n fundamental cycles in $BF(n)$, two C-edges can be removed from at most 2^{n-1} of the basic 4-cycles and at most one edge can be removed from the remaining basic 4-cycles. This leaves at least $\frac{3}{2} \cdot n2^n - 2^{n-1}$ edges in the t-spanner. For $\max(3, n-1) \leq t \leq 6$, a t-spanner with $\frac{3}{2} \cdot n2^n - 2^{n-1}$ edges can be constructed by removing one H-edge from each of the basic 4-cycles except those involving one pair of adjacent levels i and $i+1$. All C-edges between levels i and $i+1$ (i.e., both C-edges of each basic 4-cycle involving levels i and $i+1$) are then deleted. For example, if $i = 0$, then for each x the C-edge $[(0, x), (1, x)]$ is removed, and one of the two H-edges $[(j, x), (j+1, x(j))]$ and $[(j, x(j)), (j+1, x)]$ is removed for each $j > 0$. □

By Proposition 2.2, $CCC(n)$ is a spanning subgraph of $BF(n)$ with $\frac{3}{2} \cdot n2^n$ edges. In fact, the following stronger result is true.

THEOREM 3.4. *$CCC(n)$ is a minimum 3-spanner of $BF(n)$ for $n > 4$.*

Proof. To prove that $CCC(n)$ is a minimum 3-spanner of $BF(n)$, we will show

that $CCC(n)$ is missing exactly one edge from each basic 4-cycle of $BF(n)$. In Proposition 2.2, the mapping g from [10] of the vertices of $CCC(n)$ to the vertices of $BF(n)$ induces a one-to-one correspondence between the C-edges of $CCC(n)$ and the C-edges of $BF(n)$, so we only have to check what happens to the H-edges. Each basic 4-cycle of $BF(n)$ is an alternating sequence of C- and H-edges of the form $[(i,x),(i+1,x(i))]$, $[(i+1,x(i)),(i,x(i))]$, $[(i,x(i)),(i+1,x)]$, $[(i+1,x),(i,x)]$. g maps the H-edge $[(i,x),(i,x(i))]$ of $CCC(n)$ to the H-edge $[(i,x),(i+1,x(i))]$ of $BF(n)$ if x has even parity and to $[(i+1,x),(i,x(i))]$ if x has odd parity. Thus, g maps exactly one H-edge of $CCC(n)$ to one H-edge of each basic 4-cycle of $BF(n)$. It follows that $CCC(n)$ can be obtained from $BF(n)$ by deleting from each basic 4-cycle of $BF(n)$ the H-edge that is not the image of an edge under mapping g. This shows that $CCC(n)$ is a 3-spanner of $BF(n)$ because each H-edge of $BF(n)$ is spanned by the other three edges of its basic 4-cycle. $CCC(n)$ is a minimum 3-spanner of $BF(n)$ by Theorem 3.2. □

Theorem 3.2 states that every minimum 3-spanner of $BF(n)$ can be obtained by deleting one edge from each basic 4-cycle. Some of these 3-spanners will have vertices with degree 4. $CCC(n)$ has the additional property that it is a minimum 3-spanner with maximum degree 3. Note that any minimum 3-spanner with maximum degree 3 must be 3-regular. By Proposition 2.6, each basic 4-cycle of $BF(n)$ is on two 4-cycles of basic 4-cycles in $BF(n)$. This leads to an algorithm for finding any 3-regular minimum 3-spanner of $BF(n)$. The basic idea is to start by deleting any edge from any basic 4-cycle. For example, consider the basic 4-cycle *abcd* in Fig. 4. Cycle *abcd* is on two 4-cycles of 4-cycles and deleting an edge from *abcd* reduces to eight the number of ways in which edges can be deleted from each of the 4-cycles of 4-cycles. For example, suppose that edge $[a,b]$ is deleted from *abcd*. An edge incident to c must be deleted to reduce the degree of c to 3 and it cannot be another edge from *abcd*. This leaves only two choices in the other 4-cycle containing c. The deletion of $[a,b]$ also reduces to two the number of choices in each of the other 4-cycles shown in Fig. 4, either directly or indirectly. Each subsequent edge deletion further reduces the remaining choices until all remaining edge deletions are forced. A detailed description of this algorithm and the proof that it produces 3-regular minimum 3-spanners are mechanical, so they are omitted.

The analysis of the spanners of $CCC(n)$ for small n is similar to the analysis for $BF(n)$. By Proposition 2.12, the shortest cycle in $CCC(n)$ containing H-edges has length 8. The only cycles with no H-edges are the fundamental cycles, so proper t-spanners with $t < 7$ are only possible when $n < 8$. To get a minimum $(n-1)$-spanner when $n < 8$, it is necessary and sufficient to delete one C-edge from each fundamental cycle. Thus, we get the following result.

PROPOSITION 3.5. *If $n \geq 8$, $CCC(n)$ has no proper t-spanner for $t < 7$. For $n < 8$, $CCC(n)$ has a minimum $(n-1)$-spanner with $\frac{3}{2} \cdot n2^n - 2^n$ edges and has no proper t-spanners for $t < n-1$.*

THEOREM 3.6. *For $n \geq 3$, a 7-spanner of $CCC(n)$ with $\frac{5}{4} \cdot n2^n$ edges can be obtained by removing one C-edge from each nonfundamental 8-cycle in such a way that no removed C-edge is common to two nonfundamental 8-cycles. Such a spanner is a minimum 7-spanner if and only if $n > 8$, and this is the only way to obtain minimum 7-spanners if $n > 8$.*

Proof. For any $n \geq 3$, a 7-spanner of $CCC(n)$ can be obtained by removing exactly one C-edge from each nonfundamental 8-cycle. If none of the removed C-edges is common to two nonfundamental 8-cycles, then this gives a 7-spanner with $\frac{5}{4} \cdot n2^n$ edges.

For example, deleting the $n2^{n-2}$ C-edges of the form $[(i, x_0 \cdots x_{i-1}00x_{i+2} \cdots x_{n-1}), (i+1, x_0 \cdots x_{i-1}00x_{i+2} \cdots x_{n-1})]$ with $0 \leq i \leq n-1$ and $x_j \in \{0,1\}, j \notin \{i, i+1\}$, gives a 7-spanner. For $n \leq 8$, this construction will leave some intact cycles of length 8 or less. It is not difficult to show that more edges can be removed, so these 7-spanners are not minimum spanners for $n \leq 8$. For $n > 8$, these 7-spanners are minimum because the number of edges that can be deleted cannot exceed the number of 8-cycles of $CCC(n)$. This is because every edge that is deleted must be spanned by a path of length at most 7 and no such path can be used to span two deleted edges. By Proposition 2.12, there are no cycles of length less than 8 in $CCC(n)$ when $n > 8$, so the path spanning a deleted edge must come from an 8-cycle that contained the deleted edge. Since each H-edge is on two 8-cycles, and each C-edge is on one 8-cycle by Remark 2.13, all edges removed from 8-cycles to obtain a minimum 7-spanner must be C-edges. □

THEOREM 3.7. *For $n \geq 3$, there is a 7-spanner of $BF(n)$ with $\frac{5}{4} \cdot n2^n$ edges. For $n > 8$, such a spanner is minimum and can be obtained by removing one edge from each basic 4-cycle and then one edge from each remaining 8-cycle in such a way that at least two edges remain from each basic 4-cycle and no edge common to two nonfundamental 8-cycles is removed.*

Proof. A 7-spanner of $BF(n)$ cannot be a minimum spanner if it contains any 4-cycles. To prove this, we will show that there cannot be a 4-cycle for which all four edges are needed to span edges that have been deleted elsewhere in the graph. First, recall that $BF(n)$ is an edge-disjoint union of 4-cycles by Proposition 2.6, so an edge of a 4-cycle cannot be on a path of length 3 that spans an edge in another 4-cycle. Now it is not difficult to see that, for $n > 8$, any 8-cycle of $BF(n)$ that includes an edge of a 4-cycle must include two consecutive edges of that 4-cycle, one C-edge, and one H-edge. Consider the 4-cycle *abcd* in Fig. 4. Every 8-cycle that includes an edge of *abcd* must go through *abc*, *adc*, *bad*, or *bcd*. If an edge e not on cycle *abcd* is deleted and a path of length 7 spanning e goes through both a and c, then the path can use either edge $[a, b]$ or edge $[c, d]$. Similarly, a path that goes through both b and d can use either $[a, b]$ or $[c, d]$. Thus, if either $[a, b]$ or $[c, d]$ is deleted from *abcd*, there will still be a path of length 7 through *abcd* that spans e. Clearly, the edge that is deleted from *abcd* is spanned by a path of length 3 through the remaining edges of *abcd*. It follows that at least one edge of each 4-cycle of $BF(n)$ must be deleted to obtain a minimum 7-spanner. Every possible set of edge deletions from $BF(n)$ that removes exactly one edge from each basic 4-cycle leaves exactly one 8-cycle in each 4-cycle of basic 4-cycles. To see this, examine Fig. 4. There are $n2^{n-2}$ 4-cycles of basic 4-cycles in $BF(n)$, so the number of 8-cycles remaining after one edge is deleted from each basic 4-cycle of $BF(n)$ is the same as the number of 8-cycles in $CCC(n)$. The argument in the proof of Theorem 3.6 can now be used to show that a minimum 7-spanner of $BF(n)$ has at least as many edges as a minimum 7-spanner of $CCC(n)$.

Now, to prove that a minimum 7-spanner of $BF(n)$ has at most $\frac{5}{4} \cdot n2^n$ edges, we will prove that the minimum 7-spanner of $CCC(n)$ given in the proof of Theorem 3.6 is a 7-spanner of $BF(n)$. Any minimum 7-spanner of $CCC(n)$ can be obtained from $BF(n)$ by the following two steps. In the first step, one H-edge is removed from each basic 4-cycle of $BF(n)$ to obtain $CCC(n)$. In the second step, one C-edge is removed from each 8-cycle of $CCC(n)$. The C-edges deleted in the second step are spanned by paths of length at most 7. So, to show that the spanner in the proof of Theorem 3.6 is a 7-spanner of $BF(n)$, it is sufficient to prove that each of the H-edges deleted in the first step is still spanned by a path of length at most 7. By

examining Fig. 4, it is not difficult to see that the H-edges deleted from $BF(n)$ to form $CCC(n)$ (shown as thin dashed lines) are still spanned by paths of length 7 in a minimum 7-spanner of $CCC(n)$ if the C-edges that are removed from the 8-cycles of $CCC(n)$ in step 2 are chosen so that at most one C-edge is removed from each basic 4-cycle of $BF(n)$. The C-edges deleted in the proof of Theorem 3.6 have the form $[(i, x_0 \cdots x_{i-1}00x_{i+2} \cdots x_{n-1}), (i+1, x_0 \cdots x_{i-1}00x_{i+2} \cdots x_{n-1})]$ with $0 \leq i \leq n-1$ and $x_j \in \{0,1\}, j \notin \{i, i+1\}$. Half of these edges map to C-edges of $BF(n)$ with the same labels on the endpoints (when the sum of the x_j's is even) and the other half map to C-edges of the form $[(i+1, x_0 \cdots x_{i-1}00x_{i+2} \cdots x_{n-1}), (i+2, x_0 \cdots x_{i-1}00x_{i+2} \cdots x_{n-1})]$ (when the parity is odd). No two of these deleted C-edges can be on the same basic 4-cycle of $BF(n)$; if one of the C-edges is $[(i, x), (i+1, x)]$, then the other C-edge on the same basic 4-cycle is $[(i, x(i)), (i+1, x(i))]$. □

COROLLARY 3.8. *A minimum 7-spanner of $CCC(n)$ that is a 7-spanner of $BF(n)$ is a minimum 7-spanner of $BF(n)$ for any $n > 8$.*

The proof of Theorem 3.7 shows that some 7-spanners of $CCC(n)$, including the 7-spanner described in the proof of Theorem 3.6, are 7-spanners of $BF(n)$. Figure 5(a) shows a 7-spanner of $CCC(3)$ that is also a 7-spanner of $BF(3)$. In the figure, solid and dotted lines indicate edges that are in both $CCC(3)$ and $BF(3)$ while dashed edges are in $BF(3)$ but not $CCC(3)$. The dotted edges are deleted to obtain a 7-spanner of both $CCC(3)$ and $BF(3)$.

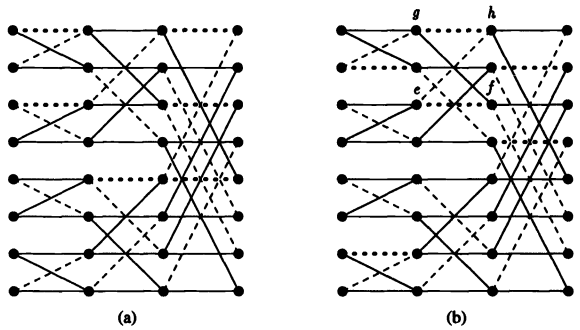


FIG. 5. *7-spanners of $CCC(n)$. The spanner in* (a) *is also a 7-spanner of $BF(n)$; the spanner in* (b) *is not.*

However, not all 7-spanners of $CCC(n)$ are 7-spanners of $BF(n)$. We can see why a 7-spanner of $CCC(n)$ is not necessarily a 7-spanner of $BF(n)$ by looking at 4-cycles of basic 4-cycles in Fig. 4. The two 8-cycles C_1 and C_2 of $CCC(n)$ use the edges of the basic 4-cycle $abcd$ of $BF(n)$. C_1 is shown with heavy solid lines and C_2 is shown with heavy dashed lines. Deleting one C-edge from each 8-cycle gives a minimum 7-spanner of $CCC(n)$, but if the two deleted edges are ab and cd, then the edge ad of $BF(n)$ is not spanned by a path of length 7 or less. The spanner in Fig. 5(b) is a 7-spanner of $CCC(3)$, but two C-edges have been removed from the basic 4-cycle $efgh$, and the H-edge eh is not spanned by a path of length 7 or less.

The arguments from the proofs of Theorems 3.6 and 3.7 can be used to construct 7-spanners of $CCC(n)$ and $BF(n)$ with $\frac{5}{4} \cdot n2^n$ edges for $n \leq 8$. However, these 7-spanners are not minimum spanners because the fundamental cycles are short enough to provide paths of length 7 or less. The presence of these short paths permits the construction of spanners with fewer edges.

Table 2 summarizes results proved in this section about minimum t-spanners of

$CCC(n)$ and $BF(n)$ for small t. The first line shows the numbers of edges in $CCC(n)$ and $BF(n)$ since dilation 1 does not permit the removal of any edges.

TABLE 2
Numbers of edges in small dilation minimum spanners of $CCC(n)$ and $BF(n)$.

Dilation	n	$CCC(n)$	$BF(n)$
1	$n \geq 3$	$\frac{3}{2} \cdot n2^n$	$2 \cdot n2^n$
2	$n = 3$	$\frac{3}{2} \cdot n2^n - 2^n = 28$ Proposition 3.5	$2n \cdot 2^n - 2^n = 40$ Proposition 3.1
	$n > 3$	$\frac{3}{2} \cdot n2^n$ Proposition 3.5	$2 \cdot n2^n$ Proposition 3.1
$3 \leq t \leq 6$	$3 \leq n \leq t+1$	$\frac{3}{2} \cdot n2^n - 2^n$ Proposition 3.5	$\frac{3}{2} \cdot n2^n - 2^{n-1}$ Proposition 3.3
	$n > t+1$	$\frac{3}{2} \cdot n2^n$ Proposition 2.12 & Proposition 3.5	$\frac{3}{2} \cdot n2^n$ Proposition 2.12 & Theorem 3.2
7	$n > 8$	$\frac{5}{4} \cdot n2^n$ Proposition 3.6	$\frac{5}{4} \cdot n2^n$ Proposition 3.7

4. Spanners of $UB(n)$ and $SE(n)$. $UB(n)$ is *pancyclic*, that is, $UB(n)$ has at least one cycle of each length i, $3 \leq i \leq n$ [24], so there is a proper t-spanner of $UB(n)$ for every t, $2 \leq t \leq n-1$. $SE(n)$ also has many cycle lengths but is not pancyclic. However, for large n, almost all of the short cycles of $UB(n)$ are 4-cycles and 8-cycles, and almost all of the short cycles of $SE(n)$ are 8-cycles. This can be seen by examining the homomorphisms of Proposition 2.3. Under homomorphism ϕ, the basic 4-cycles of $BF(n)$ map to basic 4-cycles (and the two basic 3-cycles) of $UB(n)$. Homomorphism ψ maps most 8-cycles of $CCC(n)$ to 8-cycles of $SE(n)$. (A few 8-cycles of $CCC(n)$ are mapped to 5-cycles of $SE(n)$.) Since most of the short cycles have length 4 or 8, the interesting spanners with small dilations are 3-spanners and 7-spanners. Consequently, we will concentrate on 3-spanners and 7-spanners in this section. The numbers of edges in spanners with other small dilations are presented in a table at the end of the section.

THEOREM 4.1. *For $n \geq 7$, a minimum 3-spanner of $UB(n)$ has exactly $3 \cdot 2^{n-1} - 3$ edges and is obtained by removing one edge from each basic 4- or 3-cycle.*

Proof. By Corollary 2.9, $UB(n)$ is the edge-disjoint union of basic 4-cycles and basic 3-cycles except for edge $[0101\cdots, 1010\cdots]$. For $n \geq 7$, there are only a few nonbasic cycles of length 4 or less. So, it is easy to check that at most one edge of any basic 4- or 3-cycle is also on a nonbasic cycle of length 4 or less. Thus, if more than one edge is deleted from any basic 4-cycle or basic 3-cycle, there will be two vertices at distance greater than 3 in the resulting spanner. So, to obtain a 3-spanner, at most 2^{n-1} edges can be removed from $UB(n)$. $UB(n)$ has $2^{n+1} - 3$ edges, so a minimum 3-spanner will have at least $3 \cdot 2^{n-1} - 3$ edges. The fact that there exists a 3-spanner of minimum size follows directly from Corollary 2.9. It is necessary and sufficient to remove one edge from each basic 4-cycle and each basic 3-cycle of $UB(n)$, being careful not to delete $[0101\cdots, 1010\cdots]$, the only edge which belongs to two basic 4-cycles. □

For $n < 7$ it is not difficult to show that a few more than 2^{n-1} edges can be deleted to get a minimum 3-spanner of $UB(n)$. The constructions that prove the following proposition are mechanical.

PROPOSITION 4.2. *A minimum 3-spanner of $UB(n)$ has $3 \cdot 2^{n-1} - 5$ edges for $n = 3$, 5, or 6, and $3 \cdot 2^{n-1} - 6$ edges for $n = 4$.*

THEOREM 4.3. *If $n \geq 7$ and n is even, then $SE(n)$ is a minimum 3-spanner of $UB(n)$. If $n \geq 7$ and n is odd, a minimum 3-spanner of $UB(n)$ can be obtained by deleting one edge from $SE(n)$. In both cases, the minimum 3-spanner has $3 \cdot 2^{n-1} - 3$ edges and maximum degree 3.*

Proof. Consider the inverse f^{-1} of mapping f from the proof of Proposition 2.2. f^{-1} induces a one-to-one mapping of shuffle edges of $UB(n)$ to shuffle edges of $SE(n)$. Therefore, we have only to determine the dilations due to f^{-1} of shuffle-exchange edges of $UB(n)$. Consider a shuffle-exchange edge $[x_0x_1 \cdots x_{n-1}, x_1 \cdots x_{n-1}\overline{x_0}]$ of $UB(n)$. If $x_0x_1 \cdots x_{n-1}$ has even parity, then the edge is mapped by f^{-1} to the exchange edge $[x_1x_2 \cdots x_{n-1}x_0, x_1x_2 \cdots x_{n-1}\overline{x_0}]$ in $SE(n)$ so the dilation is 1. If $x_0x_1 \cdots x_{n-1}$ has odd parity, then the endpoints of the edge are mapped to vertices $x_0x_1 \cdots x_{n-1}$ and $x_2x_3 \cdots x_{n-1}\overline{x_0}x_1$ in $SE(n)$, which are connected by the path $x_0x_1 \cdots x_{n-1}$, $x_1x_2 \cdots x_{n-1}x_0$, $x_1x_2 \cdots x_{n-1}\overline{x_0}$, $x_2 \cdots x_{n-1}\overline{x_0}x_1$ in $SE(n)$. So, $SE(n)$ is a 3-spanner of $UB(n)$ with maximum degree 3. Since $SE(n)$ has $3 \cdot 2^{n-1} - 3 + (n \bmod 2)$ edges it is a minimum 3-spanner if n is even for $n \geq 7$ by Theorem 4.1. If n is odd, $SE(n)$ contains a 4-cycle with vertices $(01)^{(n-1)/2}0$, $(10)^{(n-1)/2}0$, $(10)^{(n-1)/2}1$, and $(01)^{(n-1)/2}1$. It is not hard to show that deleting the exchange edge $[(10)^{(n-1)/2}0, (10)^{(n-1)/2}1]$ gives a 3-spanner of $SE(n)$ which is a minimum 3-spanner of $UB(n)$ with maximum degree 3. □

It is also possible to obtain 3-spanners of $UB(n)$ directly from some of the 3-spanners of $BF(n)$. It follows from Proposition 2.7 that every 3-spanner of $UB(n)$ can be obtained from $BF(n)$ by deleting exactly one edge from each basic 4-cycle between two adjacent levels of $BF(n)$ and every other edge of $BF(n)$ that has the same image under ψ as a deleted edge, being careful not to delete any edge that ψ maps to $[0101\cdots, 1010\cdots]$.

We will conclude our discussion of minimum 3-spanners of $UB(n)$ by briefly describing an algorithm for constructing any minimum 3-spanner of $UB(n)$ with maximum degree 3. The algorithm is quite similar to the algorithm in §3 for constructing any 3-regular minimum 3-spanner of $BF(n)$. The basic idea is to delete exactly one edge from each of the 2^{n-1} basic 4- and 3-cycles of $UB(n)$ in such a way that at least one edge is deleted at each vertex of degree 4. The only basic 4-cycles with which we have to be careful are the two basic 4-cycles that have the edge $[0101\cdots, 1010\cdots]$ in common. We start by deleting one edge adjacent to the common edge from one of these basic 4-cycles. This choice is important since the process fails if no edge adjacent to the common edge is deleted. This choice restricts the choices in the two 4- or 3-cycles of basic 4- or 3-cycles that contained the deleted edge. Subsequent edge deletions further reduce the remaining choices until all remaining edge deletions are forced. Every valid sequence of edge deletions gives a minimum 3-spanner with maximum degree 3. The two basic 3-cycles are treated in exactly the same way as the basic 4-cycles.

THEOREM 4.4. *For $n \geq 3$, there is a 7-spanner of $SE(n)$ with $\frac{5}{4} \cdot 2^n - 3 + (n \bmod 2)$ edges.*

Proof. The homomorphism φ from Proposition 2.3 maps a vertex $(i, x_0x_1 \cdots x_{n-1})$ of $CCC(n)$ onto the vertex $x_{i+1}x_{i+2} \cdots x_{i-1}x_i$ of $SE(n)$. Thus, ϕ maps C-edges of $CCC(n)$ onto shuffle edges of $SE(n)$ and H-edges of $CCC(n)$ onto exchange edges of $SE(n)$. (Note that the shuffle edges to which ϕ maps C-edges of the form $[(i, 0^n), (i+1, 0^n)]$ and $[(i, 1^n), (i+1, 1^n)]$ are loops at 0^n and 1^n, respectively.) It follows that

each fundamental n-cycle of $CCC(n)$ is mapped onto a cycle of $SE(n)$ that consists entirely of shuffle edges. We will call such a cycle in $SE(n)$ a *shuffle cycle.* The length of a shuffle cycle of $SE(n)$ is either n or a factor of n. If a fundamental n-cycle of $CCC(n)$ is mapped by ϕ onto a shuffle cycle of length n, then the mapping of the edges is one-to-one. If the shuffle cycle has length k where $0 < k < n$, then ϕ maps $\frac{n}{k}$ C-edges to each shuffle edge. Each fundamental cycle of $CCC(n)$ corresponds to a hypercube vertex $x_0x_1 \cdots x_{n-1}$. The fundamental cycles corresponding to $x_0x_1 \cdots x_{n-1}$, $x_1x_2 \cdots x_{n-1}x_0$, $\cdots$, $x_{n-1}x_0 \cdots x_{n-2}$, that is, all cyclic permutations of $x_0x_1 \cdots x_{n-1}$, will map to the same shuffle cycle in $SE(n)$. Most of the nonfundamental 8-cycles of $CCC(n)$ (which are alternating sequences of C-edges and H-edges) are mapped by ϕ to 8-cycles of $SE(n)$, which are alternating sequences of shuffle and exchange edges. ϕ maps n nonfundamental 8-cycles of $CCC(n)$ to each such 8-cycle of $SE(n)$. The exceptions are the nonfundamental 8-cycles of $SE(n)$ that contain a vertex $(i, 0^n)$ or $(i, 1^n)$. The n 8-cycles containing a vertex $(i, 0^n)$ are mapped onto the degenerate 8-cycle of $SE(n)$ consisting of the exchange edge $[0^n, 0^{n-1}1]$ and the 5-cycle $0^{n-1}1, 0^{n-2}10, 0^{n-2}11, 10^{n-2}1, 10^{n-1}, 0^{n-1}1$. The mapping for the n 8-cycles containing a vertex $(i, 1^n)$ is symmetric. The construction from Theorem 3.6 for 7-spanners of $CCC(n)$ can be used with mapping ϕ to construct 7-spanners of $SE(n)$ if the edges deleted from $CCC(n)$ are chosen carefully. The construction removes exactly one C-edge from each nonfundamental 8-cycle of $CCC(n)$. Since ϕ maps n nonfundamental 8-cycles of $CCC(n)$ to each 8-cycle (or degenerate 8-cycle) X of $SE(n)$, the n edges deleted from the nonfundamental cycles mapping to X should all map to the same edge of X. Also, in the degenerate 8-cycles, the edges $[0^n, 0^{n-1}1]$ and $[1^n, 1^{n-1}0]$ must not be deleted because deleting them would disconnect the graph. Since there are $\frac{n}{4} \cdot 2^n$ nonfundamental 8-cycles in $CCC(n)$, this spanner of $SE(n)$ is obtained by deleting $\frac{1}{4} \cdot 2^n$ edges, leaving $\frac{5}{4} \cdot 2^n - 3 + (n \bmod 2)$ edges. □

The spanners of $SE(n)$ from Theorem 4.4 are not minimum 7-spanners because some short cycles will remain intact. However, there is only a constant number of cycles of length less than 8 in $SE(n)$, and the number of 8-cycles of $SE(n)$ that are not images under ϕ of 8-cycles of $CCC(n)$ is also a constant. So, minimum 7-spanners of $SE(n)$ have $\frac{5}{4} \cdot 2^n - O(1)$ edges.

The following theorem uses the mapping ψ from Proposition 2.3 to obtain 7-spanners of $UB(n)$ from 7-spanners of $BF(n)$ in the same way that ϕ was used in Theorem 4.4. These 7-spanners of $UB(n)$ are not minimum spanners either, but like the 7-spanners of $SE(n)$, they are within a constant number of edges of minimum.

THEOREM 4.5. *For $n \geq 3$, there is a 7-spanner of $UB(n)$ with $\frac{5}{4} \cdot 2^n - 3$ edges.*

Proof. The homomorphism ψ from Proposition 2.3, which maps vertices of $BF(n)$ to vertices of $UB(n)$, acts in a very similar way to ϕ in the proof of Theorem 4.4. C-edges of $BF(n)$ are mapped to shuffle edges of $UB(n)$ (except the C-edges $[(i, 0^n), (i+1, 0^n)]$ and $[(i, 1^n), (i+1, 1^n)]$), H-edges of $BF(n)$ are mapped to shuffle-exchange edges of $UB(n)$, and the n fundamental n-cycles of $BF(n)$ that correspond to hypercube vertices whose bit strings are cyclic permutations of each other map to the same shuffle cycle of $UB(n)$. It follows from Proposition 2.7 and Corollary 2.8 that ψ maps n of the basic 4-cycles of $BF(n)$ to each basic 4- or 3-cycle of $UB(n)$. ψ also maps n nonfundamental 8-cycles of $BF(n)$ to each 8-cycle of $UB(n)$ that is an alternating sequence of shuffle and shuffle-exchange edges. n nonfundamental 8-cycles of $BF(n)$ are also mapped to each of the two degenerate shuffle cycles of $UB(n)$ (which contain vertex 0^n or 1^n). The construction from Theorem 3.7 for 7-spanners of $BF(n)$ can be used with mapping ψ to construct 7-spanners of $UB(n)$ as follows. The construction

of Theorem 3.7 first removes one H-edge from each basic 4-cycle of $BF(n)$ and then one C-edge from the nonfundamental 8-cycle remaining in each 4-cycle of basic 4-cycles. The deleted edges should be chosen so that when ψ maps n 8-cycles of $BF(n)$ to the same 8-cycle (or degenerate 8-cycle) X of $UB(n)$, or n basic 4-cycles of $BF(n)$ to the same basic 4- or 3-cycle X of $UB(n)$, the n deleted edges all map to the same edge of X. Care should also be taken not to disconnect vertex 0^n or vertex 1^n from the rest of the spanner. If a 7-spanner of $BF(n)$ satisfies these constraints, then its image under ψ is a 7-spanner of $UB(n)$. □

We can also use the mappings of Propositions 2.2 and 2.3 to obtain 7-spanners of $UB(n)$ directly from some 7-spanners of $CCC(n)$. Any 7-spanner of $CCC(n)$ that is also a 7-spanner of $BF(n)$ and that is mapped by ϕ onto a 7-spanner of $SE(n)$ will map to a 7-spanner of $UB(n)$ by either of the mappings $\psi \circ g$ or $f \circ \phi$.

Using the same arguments as were used for $CCC(n)$ and $BF(n)$, it can be shown that some 7-spanners of $SE(n)$ are 7-spanners of $UB(n)$ and others are not. In particular, if a 7-spanner of $CCC(n)$ is not a 7-spanner of $BF(n)$, then its image under $\psi \circ g$ or $f \circ \phi$ might not be a 7-spanner of $UB(n)$ even if its image under ϕ is a 7-spanner of $SE(n)$. Figures 6 and 7 show one example of each type. In Figs. 6 and 7, there are two sets of vertex labels. The labels along the top and left side of each figure are for $CCC(3)$ and $BF(3)$ and are the same as in Fig. 2. The labels next to the vertices in the interiors of the graphs are the labels of the vertices of $SE(n)$ and $UB(n)$ onto which the vertices of $CCC(n)$ and $BF(n)$ are mapped by ϕ and ψ, respectively.

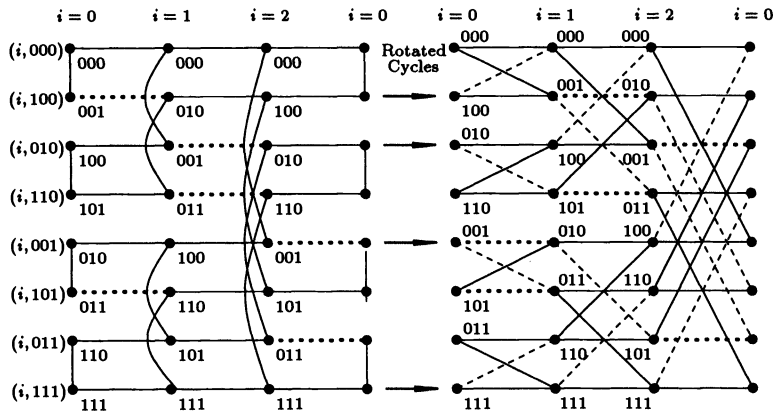


FIG. 6. *A 7-spanner S of $CCC(n)$ for which $\phi(S)$ is a 7-spanner of $SE(n)$, but $g(S)$ is not a 7-spanner of $BF(n)$ and $f \circ \phi(S) = \psi \circ g(S)$ is not a 7-spanner of $UB(n)$.*

Table 3 summarizes results about t-spanners of $SE(n)$ and $UB(n)$ for small t and $n \geq 8$. The first line shows the numbers of edges in $SE(n)$ and $UB(n)$ since dilation 1 does not permit the removal of any edges. The results for 3-spanners and 7-spanners were proved in this section. The proofs for the other results in the table are omitted. There are very few short cycles of lengths other than 4 and 8 in $SE(n)$ and $UB(n)$, so the omitted proofs involve careful analyses of the cycle structures of the graphs but no new construction or proof techniques. Results about the numbers of edges in minimum 7-spanners for $UB(n)$ and $SE(n)$ have not been proved in this paper. All of the other results for $UB(n)$ in the table are for minimum spanners. However, none of the results reported in the table for $SE(n)$ are minimum for all n (except the trivial case of dilation 1) because the formulae for the numbers of edges in small dilation spanners for $SE(n)$ are functions of both the dilation and n. Rather than list all of the

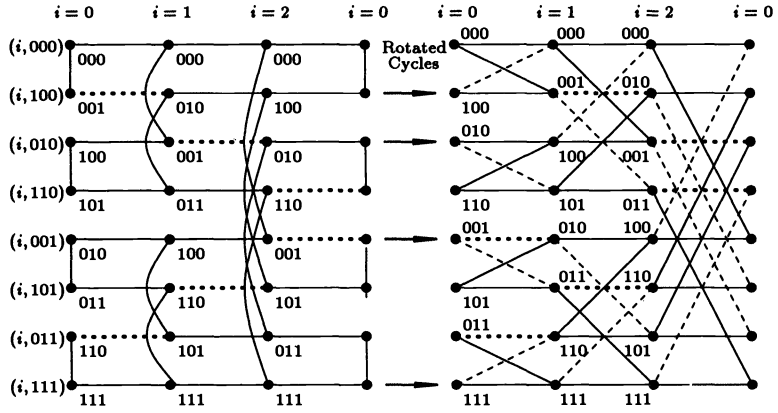


FIG. 7. *A 7-spanner S of $CCC(n)$ for which $\phi(S)$ is a 7-spanner of $SE(n)$, $g(S)$ is a 7-spanner of $BF(n)$, and $f \circ \phi(S) = \psi \circ g(S)$ is a 7-spanner of $UB(n)$.*

cases, we have reported small values that are correct for all n. Also note that there is a dilation 2 embedding of $UB(n)$ into $SE(n)$ [16], but it does not give a 2-spanner of $UB(n)$ because $SE(n)$ is not a subgraph of $UB(n)$ in this embedding.

TABLE 3
Numbers of edges in some small dilation spanners of $SE(n)$ and $UB(n)$ for $n \geq 8$.

Dilation	$SE(n)$	$UB(n)$
1	$\frac{3}{2} \cdot 2^n - 3 + (n \bmod 2)$	$2 \cdot 2^n - 3$
2	$\frac{3}{2} \cdot 2^n - 3 + (n \bmod 2)$	$2 \cdot 2^n - 7$
3	$\frac{3}{2} \cdot 2^n - 3$	$\frac{3}{2} \cdot 2^n - 3$
4	$\frac{3}{2} \cdot 2^n - 7$	$\frac{3}{2} \cdot 2^n - 7$
5	$\frac{3}{2} \cdot 2^n - 12$	$\frac{3}{2} \cdot 2^n - 12$
6	$\frac{3}{2} \cdot 2^n - 20$	$\frac{3}{2} \cdot 2^n - 20$
7	$\frac{5}{4} \cdot 2^n - 3 + (n \bmod 2)$	$\frac{5}{4} \cdot 2^n - 3$

5. Conclusions. In this paper we have concentrated on 3- and 7-spanners. The next interesting dilation is 11 since the next small cycle length in $BF(n)$ and $CCC(n)$ is 12. Results for dilation 11 are apparently much more difficult to obtain than those for dilations 3 and 7, and we believe that new proof techniques will be necessary to find minimum 11-spanners. We also believe that the trade-off between the dilation t and the number of edges that can be deleted in the classes of graphs we are studying is a well-behaved function of t and n, but without tight bounds for 11-spanners, we were not able to make this trade-off precise.

We were able to construct t-spanners of maximum degree 3 for t as small as 3. It is clearly not possible to lower the maximum degree to 2 and keep t small since any spanner with maximum degree 2 is a Hamilton cycle or path. It is known that $CCC(n)$, $UB(n)$, and $BF(n)$ are Hamiltonian and it has recently been shown [9] that $SE(n)$ has a Hamilton path. An interesting problem is to determine, for each of the four classes of graphs that we have considered, the minimum value of t for which the graph has a t-spanner of maximum degree 2. This problem can be solved by finding the Hamilton cycle or path for which the longest chord has minimum length.

REFERENCES

[1] F. ANNEXSTEIN, M. BAUMSLAG, AND A. ROSENBERG, *Group action graphs and parallel architectures*, SIAM J. Comput., 19 (1990), pp. 544–569.

[2] B. AWERBUCH, *Complexity of network synchronization*, J. Assoc. Comput. Mach., 32 (1985), pp. 804–823.

[3] B. AWERBUCH, A. BARATZ, AND D. PELEG, *Efficient broadcast and light-weight spanners*, manuscript, 1992.

[4] H. BANDELT AND A. DRESS, *Reconstruction of the shape of a tree from observed dissimilarity data*, Adv. in Appl. Math., 7 (1986), pp. 309–343.

[5] L. CAI, *Tree 2-spanners*, Tech. report CMPT TR 91-4, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, 1991.

[6] L. CAI AND M. KEIL, *Spanners in graphs with bounded degree*, Res. Report 93-1, Department of Computational Science, University of Saskatchewan, Saskatchewan, Canada, 1993.

[7] L. 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.

[8] D. DOBKIN, S. FRIEDMAN, AND K. SUPOWIT, *Delaunay graphs are almost as good as complete graphs*, Discrete Comput. Geom., 5 (1990), pp. 399–407.

[9] R. FELDMANN AND P. MYSLIWIETZ, *The shuffle exchange network has a Hamiltonian path*, in Proc. Mathematical Foundations of Computer Science 92, Lecture Notes in Computer Science 629, Springer-Verlag, Berlin, 1992, pp. 246–254.

[10] R. FELDMANN AND W. UNGER, *The cube-connected cycles network is a subgraph of the butterfly network*, Parallel Process. Lett., 2 (1992), pp. 13–19.

[11] M.-C. HEYDEMANN AND D. SOTTEAU, *A note on recursive properties of de Bruijn, Kautz and FFT digraphs*, Inform. Process. Lett., 53 (1995), pp. 255–259.

[12] F. T. LEIGHTON, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, San Mateo, 1992.

[13] A. LIESTMAN AND T. SHERMER, *Additive spanners for hypercubes*, Parallel Process. Lett., 1 (1991), pp. 35–42.

[14] ———, *Grid spanners*, Networks, 23 (1993), pp. 123–133.

[15] D. MELIKSETIAN AND C. Y. CHEN, *Optimal routing algorithm and the diameter of the cube-connected cycles*, IEEE Trans. Parallel Distrib. Systems, 4 (1993), pp. 1172–1178.

[16] B. MONIEN AND H. SUDBOROUGH, *Embedding one interconnection network in another*, Comput. Suppl., 7 (1990), pp. 257–282.

[17] D. PELEG AND A. SCHÄFFER, *Graph spanners*, J. Graph Theory, 13 (1989), pp. 99–116.

[18] D. PELEG AND J. ULLMAN, *An optimal synchronizer for the hypercube*, in Proc. 6th ACM Symposium on Principles of Distributed Computing, Vancouver, BC, 1987, pp. 77–85.

[19] D. PELEG AND E. UPFAL, *A tradeoff between space and efficiency for routing tables*, in Proc. 20th ACM Symposium on Theory of Computing, Chicago, IL, 1988, pp. 43–52.

[20] A. ROSENBERG, *Cycles in networks*, Tech. report UM-CS-1991-020, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1991.

[21] D. RICHARDS AND A. LIESTMAN, *Degree-constrained pyramid spanners*, J. Parallel Distrib. Comput., 25 (1995), pp. 1–6.

[22] R. STRONG, *On Hamiltonian cycles in Cayley graphs of wreath products*, Discrete Math., 65 (1987), pp. 75–80.

[23] P. VAIDYA, *A sparse graph almost as good as the complete graph on points in k dimensions*, Discrete Comput. Geom., 6 (1991), pp. 369–381.

[24] M. YOELI, *Binary ring sequences*, Amer. Math. Monthly, 69 (1962), pp. 852–855.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 55–62, February 1996

006

LOWER BOUNDS ON REPRESENTING BOOLEAN FUNCTIONS AS POLYNOMIALS IN Z_m*

SHI-CHUN TSAI†

Abstract. Define the MOD_m-degree of a boolean function F to be the smallest degree of any polynomial P, over the ring of integers modulo m, such that for all 0-1 assignments $\vec{x}$, $F(\vec{x}) = 0$ iff $P(\vec{x}) = 0$. By exploring the periodic property of the binomial coefficients modulo m, two new lower bounds on the MOD_m-degree of the MOD_l and $\neg MOD_m$ functions are proved, where m is any composite integer and l has a prime factor not dividing m. Both bounds improve from sublinear to $\Omega(n)$. With the periodic property, a simple proof of a lower bound on the MOD_m-degree with symmetric multilinear polynomial of the OR function is given. It is also proved that the majority function has a lower bound $\frac{n}{2}$ and the MidBit function has a lower bound $\sqrt{n}$.

Key words. boolean function complexity, circuit complexity, computational complexity

AMS subject classifications. 68Q05, 68Q15, 68Q25, 94C10

1. Introduction.

Proving lower bounds on explicitly given boolean functions is a notoriously difficult task. While the problem of proving nonlinear sized lower bounds for general boolean functions is still not in sight, very powerful results are available for restricted classes, such as monotone circuits or circuits of unbounded fan-in gates with small depth [5], [17]. One of the major tools in obtaining these lower bounds was the use of algebraic techniques [11], [12], [13]. As a result, several researchers started the study of polynomials that in some sense represent boolean functions. In particular, one wishes to find lower bounds on the degree of such polynomials: in some situations this enables us to prove a lower bound on circuit size [12]. In any case, the degree of the representing polynomial is a natural characteristic of the boolean function, and it is interesting to try to find bounds on it.

In this paper we prove lower bounds on the MOD_m-degree of some functions for composite m. These bounds substantially improve previous results [2], use elementary methods, and we hope shed some more light on the power of the MOD_m function for composite m. Recall that the Smolensky–Razborov technique does not seem applicable in this situation, and Szegedy's dissertation [13] also shows that such gates are surprisingly powerful. Thus any lower bound for these gates is of interest.

We briefly review previous main results. There are many possible representations of boolean functions as polynomials over some ring or field. Nisan and Szegedy [9] studied the degree of real polynomials that represent boolean functions. Paturi [10] studied the degree of real polynomials that approximate symmetric boolean functions. In [14], [15] Tarui proved some results on the degree complexity of boolean functions over rings and applied these results to prove the separations of certain relativized classes. In [2] Barrington, Beigel, and Rudich studied representing boolean functions as polynomials modulo m. Our work is an extension of the results in [2].

The techniques of Razborov [11] and Smolensky [12] that show polynomial sized constant depth circuits with MOD_p gates cannot compute the MOD_q function (the boolean function $MOD_m(x_1, \ldots, x_n)$ is defined to be 0 if $\sum_{i=0}^{n} x_i$ is a multiple of m and 1 otherwise) when p and q are distinct primes suggest the following definition

* Received by the editors September 14, 1993; accepted for publication (in revised form) January 10, 1995. A preliminary version of this paper appeared in *Proceedings of the 8th Annual Structure in Complexity Theory Conference*, San Diego, CA, 1993, pp. 96–101.

† Department of Computer Science, University of Chicago, 1100 East 58th Street, Chicago, IL 60637 (tsai@cs.uchicago.edu).

[2]. A polynomial P over Z_m represents a boolean function F if for all 0-1 valued assignments $\vec{x}$, $F(\vec{x}) = 0$ iff $P(\vec{x}) = 0$. The MOD_m-degree of F, denoted $\delta(F, m)$, is the degree of the lowest degree polynomial that represents it. This is an interesting complexity measure on its own and, in the case of prime moduli, yields lower bounds for circuit size.

For moduli m that are prime, or prime power, the situation is well understood [12], [3]. It is known that for the OR function of n variables $\delta(OR, p) = \lceil \frac{n}{p-1} \rceil$ [12], [16]. It is also known that $\delta(MOD_l, p) = \Omega(n)$ when l is not a power of p. These results depend crucially on the fact that Z_p is a finite field.

For composite moduli, when Z_m is a ring and not a field, many of the lemmas simply fall apart. Barrington, Beigel, and Rudich [2] proved the first few lower bounds for the OR and MOD functions for composite m. They proved that if the representing polynomial is symmetric, then $\delta(OR, m) = \Theta(n^{1/r})$, where r is the number of distinct prime factors of m. They also proved that for any nonsquarefree composite integer m, $\delta(MOD_l, m) = n^{\Omega(1)}$ and $\delta(\neg MOD_l, m) = n^{\Omega(1)}$, where l has a prime factor that is not a divisor of m and $\delta(\neg MOD_m, m) = n^{\Omega(1)}$. If m is squarefree they have $\delta(MOD_l, m) = n^{\Omega(1)}$, $\delta(\neg MOD_l, m) = \Omega(n)$, and $\delta(\neg MOD_m, m) = \Omega(n)$.

Our paper improves these results. First, we give a new simple proof for the lower bound for the OR function that does not depend on m being squarefree. We prove that the MOD_m-degree of the majority function is at least $\frac{n}{2}$. We also show a lower bound, $\sqrt{n}$, for the MidBit function. (The MidBit function, defined in [4], is the boolean function that takes $x_1, \ldots, x_n$ as input and outputs the value of the $\lfloor \frac{\log n}{2} \rfloor$th bit in the binary representation of the number $\sum_{i=1}^{n} x_i$.)

Finally, we prove that $\delta(MOD_l, m) = \Omega(n)$ and $\delta(\neg MOD_l, m) = \Omega(n)$ when l has a prime that is not a factor of m and that $\delta(-MOD_m, m) = \Omega(n)$. These improve the bounds in [2] for the case m is not squarefree. Our proofs rely on a periodic property of the binomial coefficients. They are simpler and more straightforward than the previous proofs of weaker statements.

2. Notation. As a convention, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. Let A and B be sets. We write $A \subseteq B$ if every member of A also belongs to B, i.e., A is a subset of B. We write $A \subset B$ to indicate that A is a proper subset of B. Boolean functions on n variables will be represented as polynomials in the ring $Z_m[x_1, \ldots, x_n]$. Since we will consider the domain $\{0,1\}^n$ only, we are interested in the multilinear polynomials. For $A \subseteq [n]$, we define $X_A = \prod_{i \in A} x_i$. It is clear that $\{X_A | A \subseteq [n]\}$ forms a basis for the multilinear polynomials in $Z_m[x_1, \ldots, x_n]$. And $\{X_A | A \subseteq [n], |A| \leq d\}$ forms a basis for the multilinear polynomials of degree at most d in $Z_m[x_1, \ldots, x_n]$. Let $P(x_1, \ldots, x_n)$ be any n-variable function with domain $\{0,1\}^n$. We say that $P(A)$ has the value obtained with the assignment $x_i = 1$ if $i \in A$ and 0 otherwise.

DEFINITION 2.1. *Define $\delta(F, m)$, the MOD_m-degree of a boolean function F, to be the smallest degree of any polynomial P in Z_m such that for all* 0-1 *assignments $\vec{x}$, $F(\vec{x}) = 0$ iff $P(\vec{x}) = 0$.*

DEFINITION 2.2. *Let m, r, and n be positive integers, where $0 \leq r \leq m-1$. The $MOD_{(m,r)}$ function of n variables is defined as*

$$MOD_{(m,r)}(x_1, \ldots, x_n) = \begin{cases} 0 & \text{if } \sum_i x_i \equiv r \pmod{m}, \\ 1 & \text{otherwise.} \end{cases}$$

We use MOD_m to indicate $MOD_{(m,0)}$.

DEFINITION 2.3. *Let k and n be any positive integers and $k \leq n$. The threshold function TH_k^n of n variables is defined as*

$$TH_k^n(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_i x_i \geq k, \\ 0 & \text{otherwise.} \end{cases}$$

DEFINITION 2.4. *Let n be a positive integer. The MidBit function of n variables is defined as $MidBit(x_1, \ldots, x_n) = \lfloor \frac{\log n}{2} \rfloor$th bit in the binary representation of $\sum_{i=1}^n x_i$.*

3. Main lemmas. Let j be a nonnegative integer, and let k and m be any positive integers. Fix k and m; then the integer function $f(j) = \binom{j}{k} \bmod m$ is a cyclic function of j. As H. Kwong pointed out, this property has been proved by several other authors by using different methods [6], [7], [18]. For completeness we show the proof in the following.

LEMMA 3.1. *Let p be any prime number, and let a and k be any positive integers. Then $\binom{j}{k} \bmod p^a$ has the cycle length p^{a+e}, where $e = \lfloor \log_p k \rfloor$.*

Proof. Define $L_k = p^{a+\lfloor \log_p k \rfloor}$. We want to prove $\binom{sL_k+j}{k} \equiv \binom{j}{k} \pmod{p^a}$ for any nonnegative integer s and any positive integer k by induction on j. For the base case $j = 0$, since $\binom{sL_k}{k} = \frac{sL_k}{sL_k - k}\binom{sL_k - 1}{k}$ is an integer and the largest power of p that divides $sL_k - k$ is at most $p^{\lfloor \log_p k \rfloor}$, we know $\binom{sL_k}{k}$ is divisible by p^a. Thus $\binom{sL_k}{k} \equiv 0 \pmod{p^a}$ for all nonnegative integers s and any positive integer k. Suppose $\binom{sL_k+j}{k} \equiv \binom{j}{k} \pmod{p^a}$ for any nonnegative integer s and any positive integer k, for $0 \leq j \leq i$. Now consider the case when $j = i + 1$:

$$\begin{aligned} \binom{sL_k + i + 1}{k} &= \binom{sL_k + i}{k} + \binom{sL_k + i}{k - 1} \\ &\equiv \binom{i}{k} + \binom{i}{k-1} \pmod{p^a} \quad \text{because } L_{k-1} | L_k \\ &\equiv \binom{i+1}{k} \pmod{p^a}. \end{aligned}$$

Next we need to prove that L_k is the smallest cycle. To show L_k is the smallest cycle of $\binom{j}{k} \bmod p^a$, we prove that there exists an i, $0 \leq i < k$, such that $\binom{\frac{L_k}{p}+i}{k} \not\equiv \binom{i}{k} \equiv 0 \pmod{p^a}$. Without loss of generality, let $k = p^l + i$, where l is any positive integer and $0 \leq i < p^{l+1} - p^l$. With this assumption we have $\lfloor \log_p k \rfloor = l$. By induction on i, we will prove $p^a \nmid \binom{p^{l+a-1}+i}{p^l+i}$. This is clear for the base case $i = 0$. Suppose it holds for the case up to i, i.e., $p^a \nmid \binom{p^{l+a-1}+m}{p^l+m}$, where $0 \leq m \leq i \leq p^{l+1} - p^l - 1$. Consider the case $m = i + 1$. We are interested in the case when $i + 1 < p^{l+1} - p^l$. Then $\binom{p^{l+a-1}+i+1}{p^l+i+1} = \frac{p^{l+a-1}+i+1}{p^l+i+1}\binom{p^{l+a-1}+i}{p^l+i}$. We know $\frac{p^{l+a-1}+i+1}{p^l+i+1}$ is not divisible by p. By the induction hypothesis we have $p^a \nmid \binom{p^{l+a-1}+i+1}{p^l+i+1}$. This completes the proof. □

By applying the Chinese remainder theorem we can generalize the above lemma as follows.

LEMMA 3.2. *Let m_1 and m_2 be any two relatively prime positive integers. If $\binom{j}{k} \bmod m_1$ and $\binom{j}{k} \bmod m_2$ has the cycle length l_1 and l_2, respectively, then $\binom{j}{k} \bmod m_1 m_2$ has the cycle length $l_1 l_2$.*

THEOREM 3.3. *Let $m = p_1^{a_1} \dots p_r^{a_r}$ be an arbitrary positive integer, where the p_i's are distinct prime numbers and the a_i's are positive integers. Let $e_i = \lfloor \log_{p_i} k \rfloor$ for $1 \le i \le r$; then $\binom{j}{k} \bmod m$ has the cycle length $\prod_{i=1}^{r} p_i^{a_i+e_i}$.*

Proof. The theorem is proved by Lemmas 3.1 and 3.2. □

Let $P(x_1, \dots, x_n) \in Z_m[x_1, \dots, x_n]$ be a polynomial representing a Boolean function $f(x_1, \dots, x_n)$. If the degree of P is d, then we can write $P(x_1, \dots, x_n) = \sum_{A \subseteq [n]; |A| \le d} c_A X_A$. Recall that we define $f(A) = f(a_1, \dots, a_n)$, where $a_i = 1$ if $i \in A$ and 0 otherwise. $P(A)$ is defined similarly. For convenience, for each $A \subseteq [n]$ we use b_A to denote $P(A)$. Since P represents f, it is clear that if $f(A) = 0$, then $b_A = 0$ and if $f(A) = 1$, then b_A can be any nonzero integer in Z_m. If there are M different assignments in $\{0,1\}^n$ satisfying f, then by a simple counting argument there are $(m-1)^M$ multilinear polynomials in Z_m representing f. For any $A \subseteq [n]$ it is clear that $X_D(A) = 0$ if $D \not\subseteq A$ and $X_D(A) = 1$ if $D \subseteq A$. Thus $b_A = P(A) = \sum_{D \subseteq A; |D| \le d} c_D$. There are 2^n possible inputs. With the given boolean function we can choose a set of b_A's such that $b_A = 0$ if $f(A) = 0$ and $b_A \neq 0$ if $f(A) = 1$. Then we can set up a system of linear equations with $\sum_{i=0}^{d} \binom{n}{i}$ variables if the representing polynomial has degree d. By solving the linear equations, we prove some useful relations among b_D's and c_A's over any ring Z_m. Actually, Lemma 3.4.1 is a special case of the well-known Möbius inversion theorem [8].

LEMMA 3.4. 1. *If $A \subset [n]$ and $|A| \le d$, then*

$$c_A = \sum_{D \subseteq A} (-1)^{|A|-|D|} b_D.$$

2. *If $A \subseteq [n]$ and $|A| > d$, then*

$$b_A = \sum_{D \subset A; |D| \le d} (-1)^{d-|D|} \binom{|A| - |D| - 1}{d - |D|} b_D.$$

Proof. 1. The first part of the lemma is proved by induction on the size of A. It is clear for the cases of $|A| = 0$ and 1. Suppose it is true for all $|A| \le k$. Consider the case of $d \ge |A| = k + 1$. Then

$$\begin{aligned} P(A) &= b_A \\ &= c_A + \sum_{S \subset A} c_S \\ &= c_A + \sum_{S \subset A} \left(b_S + \sum_{D \subset S} (-1)^{|S|-|D|} b_D \right). \end{aligned}$$

The third equality is by induction hypothesis. Then

$$c_A = b_A - \sum_{S \subset A} \left(b_S + \sum_{D \subset S} (-1)^{|S|-|D|} b_D \right).$$

Let D_l be any subset of A with size $l < |A|$. We want to determine the common coefficient of b_{D_l}'s for a fixed l. It can be found as

$$- \left(\sum_{D_l \subset A} b_{D_l} + \sum_{S \subset A} \sum_{D_l \subset S} (-1)^{|S|-l} b_{D_l} \right)$$

$$
\begin{aligned}
&= -\left(\sum_{D_l \subset A} b_{D_l} + \sum_{D_l \subset A} \sum_{i=1}^{|A|-l-1} (-1)^i \binom{|A|-l}{i} b_{D_l}\right) \\
&= -\sum_{D_l \subset A} \sum_{i=0}^{|A|-l-1} (-1)^i \binom{|A|-l}{i} b_{D_l} \\
&= -\sum_{D_l \subset A} (-1)^{|A|-l-1} b_{D_l} \\
&= \sum_{D_l \subset A} (-1)^{|A|-l} b_{D_l}.
\end{aligned}
$$

This proves the first claim.

2.

$$
\begin{aligned}
b_A &= \sum_{D \subset A; |D| \le d} c_D \\
&= \sum_{D \subset A; |D| \le d} \left(\sum_{T \subseteq D} (-1)^{|D|-|T|} b_T\right) \\
&= \sum_{T \subset A; |T| \le d} \sum_{i=0}^{d-|T|} (-1)^i \binom{|A|-|T|}{i} b_T \\
&= \sum_{T \subset A; |T| \le d} (-1)^{d-|T|} \binom{|A|-|T|-1}{d-|T|} b_T \\
&= \sum_{D \subset A; |D| \le d} (-1)^{d-|D|} \binom{|A|-|D|-1}{d-|D|} b_D.
\end{aligned}
$$

In the second-to-last equation we use the fact $\sum_{i=0}^{l} (-1)^i \binom{m}{i} = (-1)^l \binom{m-1}{l}$. □

From the above lemma we know the first $(\sum_{i=0}^{d} \binom{n}{i})$ b_A's will determine the coefficients of the polynomial P and the rest of the b_A's. With the above lemmas we can prove several lower bounds systematically. The key of the proof relies on the periodic property of the binomial coefficients modulo m.

4. Lower bounds. From Lemma 3.4 we have an immediate lower bound for the threshold function.

THEOREM 4.1. *$\delta(TH_k^n, m) \ge k$, where m is any positive integer.*

Proof. Let P be a polynomial over Z_m, which represents TH_k^n and has degree less than k. By the definition of the threshold function, we know $b_A = 0$ for all $A \subset [n]$ with $|A| < k$, and by Lemma 3.4.1 we have $c_A = 0$ for all $A \in [n]$ with $|A| < k$. By Lemma 3.4.2 $b_A = 0$ for all $A \subseteq [n]$ with $|A| \ge k$, which is a contradiction to the definition of the threshold function. Thus the degree must be at least k. □

Let's look at the majority function. We define the majority function on n variables as $MAJ(x_1, \ldots, x_n) = 1$ if there are at least $\frac{n}{2}$ 1's in the input and 0 otherwise. Clearly, it is a special case of the threshold function. Thus we have the following immediate corollaries.

COROLLARY 4.2. *Let m be any positive integer.*

1. $\delta(MAJ, m) \ge \frac{n}{2}$.
2. $\delta(AND, m) = n$.

3. $\delta(MidBit, m) = \Omega(\sqrt{n})$.

It is clear that part 2 is trivial and part 3 follows from the proof of Theorem 4.1. Next we are going to prove the lower bound for the OR function. With the previous lemmas we prove it without distinguishing between the cases of squarefree and nonsquarefree m.

THEOREM 4.3. *Let m be any nonprime-power composite number. If the OR function of n variables is represented by a symmetric polynomial modulo m, then the degree is $\Omega(n^{1/r})$, where r is the number of distinct primes dividing m.*

Proof. Let P be a symmetric polynomial of degree d that represents the OR function over Z_m. Say $m = \prod_{i=1}^{r} p_i^{a_i}$. Suppose $p_i^{e_i} \leq d < p_i^{e_i+1}$ for $1 \leq i \leq r$. It is clear that $P(\emptyset) = b_\emptyset = 0$ and $P(A) = b_A \neq 0$ if $A \subseteq [n]$ and $A \neq \emptyset$. Since P is symmetric for any $A, B \subseteq [n]$, if $|A| = |B|$, then $b_A = b_B$ and so $c_A = c_B$. Thus we can use $c_{|A|}$ to indicate c_A and the others of the same cardinality. Consider the case $|A| > d$. It is easy to see $b_A = \sum_{i=1}^{d} \binom{|A|}{i} c_i$. By Theorem 3.3, we know $\binom{|A|}{d} \bmod m$ has the cycle length $\prod_{j=1}^{r} p_j^{e_j+a_j}$, when $|A|$ ranges over the nonnegative integers. We denote the cycle length as L. If $i > j$, then the cycle length of $\binom{|A|}{i} \bmod m$ is a multiple of the cycle length of $\binom{|A|}{j} \bmod m$. Thus $\binom{0+L}{i} \equiv 0 \pmod{m}$, for $1 \leq i \leq d$, since $\binom{0}{i} \equiv 0 \pmod{m}$. We must have $n < L$, otherwise $b_A = 0$ for $|A| = L$, which contradicts the definition of the OR function. Hence, $n < L = \prod_{j=1}^{r} p_j^{e_j+a_j} \leq md^r$, and therefore $d = \Omega(n^{1/r})$. □

It is still open to find an upper bound for the nonsymmetric case. A very weak lower bound can be obtained by a Ramsey argument. Meanwhile in [1], Alon et al. proved a better lower bound than the one that follows from the Ramsey argument. To resolve the problem, we can either try to improve the lower bound or find a small upper bound for the nonsymmetric representation of the OR function.

THEOREM 4.4. *Let m be a positive composite integer that is not a prime power; then $\delta(\neg MOD_m, m) = \Omega(n)$.*

Proof. Let P be a representing polynomial of the $-MOD_m$ function of degree d. Choose p^l such that $p^l | m$ and $p^l \nmid c_\emptyset$, where p is a prime factor of m. Such p and l exist, since $c_\emptyset \not\equiv 0 \pmod{m}$. It is clear that $b_\emptyset = c_\emptyset$. Without loss of generality, let $n = p^{k+l} + p^k - 1$, where k is an arbitrary integer. From Lemma 3.4.2, we have $b_A = \sum_{D \subset A; |D| \leq d} (-1)^{d-|D|} \binom{|A|-|D|-1}{d-|D|} b_D$, for $|A| > d$. We are interested in the sets $A \subset [n]$ with $|A| = p^{k+l}$ and the sum of the corresponding b_A's:

$$\begin{aligned}
\sum_{A \subset [n]} b_A &= \sum_{A \subset [n]} \sum_{D \subset A; |D| \leq d} (-1)^{d-|D|} \binom{|A|-|D|-1}{d-|D|} b_D \\
&= \sum_{D \subset [n]; |D| \leq d} (-1)^{d-|D|} \binom{|A|-|D|-1}{d-|D|} \binom{n-|D|}{|A|-|D|} b_D \\
&= \sum_{D \subset [n]; |D| \leq d} (-1)^{d-|D|} \binom{|A|-|D|-1}{d-|D|} \binom{n-|D|}{n-|A|} b_D.
\end{aligned}$$

It is clear that $n - |A| = p^k - 1$ and $b_A \equiv 0 \pmod{m}$ since $m \nmid |A|$ and by the definition of the $\neg MOD_m$ function. If $0 < |D| < p^k$, then $n - |D| = p^{k+l} + (p^k - |D| - 1)$. And so $\binom{n-|D|}{n-|A|} = \binom{n-|D|}{p^k-1} \equiv \binom{p^k-|D|-1}{p^k-1} \equiv 0 \pmod{p^l}$ since by Lemma 3.1, $\binom{j}{p^k-1} \bmod p^l$

has the cycle length p^{k+l-1}. Therefore, we have

$$\begin{aligned} 0 &\equiv (-1)^d \binom{|A|-1}{d} \binom{n}{n-|A|} b_\emptyset \pmod{p^l} \\ &\equiv (-1)^d \binom{|A|-1}{d} b_\emptyset \pmod{p^l}. \end{aligned}$$

If we let $d < p^k$, then $\binom{j}{d} \pmod{p^l}$ has the cycle length at most p^{k+l-1}. Thus,

$$\begin{aligned} 0 &\equiv (-1)^d \binom{|A|-1}{d} b_\emptyset \pmod{p^l} \\ &\equiv (-1)^d \binom{p^{l+\lfloor \log_p d \rfloor}-1}{d} b_\emptyset \pmod{p^l} \\ &\equiv b_\emptyset \sum_{i=0}^{d} (-1)^i \binom{p^{l+\lfloor \log_p d \rfloor}}{i} \pmod{p^l} \\ &\equiv b_\emptyset \pmod{p^l}. \end{aligned} \tag{1}$$

The modular equality in (1) follows from the identity $\sum_{i=0}^{n}(-1)^i\binom{m}{i} = (-1)^n\binom{m-1}{n}$. It leads to a contradiction. Therefore , $d \geq p^k = \frac{n+1}{p^l+1}$. The theorem is clear now for the case of $n = p^{k+l} + p^k - 1$. For the case $p^{k+l} + p^k - 1 < n < p^{k+1+l} + p^{k+1} - 1$, the MOD_m-degree must be at least p^k, which is at least $\frac{n+1}{p^{l+1}+p}$, and the proof is complete. □

By following the above proof we have an immediate corollary.

COROLLARY 4.5. *Let p be a prime number that is not a factor of m; then $\delta(\neg MOD_p, m) = \Omega(n)$.*

It is a fact that for any positive integer l, $\delta(MOD_l, m) = \delta(MOD_{(l,1)}, m) + c$, where c is a constant. Thus, following the lines in the proof of Theorem 4.4, we have the following theorem, which not only improves a previous sublinear lower bound in [2] but also extends it to any composite integer m.

THEOREM 4.6. *Let m be a positive composite integer and q be a prime number that is not a factor of m; then $\delta(MOD_q, m) = \Omega(n)$.*

Proof. Instead of proving the lower bound for the MOD_q function we prove it for the $MOD_{(q,1)}$ function. As usual let P be a representing polynomial for the $MOD_{(q,1)}$ function. Choose p^l as we did in Theorem 4.4. Without loss of generality, let $n = p^{(k+l)(q-1)} + p^{k(q-1)} - 1$, where k is an arbitrary integer. Then by following the proof in Theorem 4.4 and Fermat's little theorem we can prove the lower bound. □

Theorems 4.4 and 4.6 show that it is difficult to compute the functions MOD_p and $\neg MOD_p$ by polynomials over Z_m, when p is not a factor of m. By a simple counting argument, we know that almost every boolean function needs MOD_m-degree at least $\frac{n}{2} - o(\sqrt{n})$ for any positive integer m. It will be interesting to see whether $\frac{n}{2} - o(\sqrt{n})$ is enough for almost every boolean function when n is big enough.

Acknowledgments. I am grateful to Richard Beigel and Janos Simon for very useful comments on the earlier version of this paper. I would also like to thank Lance Fortnow, Katalin Friedl, and Mario Szegedy for helpful discussion; Harris Kwong for telling me about the papers on the periodic property of binomial coefficients modulo m; and last but not least the anonymous referees for their useful comments.

REFERENCES

[1] N. ALON, D. KLEITMAN, R. LIPTON, R. MESHULAM, M. RABIN, AND J. SPENCER, *Set systems with no union of cardinality 0 modulo m*, Graphs and Combinatorics, 7 (1991), pp. 97–99.

[2] D. A. BARRINGTON, R. BEIGEL, AND S. RUDICH, *Representing boolean functions as polynomials modulo composite numbers*, in Proc. 24th Annual ACM Symposium on Theory of Computing, Victoria, BC, Canada, 1992, pp. 455–461.

[3] R. BEIGEL AND J. TARUI, *On ACC*, in Proc. 32nd IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1991, pp. 783–792.

[4] F. GREEN, J. KÖBLER, AND J. TORÁN, *The power of the middle bit*, in Proc. 7th Annual IEEE Conference on Structure in Complexity Theory, Boston, MA, 1992, pp. 111–117.

[5] J. HÅSTAD, *Computational Limitations of Small-Depth Circuits*, MIT Press, Cambridge, MA, 1986.

[6] Y. H. KWONG, *Minimum periods of binomial coefficients modulo M*, Fibonacci Quart., 27 (1989), pp. 348–351.

[7] ———, *Periodicities of a class of infinite integer sequences modulo M*, J. Number Theory, 31 (1989), pp. 64–79.

[8] L. LOVÁSZ, *Combinatorial Problems and Exercises*, 2nd ed., North-Holland, Amsterdam, 1993.

[9] N. NISAN AND M. SZEGEDY, *On the degree of boolean functions as real polynomials*, in Proc. 24th Annual ACM Symposium on Theory of Computing, Victoria, BC, Canada, 1992, pp. 462–467.

[10] R. PATURI, *On the degree of polynomials that approximate symmetric boolean functions*, in Proc. 24th Annual ACM Symposium on Theory of Computing, Victoria, BC, Canada, 1992, pp. 467–474.

[11] A. A. RAZBOROV, *Lower bounds for the size of circuits of bounded depth with basis* $\{\wedge, \oplus\}$, Math. Notes of the Academy of Science of the USSR, 41-4 (1987), pp. 333–338.

[12] R. SMOLENSKY, *Algebraic methods in the theory of lower bounds for boolean circuit complexity*, in Proc. 19th Annual ACM Symposium on Theory of Computing, New York, NY, 1987, pp. 77–82.

[13] M. SZEGEDY, *Algebraic Methods in Lower Bounds for Computational Models with Limited Communication*, Ph.D. thesis, Department of Computer Science, University of Chicago, Chicago, IL, 1989.

[14] J. TARUI, *Randomized polynomials, threshold circuits, and the polynomial hierarchy*, in Proc. 8th Annual Symposium on Theoretical Aspects of Computer Science, Springer-Verlag, Berlin, New York, 1991, pp. 238–250.

[15] ———, *Degree complexity of boolean functions and its applications to relativized separations*, in Proc. 6th Annual IEEE Conference on Structure in Complexity Theory, Chicago, IL, 1991, pp. 382–390.

[16] S. C. TSAI, *A Simple Proof of Representing Boolean Functions as Polynomials Modulo p*, Technical report CS 92-27, Department of Computer Science, University of Chicago, Chicago, IL, 1992.

[17] A. C-C. YAO, *Separating the polynomial-time hierarchy by oracles*, in Proc. 26th IEEE Symposium on Foundations of Computer Science, Washington, D.C., 1985, pp. 1–10.

[18] S. ZABEK, *Sur la périodicité modulo m des suites de nombres* $\binom{n}{k}$, Ann. Univ. Mariae Curie-Sklodowska, A10 (1956), pp. 37–47.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 63–70, February 1996

007

ON CONVEX SUBSETS IN TOURNAMENTS*

DAVID J. HAGLIN† AND MARTY J. WOLF‡

Abstract. A tournament is a complete directed graph. A convex subset is a vertex subset with the property that every two-path beginning and ending inside the convex subset is contained completely within the subset. This paper shows that every nontrivial convex subset is the closure of a subset of vertices of cardinality two. This result leads to algorithms that find all convex subsets in a tournament in $O(n^4)$ serial time and in $O(\log^2 n)$ parallel time using $O(n^4)$ processors. Several variations of the problem that are solvable with this new algorithm are also presented.

Key words. tournament decomposition, convex subsets, $\mathcal{NC}$ algorithm

AMS subject classification. 05C20

1. Introduction.

A *tournament* is a directed graph on n vertices that is obtained by directing all of the edges in a complete undirected graph. A *convex subset* is a subset of the vertices such that any vertex not in the subset either dominates or is dominated by all of the vertices in the convex subset. A convex subset partitions the vertex set into three subsets: the convex subset, those vertices that dominate all of the vertices in the convex subset, and those vertices that are dominated by all of the vertices in the convex subset. This observation leads to a complete characterization of the convex subsets in a tournament, which leads to a serial algorithm that finds all convex subsets in any tournament within $O(n^4)$ time. Although it is known that a single convex subset can be found in $O(n^3)$ time [2], our algorithm enumerates all of them (potentially $\Theta(n^2)$ [5]) in $O(n^4)$ time. Furthermore, we give a work-efficient $\mathcal{NC}$ algorithm that takes $O(\log^2 n)$ time and $O(n^4)$ processors for finding all convex subsets in any tournament. Finally, our algorithm (in both the parallel and sequential versions) can be easily modified to provide solutions to variations of the convex subsets problem, including finding all of the convex subsets that contain a given subset of vertices and efficiently finding the smallest convex subset that contains a given subset with two or more vertices.

1.1. Background.

Varlet investigated many properties of convexity in tournaments and suggested that "it would be nice to have a simple algorithm giving all convex subsets of a tournament" [5, p. 574]. Determining a single convex subset in a tournament is essentially equivalent to finding a *split* in a directed graph (*digraph*) when the digraph is a tournament. This fact is alluded to in [1], although no proof is given in the literature. For completeness we have included a proof in the appendix. The graphs induced by a split are called a *simple decomposition* of the graph. Cunningham gives an $O(n^4)$ algorithm for determining whether a graph has a split and for computing a generalization of a simple decomposition called a *prime decomposition* [3]. When restricted to a tournament, a simple decomposition yields a convex subset if one exists. A prime decomposition of a tournament yields a list of graphs such that each graph directly corresponds to either a convex subset that does not properly contain another convex subset (i.e., a minimal convex subset) or a subtournament that

* Received by the editors June 29, 1993; accepted for publication (in revised form) January 11, 1995. This research was supported in part by a Mankato State University Academic Affairs Research Fund grant.

† Computer and Information Sciences Department, Mankato State University, Mankato, MN 56002 (haglin@mankato.msus.edu).

‡ Computer and Information Sciences Department, Mankato State University, Mankato, MN 56002 (mjwolf@mankato.msus.edu).

contains no (nontrivial) convex subsets. The prime decomposition of a tournament of size n has $O(n)$ graphs. Since there are tournaments with $\Omega(n^2)$ convex subsets [5], the prime decomposition will not immediately give rise to *all* convex subsets in a tournament. An obvious algorithm that extracts all convex subsets from a prime decomposition by repeatedly considering the union of known convex subsets with other known convex subsets and parts of the prime decomposition takes $O(n^5)$ time. Any approach that repeatedly checks candidate subsets for convexity will likely take $\Omega(n^4)$ time in the worst case since $\Omega(n^2)$ subsets might be considered, and each convexity check takes $\Omega(n^2)$ time in the worst case.

Bouchet gives an $O(n^3)$ time algorithm for finding a split of a digraph [2] (a convex subset in a tournament) if one exists. This algorithm shares the limitations of Cunningham's algorithm in generating all convex subsets of a tournament. Astie-Vidal and Matteo were the first to give an algorithm that enumerates all of the convex subsets of a tournament [1]. However, their algorithm applies only to *regular* tournaments (tournaments where the outdegree of each vertex is equal to its indegree). They claim that the number of "elementary operations" for their algorithm is $O(n^3)$. However, they include operations such as set intersection and set subtraction as elementary operations. In this paper, we view those operations (as well as other related operations) as taking $O(n)$ time. Their algorithm has an $O(n^4)$ time bound with this interpretation. Our algorithm with that same time bound computes all of the convex subsets of any tournament. Furthermore, we parallelize our algorithm to show that the problem of computing all of the convex subsets of a tournament is in $\mathcal{NC}$. The algorithm given by Astie-Vidal and Matteo does not have an obvious parallelization. Also note that Cunningham's algorithm is not obviously parallelizable since it requires vertices to be considered in a very special order.

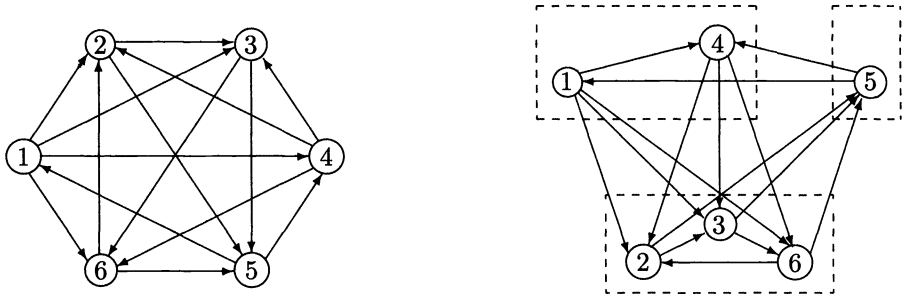
1.2. Definitions. Let $T = (V, E)$ be a tournament on n vertices. Assume that $V = \{1, 2, \ldots, n\}$. For vertices $v, w \in V$, we say that v dominates w (denoted $v \to w$) if the edge between v and w is directed from v to w. A *convex subset* in T is a set $C \subseteq V$ such that for any $v \in V - C$ either v dominates every vertex in C (denoted $v \Rightarrow C$) or every vertex in C dominates v (denoted $C \Rightarrow v$). This notion is illustrated in Fig. 1. Since every subset of V of size 0, 1, and n is convex, these convex subsets are called *trivial*. This paper deals only with finding the nontrivial convex subsets and, henceforth, we use the term *convex* to mean *nontrivial convex*. Finally, note that we call sets A, B, and C *strongly incomparable* if $A \not\subseteq B \cup C, B \not\subseteq A \cup C$, and $C \not\subseteq A \cup B$.

Let $S \subset V$ be any set of vertices within a tournament. S partitions the remaining vertices into three sets: *winners*, *losers*, and *mediocre players*. The winners are the vertices that dominate all of the vertices in S. The losers are the vertices dominated by all of the vertices in S. We are interested in the mediocre players, those that dominate some of the vertices in S and are dominated by others. Let $M(S) = \{v \in V - S | \exists x, y \in S$, where $x \to v$ and $v \to y\}$ be the set of mediocre players relative to S.

Since a nontrivial convex subset contains at least two vertices, we use every two-vertex subset of V as a starting point for computing all of the convex subsets. For $i, j \in V, i \neq j$, let $C_{i,j}(k)$ be defined inductively as follows:

$$\begin{aligned} C_{i,j}(0) &= \{i, j\}, \\ C_{i,j}(k) &= C_{i,j}(k-1) \cup M(C_{i,j}(k-1)) \text{ for } k > 0. \end{aligned}$$

This definition implies a closure operator and immediately suggests an obvious



a) Tournament

b) Winners, losers, and convex subset

FIG. 1. *Example tournament with a convex subset.*

$O(n^5)$ time algorithm for computing all convex subsets. In §2 we present this algorithm more completely and prove its correctness. We also show that every convex subset is the closure of a subset of cardinality two by showing that every convex subset is equal to some $C_{i,j}$ (defined below). The time bound is easy to verify. Computing $M(C_{i,j}(k-1))$ takes $O(n^2)$ time since at most each pair of vertices needs to be considered. We can stop computing each $C_{i,j}(k)$ when $k = n-1$ or when $C_{i,j}(k) = C_{i,j}(k-1)$. Let $C_{i,j}$ denote the resulting set. Since there are $O(n^2)$ sets to compute, the claimed bound is immediate.

We use the following notation and lemma in the demonstration of a polylogarithmic time bound for our parallel algorithm in §3 as well as in our improvement of the time bound of the obvious algorithm in §4. The lemma shows a simple relationship among vertices in a convex subset and, curiously, allows us to ignore the vertex j in our algorithms.

Let $r_{i,j}(v)$ denote the *round* of induction when vertex v is brought into $C_{i,j}$. Thus, $r_{i,j}(v) = k$ if and only if $v \in C_{i,j}(k)$ and $v \notin C_{i,j}(k-1)$. Note that $r_{i,j}(i) = r_{i,j}(j) = 0$. If a vertex $v \notin C_{i,j}$, then $r_{i,j}(v) = \infty$. We use $R_{i,j}(k) = \{ v \in V \mid r_{i,j}(v) = k \}$ to denote the set of vertices brought into $C_{i,j}$ in round k.

LEMMA 1.1. *Let $i, j, v \in C_{i,j}$ be three distinct vertices such that $r_{i,j}(v) = k > 0$. Then there exists a vertex $w \in C_{i,j}$ such that $r_{i,j}(w) = k-1$ and either $w \to v \to i$ or $i \to v \to w$.*

Proof. For a given v where $r_{i,j}(v) = k > 0$, let $x, y \in C_{i,j}(k-1)$ be as in the definition of $M(C_{i,j}(k-1))$. Note that either $r_{i,j}(x) = k-1$ or $r_{i,j}(y) = k-1$, otherwise $r_{i,j}(v)$ would be less than k.

Suppose $r_{i,j}(x) = r_{i,j}(y) = k-1$. Now, either $v \to i$ or $i \to v$. Because of the nature of x and y, either x or y dominates v and the other is dominated by v. In either case, we have the claimed two-path.

Now suppose exactly one of $r_{i,j}(x)$ and $r_{i,j}(y)$ is $k-1$. Without loss of generality, assume it is $r_{i,j}(x)$. Thus, $r_{i,j}(y) \leq k-2$ and y must have the same orientation to v as i does; otherwise $r_{i,j}(v)$ would be one more than $r_{i,j}(y)$. This implies that either $x \to v \to i$ or $i \to v \to x$. □

Since we know there is always at least one such "predecessor" vertex, w, we define a function $P_{i,j} : (C_{i,j} - \{ i, j \}) \to C_{i,j}$ such that $P_{i,j}(v) = w$, where w is the lowest numbered vertex satisfying $r_{i,j}(w) = r_{i,j}(v) - 1$ and either $w \to v \to i$ or $i \to v \to w$. Note that these predecessor functions compose with themselves. We will use $P_{i,j}^l$ to denote the composition of $P_{i,j}$ with itself l times.

2. Finding all convex subsets. In this section we show that the algorithm in Fig. 2 finds all convex subsets of a tournament. The modifications used to provide a faster serial algorithm and an $\mathcal{NC}$ algorithm do not invalidate the results of this section. We begin with a useful lemma from [5].

```
1 For each pair (i, j) with 1 ≤ i < j ≤ n do
2     C_{i,j}(0) ← {i, j}
3     k ← 0
4     Repeat
5         k ← k + 1
6         M_{i,j} ← M(C_{i,j}(k − 1))
7         C_{i,j}(k) ← C_{i,j}(k − 1) ∪ M_{i,j}
8     Until |C_{i,j}(k)| = |C_{i,j}(k − 1)|
```

FIG. 2. *Sequential algorithm.*

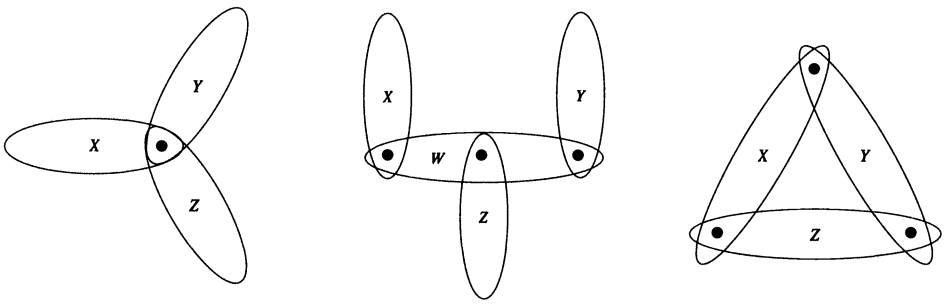
LEMMA 2.1 (see [5]). *Let A, B be convex subsets such that $A \cap B \neq \emptyset$. Then $A \cup B$ is convex.*

The following lemma is of particular importance in proving the correctness of our algorithms. We use this lemma to show that three strongly incomparable convex subsets cannot be related to one another via various "intersection" properties. Three of these intersection properties are identified in the corollaries that follow the lemma.

LEMMA 2.2. *There do not exist three distinct vertices $x, y, z \in V$ and three distinct convex subsets A, B, and C such that $x \notin A$, $y, z \in A$, $y \notin B$, $x, z \in B$, $z \notin C$, and $x, y \in C$.*

Proof. Assume that three such vertices $x, y, z \in V$ along with the corresponding convex subsets A, B, and C do exist. Consider the triangle of vertices x, y, and z. Without loss of generality, assume that $y \rightarrow z$. Since $y, z \in A$ and A is convex, either $x \rightarrow y$ and $x \rightarrow z$ or $y \rightarrow x$ and $z \rightarrow x$. If $x \rightarrow y$, then the set B cannot be convex because of the two-path $x \rightarrow y \rightarrow z$, which is a contradiction. So it must be that $y \rightarrow x$ and $z \rightarrow x$. Similarly, the two-path $y \rightarrow z \rightarrow x$ implies that C cannot be convex, leading to a contradiction and thus implying the claim. □

The following corollaries provide a more concrete interpretation of the previous lemma. The first one shows that no three strongly incomparable convex subsets share a common intersection as shown in Fig. 3a.



a) Clover leaf b) Goal post c) Triangle

FIG. 3. *Impossible convex subset overlaps.*

COROLLARY 2.3. *For any vertex $i \in V$, there do not exist three strongly incomparable convex subsets X, Y, and Z such that $i \in X \cap Y \cap Z$.*

Proof. Assume the claim is not true and let X, Y, and Z be three strongly incomparable convex subsets such that $i \in X \cap Y \cap Z$. Now, $A = Y \cup Z$, $B = X \cup Z$, and $C = X \cup Y$ are all convex due to Lemma 2.1. Since X, Y, and Z are strongly incomparable, there exists $x \in X$ such that $x \notin Y \cup Z$, there exists $y \in Y$ such that $y \notin X \cup Z$, and there exists $z \in Z$ such that $z \notin X \cup Y$. Lemma 2.2 implies that vertices x, y, and z and convex subsets A, B, and C cannot exist. This contradiction gives the desired result. □

Another impossible overlap is shown in Fig. 3b.

COROLLARY 2.4. *Let X, Y, and Z be disjoint convex subsets. There does not exist a convex subset W such that $W \cap X \neq \emptyset$, $W \cap Y \neq \emptyset$, and $W \cap Z \neq \emptyset$.*

Proof. Assume the claim is not true and let $A = Y \cup Z \cup W$, $B = X \cup Z \cup W$, and $C = X \cup Y \cup W$. The rest of the argument is similar to Corollary 2.3. □

The next corollary of Lemma 2.2 is central to the proof of correctness of our algorithms. It demonstrates that the overlap shown in Fig. 3c is also impossible.

COROLLARY 2.5. *There do not exist strongly incomparable convex subsets X, Y, and Z such that $X \cap Y \neq \emptyset$, $Y \cap Z \neq \emptyset$, and $Z \cap X \neq \emptyset$.*

Proof. Assume the claim is not true and let $A = Y \cup Z$, $B = X \cup Z$, and $C = X \cup Y$. The result now follows as in the previous two corollaries. □

We also use the following simple observation, stated without proof, in the proof of Theorem 2.7.

LEMMA 2.6. *Let $a, b, i, j \in V$. If $a, b \in C_{i,j}$, then $C_{a,b} \subseteq C_{i,j}$.*

THEOREM 2.7. *Let C be any convex subset in a tournament or the trivial convex subset consisting of the entire tournament. Then $C = C_{i,j}$ for some $i, j \in V$.*

Proof. Suppose there are convex subsets which are different than every $C_{i,j}$ for $i, j \in V$. Let C be such a convex subset of minimal cardinality. Pick any vertex $x \in C$. Consider $C_{x,c}$ for all $c \in C$, $c \neq x$. None of these $C_{x,c}$'s can be C by our assumption, and each of these $C_{x,c}$'s must be a subset of C by the definition of $C_{x,c}$. Now choose all of the "largest" $C_{x,c}$'s where largest means that $C_{x,c}$ is not a proper subset of any other $C_{x,c'}$ for $c' \in C$. Note that by Corollary 2.3 there are at most two such largest convex subsets. Also note that there must be at least two such largest convex subsets, otherwise the largest would be equal to C, violating our initial assumption. Therefore, vertex x divides C into two subsets $A \subset C$ and $B \subset C$ such that $A \cup B = C$ and $x \in A \cap B$. Choose a vertex $y \in A$ and $z \in B$ such that $C_{x,y} = A$ and $C_{x,z} = B$. Such a y and z must exist because of the minimality of C. Note that $y, z \notin A \cap B$ since $C_{a,b} \subseteq A \cap B$ for any $a, b \in A \cap B$.

We now turn our attention to $C_{y,z}$ and ask the question, "How big is $C_{y,z}$?" Using $C_{x,y}, C_{x,z}$, and $C_{y,z}$, Corollary 2.5 implies that $x \in C_{y,z}$. But, since $x, y \in C_{y,z}$, we know that $C_{x,y} \subseteq C_{y,z}$ by Lemma 2.6. Similarly, since $x, z \in C_{y,z}$, we know that $C_{x,z} \subseteq C_{y,z}$. We therefore have $C_{y,z} = C$, a statement which contradicts our original assumption. □

Note that Theorem 2.7 does not extend to half-splits in general digraphs because half-splits are not necessarily the closure of subsets of cardinality two. However, a direct result of this theorem is a tight upper bound on the number of convex subsets in a tournament.

COROLLARY 2.8. *There are at most $n(n+1)/2 + 1$ convex subsets (including trivial ones) in any tournament on n vertices. Furthermore, this bound is tight.*

Proof. Theorem 2.7 implies an upper bound of $n(n-1)/2 - 1$ nontrivial convex subsets. Clearly, there are exactly $n + 2$ trivial convex subsets. The tightness of the bound follows from Varlet's claim that a transitive tournament has exactly $n(n+1)/2 + 1$ convex subsets [5]. □

3. Finding convex subsets in parallel. In this section we describe a parallel algorithm that finds all convex subsets in $O(\log^2 n)$ time using $O(n^4)$ processors on a CREW-PRAM. (See Já Já [4] for a description of CREW-PRAM.)

An obvious parallelization of the sequential algorithm given in §2 assigns a group of processors to each set $C_{i,j}$. This algorithm, however, may not result in a polylogarithmic time algorithm since the computation of each $C_{i,j}$ may require as many as $n-1$ rounds. A more aggressive approach that takes advantage of Lemmas 1.1 and 2.6 substantially reduces the number of rounds.

We introduce slightly different notation for the intermediate sets in the parallel computation of $C_{i,j}$. This new notation is used to differentiate between the two constructions. Intuitively, $C'_{i,j}(k)$ denotes the set of vertices in $C_{i,j}$ after k rounds of the parallel algorithm. A more formal notion comes from the algorithm given in Fig. 4.

```
1  For each pair (i, j) with 1 ≤ i < j ≤ n pardo
2      C'_{i,j}(0) ← {i, j}
3      k ← 0
4      Repeat
5          k ← k + 1
6          M'_{i,j} ← M(C'_{i,j}(k − 1))
7          C'_{i,j}(k) ← C'_{i,j}(k − 1) ∪ M'_{i,j}
8          For each x ∈ M'_{i,j} pardo
9              C'_{i,j}(k) ← C'_{i,j}(k) ∪ C'_{i,x}(k − 1)
10     Until |C'_{i,j}(k)| = |C'_{i,j}(k − 1)|
```

FIG. 4. *Parallel algorithm.*

THEOREM 3.1. *The algorithm in Fig. 4 finds all of the convex subsets of a tournament.*

Proof. The only difference between Fig. 4 and the algorithm in Fig. 2 is the extra loop in lines 8 and 9. This does not alter the output of the algorithm as demonstrated by Lemma 2.6. □

The following lemma shows the time bound for the algorithm in Fig. 4.

LEMMA 3.2. *The Repeat loop in lines 4–10 of Fig. 4 stops after $O(\log n)$ iterations.*

Proof. We show this by proving the following claim. For every vertex $v \in V$, either $r_{i,j}(v) = \infty$ or $v \in C'_{i,j}(k)$, where $k = \lceil \log_2(r_{i,j}(v) + 1) \rceil$. We prove this by induction on k. The base case of $k = 0$ is trivial in that the only vertices that cause $k = 0$ are i and j. Assume that the claim holds for $k \geq 0$. We now show that the claim holds for $k+1$. Consider a vertex v such that $2^k \leq r_{i,j}(v) < 2^{k+1}$. By following the sequence of vertices $P_{i,j}(v), P^2_{i,j}(v), \ldots, P^l_{i,j}(v)$ until $r_{i,j}(P^l_{i,j}(v)) = 2^k - 1$, we find a "magnet" vertex $m = P^{l-1}_{i,j}(v)$. If $l = 1$, then $m = v$, and by our induction hypothesis, $P_{i,j}(m) \in C'_{i,j}(k)$. Even though $m \notin C'_{i,j}(k)$, after line 7 of the algorithm $m \in C'_{i,j}(k+1)$. What remains to be shown is that if $m \neq v$, then $v \in C'_{i,j}(k+1)$ after line 9 of the algorithm. Since $m = P^{l-1}_{i,j}(v)$ and $l \leq 2^k$, $r_{i,m}(v) < 2^k$. This implies that $v \in C'_{i,m}(k)$ by our induction hypothesis. Thus $v \in C'_{i,j}(k+1)$ and the lemma follows. □

THEOREM 3.3. *The algorithm in Fig. 4 finds all convex subsets in an n vertex tournament in $O(\log^2 n)$ time using $O(n^4)$ processors in the CREW-PRAM computing model.*

Proof. We begin by analyzing the work required for a single vertex pair (i, j).

Then, we simply multiply the processor requirement by $O(n^2)$ to arrive at the claimed resource bound.

$O(n^2)$ processors can compute $M'_{i,j}$ (line 6) in $O(\log n)$ time without the need for concurrent write. We assign a processor to each pair of vertices x, y, for $x \in C'_{i,j}(k-1)$, $y \notin C'_{i,j}(k-1)$. Each processor checks whether $x \to y \to i$ or $i \to y \to x$. If so, this processor outputs a 0, otherwise it outputs a 1. (Note that by Lemma 1.1 this check is sufficient.) The output goes into an array where the $|C'_{i,j}(k-1)|$ outputs associated with the vertex y are stored. Adding these outputs will determine whether y belongs in the set $M'_{i,j}$. The union at line 7 can clearly be done in $O(1)$ time using $O(n)$ processors without using concurrent write. The $O(n)$ unions at line 9 of the algorithm take $O(\log n)$ time using $O(n^2)$ processors using the exclusive write model.

Since there are $O(n^2)$ vertex pairs, the total processor requirement is $O(n^4)$. Furthermore, since each of the $O(\log n)$ iterations of the repeat loop of lines 4–10 takes $O(\log n)$ time, the claimed time bound holds. □

4. A better analysis of the sequential algorithm. The running time we originally gave for the sequential algorithm in Fig. 2 is not as tight as possible. Here we improve our analysis by showing that the computation of the mediocre sets at line 6 is not as expensive as first observed.

THEOREM 4.1. *The algorithm in Fig. 2 finds all the convex subsets of a tournament in $O(n^4)$ time.*

Proof. Clearly the For loop at line 1 requires $O(n^2)$ iterations. Thus, to remain within our claimed time bound the Repeat loop at lines 4–8 must not consume more than $O(n^2)$ time. We already know that it may take as many as $n-1$ iterations and that the union operation requires $O(n)$ time. We have already consumed $O(n^2)$ time in the Repeat loop.

What remains to be shown is that the computation of $M(C_{i,j}(k-1))$ does not cause the time bound to be exceeded. This is done by looking at the total work performed over all such computations of mediocre sets (hence, over all iterations of the Repeat loop). Lemma 1.1 tells us that we need consider only the vertices $x \in R_{i,j}(k-1)$, vertices $y \in V - C_{i,j}(k-1)$, and the vertex i in the following manner. For $x \in R_{i,j}(k-1)$ and all vertices $y \notin C_{i,j}(k-1)$ we determine whether there exists a two-path of the form $x \to y \to i$ or $i \to y \to x$. The vertex y is included in $M(C_{i,j}(k-1))$ only if such a two-path exists. This observation implies every pair of vertices will be considered at most once. Since this check is done in $O(1)$ time, the Repeat loop at lines 4–8 will take $O(n^2)$ time. □

This algorithm is asymptotically not as fast as Bouchet's algorithm for finding a split of a digraph. However, our algorithm finds all convex subsets (or splits) in $O(n^4)$ time, whereas Bouchet's algorithm finds only one in $O(n^3)$ time. It is not clear how to modify Bouchet's algorithm to find all of the convex subsets.

5. Summary. We have proven properties about the relationships among convex subsets in tournaments that provide a basis for sequential and parallel algorithms for finding convex subsets in tournaments. Because of these properties, our algorithm can be easily modified to an $O(n^3)$ time algorithm to determine all of the convex subsets that contain a given subset of vertices. Let S be a given subset. Let $V - S = \{v_1, v_2, \ldots, v_k\}$. Now all convex subsets that contain S can be computed by using $S, S \cup \{v_1\}, \ldots, S \cup \{v_k\}$ as starting points for computing mediocre sets.

Computing the smallest convex subset that contains a set S can be done simply by using S as the starting point. This can be done in $O(n^2)$ time if S has at least

two vertices. If $|S| = 1$ then, using this approach, all the starting subsets mentioned above must be used and the time bound is $O(n^3)$.

Finally, note that the parallel version of our algorithm will solve both of the variations mentioned above in $O(\log^2 n)$ time using $O(n^4)$ processors.

Appendix: Splits in tournaments. Here we show the relationship between finding convex subsets in a tournament and finding splits in a digraph when the digraph is restricted to a tournament. We use notation adapted from [3]. Let $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$ be two digraphs such that $V_1 \cap V_2 = \{v\}$. Define a *composition* of G_1 and G_2, written as $G = G_1 * G_2$, as follows. Let $A_{in} = \{(x,y) \mid (x,y) \in A_1 \cup A_2, x \neq v \neq y\}$. Let $A_{cross} = \{(x,y) \mid (x,v) \in A_1$ and $(v,y) \in A_2$ or $(x,v) \in A_2$ and $(v,y) \in A_1\}$. Now, let $G = (V, A)$, where $V = V_1 \cup V_2 - \{v\}$ and $A = A_{in} \cup A_{cross}$. Given a digraph $G = G_1 * G_2$, we call $\{G_1, G_2\}$ a *simple decomposition* of G, and $\{V_1, V_2\}$ is a *split* of G with the associated *marker* element v. This discussion assumes that the convex subsets are nontrivial and that $|V_1| \geq 2 \leq |V_2|$.

We now claim that a split of a tournament exists if and only if the tournament has a convex subset. We show this by demonstrating that a convex subset implies a split and that either V_1 or V_2 is a convex subset if $\{V_1, V_2\}$ is a split.

THEOREM A.1. *If a tournament has a convex subset, then the tournament has a split.*

Proof. Let C be a convex subset of the tournament $G = (V, A)$. We create the simple decomposition as follows. Let $V_1 = C \cup \{v\}$ and $V_2 = (V - C) \cup \{v\}$. Let $A_1 = \{(x,y) \mid x,y \in C\} \cup \{(x,v) \mid x \in C\} \cup \{(v,x) \mid x \in C\}$. Let $A_2 = \{(x,y) \mid x,y \in V - C\} \cup \{(x,v) \mid x \in V - C$ and $x \Rightarrow C\} \cup \{(v,x) \mid x \in V - C$ and $C \Rightarrow x\}$. Clearly $\{V_1, V_2\}$ is a split. □

THEOREM A.2. *If a tournament has a split $\{V_1, V_2\}$, then either $V_1 - \{v\}$ or $V_2 - \{v\}$ is a convex subset, where v is the associated marker.*

Proof. Suppose that $V_1 - \{v\}$ is not convex. Then, there exists vertices $a, c \in V_1$ and $b \in V_2$ such that either $a \rightarrow b \rightarrow c$ or $c \rightarrow b \rightarrow a$. Thus, the edge set A_2 must contain (b,v) and (v,b). This condition implies that for every vertex $x \in V_1$ either $(x,v) \in A_1$ or $(v,x) \in A_1$, but not both, since the tournament does not have any bidirectional edges. Furthermore, since the tournament must have an edge between every pair of vertices, if the edge $(x,v) \in A_1$, then $x \Rightarrow V_2$. Similarly, if the edge $(v,x) \in A_1$, then $V_2 \Rightarrow x$. Hence, $V_2 - \{v\}$ is convex. □

Acknowledgments. We extend our gratitude to Max Hailperin and Giora Slutzki for their careful reading of and comments on an early draft of this paper and to the anonymous referee for comments that led to clarification of many points in the paper.

REFERENCES

[1] A. ASTIE-VIDAL AND A. MATTEO, *Non-simple tournaments: Theoretical properties and a polynomial algorithm*, in Proc. of the Fifth Applied Algebra, Algebraic Algorithms and Error-Correcting Codes International Conference, Lecture Notes in Computer Science Vol. 5, Springer–Verlag, Berlin, New York, 1989, pp. 1–15.

[2] A. BOUCHET, *Digraph decompositions and eulerian systems*, SIAM J. Algebraic Discrete Meth., 8 (1987), pp. 323–337.

[3] W. H. CUNNINGHAM, *Decomposition of directed graphs*, SIAM J. Algebraic Discrete Meth., 3 (1982), pp. 214–228.

[4] J. JÁ JÁ, *An Introduction to Parallel Algorithms*, Addison-Wesley, Reading, MA, 1992.

[5] J. C. VARLET, *Convexity in tournaments*, Bull. Soc. Roy. Sci. Liège, 45 (1976), pp. 570–586.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 71–86, February 1996

008

HORIZONTAL PRINCIPAL STRUCTURE OF LAYERED MIXED MATRICES: DECOMPOSITION OF DISCRETE SYSTEMS BY DESIGN-VARIABLE SELECTIONS*

SATORU IWATA† AND KAZUO MUROTA‡

Abstract. A matrix $A = \binom{Q}{T}$ is called a layered mixed matrix (LM-matrix) if the set of nonzero entries of T is algebraically independent over the field to which the entries of Q belong. This concept has been proposed as a mathematical tool for describing discrete physical/engineering systems. It is known that there uniquely exists a finest block-triangularization of an LM-matrix, which is called the combinatorial canonical form (CCF).

In this paper, associated with an LM-matrix we introduce a new submodular function q characterizing its rank. This submodular function q is defined on a modular lattice. It will be shown that the principal structure of q gives the coarsest decomposition of the row side that is finer than any decomposition induced by the CCF of the submatrix consisting of a base of the column vectors of A. This gives a best possible bound on the extent to which the whole system can be decomposed by a suitable choice of design variables.

Key words. layered mixed matrix, design variable, combinatorial canonical form (CCF), principal structure, submodular function

AMS subject classifications. 15A21, 93A15, 05C50

1. Introduction.

Consider a discrete physical/engineering system described by a set of variables $\boldsymbol{x} = (x_j \mid j \in C)$ subject to a set of linear equations $A\boldsymbol{x} = \boldsymbol{b}$ such as an electric network. When the system of equations is not uniquely solvable, i.e., the rank r of A is smaller than n, the number of variables, we must pick suitable $n - r$ variables and specify their values so that the state of the system is determined uniquely. The variables to be picked are called *design variables* in engineering terms, and usually we have a chance to choose a nice set of design variables. From a computational point of view, it is desirable to select a set of design variables that makes the remaining system hierarchically decomposable as finely as possible. Even though we may not expect an optimal system in this respect, we would like to know "How fine can we decompose the system after a clever choice of design variables?" We will answer this question in this paper by using the extension of the principal structure for submodular systems proposed by Fujishige [5].

To be more concrete, let us begin with an electric network shown in Fig. 1. Such a network is called a two-port network since it has two pairs of terminals open for the connection with other elements such as current/voltage sources (cf. Iri [7] and Recski [19]). We denote by ξ^μ and η_μ the current in branch e_μ and the voltage across branch e_μ, respectively. All these variables are governed by the following system of

* Received by the editors May 14, 1993; accepted for publication (in revised form) January 11, 1995.

† Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606, Japan (iwata@kurims.kyoto-u.ac.jp). The research of this author was supported by the Japan Society for the Promotion of Science Fellowship for Japanese Junior Scientists.

‡ Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606, Japan. The research of this author was partially supported by the Sumitomo Foundation.

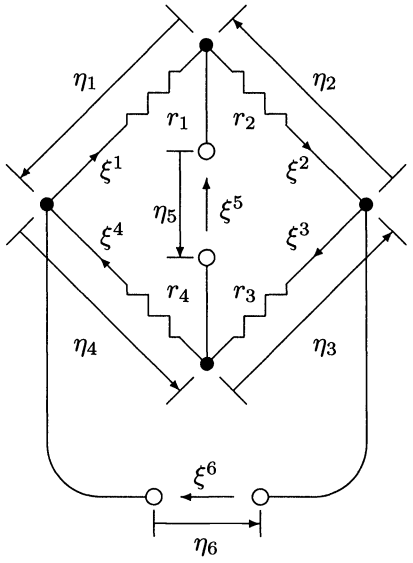


FIG. 1. *An electric network (from Iri [7]).*

equations:

$$(1)\qquad \left(\begin{array}{cccccccccccc} 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ \hline r_1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_2 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_3 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_4 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{array}\right) \left(\begin{array}{c} \xi^1 \\ \xi^2 \\ \xi^3 \\ \xi^4 \\ \xi^5 \\ \xi^6 \\ \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \\ \eta_5 \\ \eta_6 \end{array}\right) = \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right).$$

The coefficient matrix is of row-full rank, but the rank is less than the number of variables by two, which implies the system of equations is not uniquely solvable unless we specify the values of these certain two variables. Although we have various candidates of such variables, we pay attention here to two examples.

For the first instance, we choose $\{\xi^5, \xi^6\}$ as a set of design variables. This means in practice that we attach current sources to branch e_5 and branch e_6. The coefficient matrix of the system of equations that governs the remaining variables can be rewritten in a block-triangular form with three diagonal blocks by certain permutations of

the row-set and the column-set as follows:

$$\begin{array}{c} \begin{matrix} \eta_5 & \eta_6 & \xi^1 & \xi^2 & \xi^3 & \xi^4 & \eta_1 & \eta_2 & \eta_3 & \eta_4 \end{matrix} \\ \begin{pmatrix} 1 & & & & & & & 1 & 1 & \\ & 1 & & & & & 1 & 1 & & \\ & & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ & & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ & & r_1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ & & 0 & r_2 & 0 & 0 & 0 & -1 & 0 & 0 \\ & & 0 & 0 & r_3 & 0 & 0 & 0 & -1 & 0 \\ & & 0 & 0 & 0 & r_4 & 0 & 0 & 0 & -1 \end{pmatrix} \end{array}.$$

The block-triangularization of a matrix under two independent permutations of the row-set and the column-set is obtained by the *Dulmage–Mendelsohn decomposition* (or *DM-decomposition*) [4] of a bipartite graph representing the zero/nonzero patterns of the matrix.

When we talk about a decomposition of a discrete physical/engineering system, we must consider a transformation group of physical importance. In this respect the mere permutations are not adequate to be considered. It is an essential observation that the equations that govern a discrete physical/engineering system are classified into two types: *structural equations* such as Kirchhoff's laws and *constitutive equations* such as Ohm's laws. Accordingly, the rows of the coefficient matrix are divided into two groups. For example, the first six rows of the matrix in (1) represent structural equations, and the last four rows represent constitutive equations. Such a matrix is called a *layered mixed matrix* or *LM-matrix* (see §2 for the precise definition). As a natural transformation for an LM-matrix we should admit not only permutations but also the linear combinations of the structural equations. Such transformation is called an *admissible transformation* and the finest block-triangularization of an LM-matrix under admissible transformations is known to exist and is called the *combinatorial canonical form* or *CCF* [14], [18] (see §2 for more detail). It can be shown that the above matrix cannot be decomposed any more by admissible transformations.

Suppose instead that we choose $\{\xi^5, \eta_6\}$ as a set of design variables or, in other words, we attach a current source to branch e_5 and a voltage source to branch e_6. Then the coefficient matrix of the remaining system can be decomposed into four blocks as follows by an admissible transformation for an LM-matrix:

$$\begin{array}{c} \begin{matrix} \xi^6 & \eta_5 & \xi^1 & \xi^2 & \eta_1 & \eta_2 & \xi^3 & \xi^4 & \eta_3 & \eta_4 \end{matrix} \\ \begin{pmatrix} -1 & & 1 & & & & & -1 & & \\ & 1 & & & & 1 & & & 1 & \\ & & -1 & 1 & 0 & 0 & & & & \\ & & 0 & 0 & 1 & 1 & & & & \\ & & r_1 & 0 & -1 & 0 & & & & \\ & & 0 & r_2 & 0 & -1 & & & & \\ & & & & & & 1 & -1 & 0 & 0 \\ & & & & & & 0 & 0 & 1 & 1 \\ & & & & & & r_3 & 0 & -1 & 0 \\ & & & & & & 0 & r_4 & 0 & -1 \end{pmatrix} \end{array}.$$

Thus we observe that how fine the remaining system can be decomposed depends on the selection of design variables. In this paper we are interested in an upper bound

on the decomposition, i.e., the coarsest decomposition of the row side that is finer than any decomposition induced by the CCF of a submatrix consisting of a base of the column vectors. See Example 2.1 for a concrete example.

A matrix is said to be a generic matrix if its nonzero elements are independent indeterminates (see Brualdi and Ryser [2] for the precise definition). Admissible transformations for a generic matrix are independent permutations of the row-set and the column-set. In this case the finest block-triangularization is obtained by the DM-decomposition. When we are restricted to generic matrices, the bound of our present interest has been given by McCormick [12] and called "SP-decomposition." See also Chang and McCormick [3] and McCormick and Chang [13] for related topics.

As to LM-matrices, Murota [16] characterized the coarsest decomposition of the column-set finer than any decomposition induced by the CCF of a submatrix consisting of a base of row vectors by applying the concept of principal structure of submodular systems proposed by Fujishige [5] to a submodular function p associated with an LM-matrix. This result gives an interpretation of McCormick's SP-decomposition in terms of submodular systems as well. However, it does not lead us directly to a solution of our present problem, because what we would like to know is concerned with submatrices consisting of a base of not row vectors but column vectors and the transpose of an LM-matrix is no longer an LM-matrix.

Recently, block-triangularizations of partitioned matrices, which contain the CCF of LM-matrices as a special case, were investigated in [8], where a submodular function p on a modular lattice was introduced in association with a partitioned matrix. The transpose of an LM-matrix is also a partitioned matrix as well, and we are lead to the idea of introducing a new submodular function q associated with an LM-matrix. Although we will not explicitly deal with a transposed LM-matrix, the definition of the new submodular function q in this paper comes from the application of the submodular function for partitioned matrices to transposed LM-matrices. The new submodular function q is defined on a nondistributive lattice. Therefore we need to extend the notion of principal structure to a submodular function on general lattices, whereas Fujishige [5] assumes boolean lattices. Such an extension was already given by Tomizawa and Fujishige [20]. Then we will show in Theorem 4.3 that the principal structure of the submodular function q gives the best possible upper bound on the extent to which the discrete physical/engineering system described by the LM-matrix can be decomposed after a suitable choice of design variables.

This result leads us to call the principal structure of the submodular function q the *horizontal principal structure* of an LM-matrix. In contrast, the principal structure of p will be called here the *vertical principal structure* of an LM-matrix, although Murota [16] simply called it the *principal structure* of an LM-matrix.

The outline of this paper is as follows. Section 2 covers the preliminaries on the CCF of LM-matrices. Section 3 is devoted to the extension of the notion of the principal structure. Sections 4 and 5 contain the main theorem and its proof, respectively.

2. Preliminaries on LM-matrices. Let $\boldsymbol{K}$ be a subfield of $\boldsymbol{F}$. An $m \times n$ matrix $A = \binom{Q}{T}$ is called an *LM-matrix* with respect to $\boldsymbol{F}/\boldsymbol{K}$ when Q is an $m_Q \times n$ matrix over $\boldsymbol{K}$, T is an $m_T \times n$ matrix over $\boldsymbol{F}$, and the set of nonzero entries of T is algebraically independent over $\boldsymbol{K}$. For a matrix M in general, we denote the row-set and the column-set of M by $Row(M)$ and $Col(M)$, respectively, and the submatrix with the row-set R_* and the column-set C_* by $M[R_*, C_*]$.

The rank of an LM-matrix is characterized by a submodular function p defined

on column subsets as follows. Given an LM-matrix $A = \binom{Q}{T}$, set $R_Q = Row(Q)$, $R_T = Row(T)$, and $C = Col(A)$. Put

$$p(J) = \rho(J) + \gamma(J) - |J|, \quad J \subseteq C,$$

where $\rho(J)$ denotes the rank of $Q[R_Q, J]$ and $\gamma(J)$ the number of nonzero rows in $T[R_T, J]$. Then the function $p : 2^C \to Z$ is submodular:

$$p(J_1) + p(J_2) \geq p(J_1 \cup J_2) + p(J_1 \cap J_2), \quad J_h \subseteq C \ \ (h = 1, 2).$$

The following identity is fundamental.

LEMMA 2.1 (rank identity for an LM-matrix [14], [18]). *For an LM-matrix A,*

$$\mathrm{rank}A = \min\{p(J) \mid J \subseteq C\} + |C|.$$

The *combinatorial canonical form* (*CCF*) of an LM-matrix A is the finest proper block-triangularization by admissible transformations:

$$\widetilde{A} = \begin{pmatrix} S_Q & O \\ O & I_T \end{pmatrix} \begin{pmatrix} Q \\ T \end{pmatrix},$$

where S_Q is an m_Q-dimensional nonsingular matrix over the subfield $\boldsymbol{K}$ and I_T is the m_T-dimensional unit matrix. We say that A and A' are *LM-equivalent* if they are connected by an admissible transformation.

We say that $\widetilde{A}$ is in *block-triangular form* or is *block-triangularized* if $R = Row(\widetilde{A})$ and $C = Col(\widetilde{A})$ are split into a certain number of blocks, that is, $(R_0; R_1, \ldots, R_b; R_\infty)$ and $(C_0; C_1, \ldots, C_b; C_\infty)$ in such a way that

$$\begin{array}{lll} |R_0| < |C_0| & \text{or} & |R_0| = |C_0| = 0, \\ |R_k| = |C_k| > 0 & \text{for} & k = 1, \ldots, b, \\ |R_\infty| > |C_\infty| & \text{or} & |R_\infty| = |C_\infty| = 0, \end{array}$$

and

$$\widetilde{A}[R_k, C_l] = O \quad \text{if} \quad 0 \leq l < k \leq \infty.$$

An LM-matrix $\widetilde{A}$ is said to be *properly* block-triangularized if in addition

$$\mathrm{rank}\widetilde{A}[R_k, C_k] = \min(|R_k|, |C_k|) \quad \text{for} \quad k = 0, 1, \ldots, b, \infty$$

is satisfied. Note that

$$\mathrm{rank}\widetilde{A} = m - |R_\infty| + |C_\infty| = n - |C_0| + |R_0| \tag{2}$$

if $\widetilde{A}$ is properly block-triangularized. The submatrices $\widetilde{A}[R_0, C_0]$ and $\widetilde{A}[R_\infty, C_\infty]$ are called the *horizontal tail* and the *vertical tail* of $\widetilde{A}$, respectively. It is clear that if $\widetilde{A}$ is block-triangularized in the above sense, we can put it into an explicit upper block-triangular form $\bar{A} = P_r \widetilde{A} P_c$ in the usual sense by using certain permutation matrices P_r and P_c.

Moreover, for a block-triangularized matrix $\widetilde{A}$, a partial order is naturally induced among the blocks $(C_1, \ldots, C_b)$ by the zero/nonzero structure of $\widetilde{A}$. The partial order $\preceq$ is the reflexive and transitive closure of the relation defined by

$$C_k \text{ is "smaller" than or equal to } C_l \text{ if } \widetilde{A}[R_k, C_l] \neq O.$$

As is well known [1], the ideals of the poset $(\{C_1, \dots, C_b\}, \preceq)$ constitute a distributive lattice, which we denote by $\mathcal{D}_*(\widetilde{A})$. By adding C_0 to each element of $\mathcal{D}_*(\widetilde{A})$, we obtain an isomorphic lattice $\mathcal{D}(\widetilde{A})$, i.e.,

$$\mathcal{D}(\widetilde{A}) = \{C_* \cup C_0 \mid C_* \in \mathcal{D}_*(\widetilde{A})\}. \tag{3}$$

Suppose both $\widetilde{A}$ and $\widetilde{A}'$ are LM-equivalent to A. Then we say that $\widetilde{A}$ is finer, as a decomposition of A, than $\widetilde{A}'$ if $\mathcal{D}(\widetilde{A}')$ is a sublattice of $\mathcal{D}(\widetilde{A})$.

LEMMA 2.2 (the CCF of an LM-matrix [14], [18]). *For an LM-matrix A, there uniquely exists a properly block-triangularized matrix $\widetilde{A}$ that is finer than any other properly block-triangularized matrix LM-equivalent to A.*

The block-triangular matrix $\widetilde{A}$ in Lemma 2.2 is called the CCF (combinatorial canonical form) of an LM-matrix A. The submodular function p has played an essential role in the construction of the CCF. In particular, the set of minimizers of p is a sublattice of 2^C, which agrees with the distributive lattice $\mathcal{D}(\widetilde{A})$ for $\widetilde{A}$, i.e., the CCF of A.

It should be noted here that the CCF of LM-matrices is a proper extension of the Dulmage–Mendelsohn decomposition for bipartite graphs [2], [4], [11]. See [15], [17] for other properties and applications of LM-matrices.

Example 2.1. Consider the LM-matrix

$$A = \begin{array}{c} \\ y_1 \\ y_2 \\ z_1 \\ z_2 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 2 & 0 \\ t_1 & & & & t_2 \\ t_3 & & t_4 & t_5 & \end{pmatrix} \end{array},$$

where $C = \{x_1, x_2, x_3, x_4, x_5\}$, $R_Q = \{y_1, y_2\}$, and $R_T = \{z_1, z_2\}$. The whole matrix A is the horizontal tail of its CCF, i.e., $C_0 = C$.

Let us illustrate our present problem using this instance. Put $B_h = C - \{x_h\}$; then submatrix $A[R, B_h]$ is nonsingular for each $h = 1, \dots, 5$. The CCFs of $A[R, B_h]$'s are as follows:

$$\widetilde{A}_1 = \begin{pmatrix} 1 & 1 & & \\ & t_5 & t_4 & \\ & & 1 & 1 \\ & & & t_2 \end{pmatrix}, \quad \widetilde{A}_2 = \begin{pmatrix} 0 & 1 & 1 & \\ t_1 & 0 & t_2 & \\ t_3 & t_4 & 0 & t_5 \\ & & & 1 \end{pmatrix}, \quad \widetilde{A}_3 = \begin{pmatrix} 1 & 1 & & \\ & t_5 & t_3 & \\ & & t_1 & t_2 \\ & & & 1 \end{pmatrix},$$

$$\widetilde{A}_4 = \begin{pmatrix} 0 & 1 & 1 & \\ t_1 & 0 & t_2 & \\ t_3 & t_4 & 0 & \\ & & & 1 \end{pmatrix}, \quad \widetilde{A}_5 = \begin{pmatrix} 1 & 1 & & \\ & t_5 & t_3 & t_4 \\ & & t_1 & \\ & & & 1 \end{pmatrix},$$

where $\widetilde{A}_h$ denotes the CCF of $A[R, B_h]$. We are interested in the common refinement of these decompositions.

3. Principal structure of submodular functions. This section is devoted to the extension, proposed in [20], of the principal structure of submodular systems [5] to that of submodular functions on arbitrary lattices. The following definition agrees with the original one when $\mathcal{L}$ is a boolean lattice, as will be explained in Remark 3.1.

Let $\mathcal{L}$ be a lattice with finite length and f a submodular function on it, i.e.,

$$f(X) + f(Y) \geq f(X \vee Y) + f(X \wedge Y), \quad X, Y \in \mathcal{L}.$$

The partial order $\preceq$ in $\mathcal{L}$ is defined by

$$X \preceq Y \text{ if } X \vee Y = Y \text{ or equivalently } X \wedge Y = X.$$

Given an element $X \in \mathcal{L}$ we denote by $D(f; X)$ the minimum element of the sublattice

$$\{Y \in \mathcal{L} \mid X \preceq Y,\ f(Y) = \min\{f(Y') \mid X \preceq Y' \in \mathcal{L}\}\ \}.$$

A mapping $\varphi : \mathcal{L} \to \mathcal{L}$ is said [1] to be a *closure function* if it satisfies the following three conditions.

(CL0) $\forall X \in \mathcal{L} : X \preceq \varphi(X)$.
(CL1) $\forall X, Y \in \mathcal{L} : X \preceq Y \Rightarrow \varphi(X) \preceq \varphi(Y)$.
(CL2) $\forall X \in \mathcal{L} : \varphi(\varphi(X)) = \varphi(X)$.

Then we have the following lemma.

LEMMA 3.1. *The mapping $D(f; \cdot) : \mathcal{L} \to \mathcal{L}$ is a closure function on $\mathcal{L}$.*

Proof. The conditions (CL0) and (CL2) are immediate from the definition. The condition (CL1) is proved as follows. Because of the definition of $D(f; \cdot)$, we have

$$f(D(f;Y)) \leq f(D(f;X) \vee D(f;Y)).$$

It follows from the submodularity of f and the above inequality that

$$f(D(f;X)) \geq f(D(f;X) \wedge D(f;Y)).$$

On the other hand, if $X \preceq Y$, it holds that $X \preceq D(f;X) \wedge D(f;Y)$. Thus from the minimality of $D(f;X)$ we have $D(f;X) = D(f;X) \wedge D(f;Y)$, which implies $D(f;X) \preceq D(f;Y)$. □

For a closure function φ, it can easily be shown that $\varphi(X \wedge Y) = X \wedge Y$ if $\varphi(X) = X$ and $\varphi(Y) = Y$. That is to say, the family $\{X \in \mathcal{L} \mid \varphi(X) = X\}$ of "closed sets" is a lower semilattice. Therefore the subset $\mathcal{K}(f)$ defined by

$$\mathcal{K}(f) = \{X \in \mathcal{L} \mid D(f;X) = X\}$$

is a lower semilattice containing the maximum element of $\mathcal{L}$. We say that $\mathcal{K}(f)$ is the *principal structure* of $(\mathcal{L}, f)$. Denoting the minimum sublattice that contains $\mathcal{K}(f)$ by $\mathcal{L}(f)$, we will call $\mathcal{L}(f)$ the *principal sublattice* of $(\mathcal{L}, f)$.

Remark 3.1. Originally in [5], the principal structure of submodular systems is defined as follows. See also [6] for the details on submodular systems.

Let E be a finite set and $f : 2^E \to Z$ be a submodular function, i.e.,

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y), \quad X, Y \subseteq E.$$

The pair (E, f) is called a *submodular system*. Given an element $e \in E$, we denote by $D(f; e)$ the minimum element of the distributive lattice

$$\mathcal{D}(f; e) = \{X \subseteq E \mid e \in X,\ f(X) = \min\{f(Y) \mid e \in Y \subseteq E\}\}.$$

Since the relation $\sqsubseteq$ defined by

$$e \sqsubseteq e' \Leftrightarrow e \in D(f; e')$$

is reflexive and transitive, E is decomposed into partially ordered blocks as follows. Consider the equivalence relation $\sim$ defined by

$$e \sim e' \Leftrightarrow e \sqsubseteq e',\ e' \sqsubseteq e,$$

and split E into the equivalence classes $\{E_1, \ldots, E_s\}$. A partial order $\preceq$ is induced among the equivalence classes in such a way that $E_k \preceq E_l$ iff $e \sqsubseteq e'$ for $e \in E_k$ and $e' \in E_l$. This decomposition, together with the partial order $\preceq$ among the blocks, is called the *principal structure* of the submodular system (E, f).

Put $\mathcal{L} = 2^E$, and then the principal sublattice $\mathcal{L}(f)$ in the sense of the present paper is a distributive lattice, which, according to Birkhoff's representation theorem [1], can be represented as a family of ideals of a poset. This poset is nothing but the principal structure of submodular system (E, f) in Fujishige's sense.

4. Result. In addition to the function p described in §2 we introduce here another submodular function q associated with an LM-matrix A. This submodular function turns out to give a solution to our problem as is stated in Theorem 4.3.

For an LM-matrix $A = \binom{Q}{T}$, we may interpret the $m_Q \times n$ matrix Q as a representation of a linear transformation from $\boldsymbol{K}^n$ to the dual space $V_Q{}^*$ of V_Q, where $V_Q \cong \boldsymbol{K}^{m_Q}$ and $V_Q{}^* \cong \boldsymbol{K}^{m_Q}$.[1]

Consider the lattice $\mathcal{L}$ that consists of the pairs of a subspace of V_Q and a subset of R_T. The operations $\wedge$ and $\vee$ in $\mathcal{L}$ are defined by

$$\left.\begin{array}{rcl}(W_1, I_1) \wedge (W_2, I_2) & = & (W_1 \cap W_2, I_1 \cap I_2) \\ (W_1, I_1) \vee (W_2, I_2) & = & (W_1 + W_2, I_1 \cup I_2)\end{array}\right\} \quad W_h \subseteq V_Q,\ I_h \subseteq R_T\ \ (h = 1, 2).$$

Obviously $\mathcal{L}$ is a modular lattice. For a subspace W of V_Q, we denote by $W^\perp$ the subspace of $V_Q{}^*$ annihilating W, i.e.,

$$W^\perp = \{v \in V_Q{}^* \mid \langle w, v \rangle = 0,\ \forall w \in W\},$$

where $\langle \cdot\ ,\ \cdot \rangle$ means the inner product (dual pairing). For $H \subseteq R_Q$, we denote by $\Upsilon(H)$ the subspace of V_Q defined by

$$\Upsilon(H) = \{w \in V_Q \mid \langle w, e_i \rangle = 0,\ \forall i \notin H\},$$

where $e_i \in V_Q$ is the ith unit vector. Note that $\dim \Upsilon(H) = |H|$.

We introduce a function $\kappa : \mathcal{L} \times 2^C \to Z$ defined by

$$\kappa((W, I), J) = |\{j \in J \mid Q_j \notin W^\perp \text{ or } \exists i \in I \text{ s.t. } T_{ij} \neq 0\}|,$$

where Q_j denotes the column vector of Q indexed by $j \in C$ and T_{ij} the (i, j)-component of T. Note that $Q_j \in V_Q{}^*$ and $\mathrm{Im}Q \subseteq V_Q{}^*$.

The intuitive meaning of this function can be explained as follows. Take a basis of W and augment it to a basis of V_Q. Let H denote the index set for the basis of W. Then re-express the matrix Q. Define

$$\Omega = \{j \in J \mid Q_j \in W^\perp,\ T[I, j] = 0\},$$

[1] This setting might seem strange because the column vectors belong to the "dual" space $V_Q{}^*$. However, we adopt this convention since the "primal" space V_Q is the linear space on which the transpose of Q acts, and we introduce a submodular function q by implicitly considering the transpose of A.

and further put

$$K = J - \Omega, \quad R'_Q = R_Q - H, \quad R'_T = R_T - I.$$

Then we have the form

$$\widehat{A}[R, J] = \begin{pmatrix} \widehat{Q}[R_Q, J] \\ T[R_T, J] \end{pmatrix} = \begin{matrix} \\ R'_Q \\ H \\ R'_T \\ I \end{matrix} \begin{matrix} \begin{matrix} \Omega & K \end{matrix} \\ \begin{pmatrix} \widehat{Q}[R'_Q, C'] & \widehat{Q}[R'_Q, K] \\ O & \widehat{Q}[H, K] \\ T[R'_T, C'] & T[R'_T, K] \\ O & T[I, K] \end{pmatrix} \end{matrix},$$

from which it is clear that $\kappa((W, I), J)$ coincides with the cardinality of K.

Put

$$q((W, I), J) = \kappa((W, I), J) - \dim W - |I|.$$

We now have the following lemma.

LEMMA 4.1. *The function $q : \mathcal{L} \times 2^C \to Z$ is bisubmodular, i.e.,*

$$q(X_1, J_1) + q(X_2, J_2) \geq q(X_1 \vee X_2, J_1 \cap J_2) + q(X_1 \wedge X_2, J_1 \cup J_2), \\ X_h \in \mathcal{L}, \quad J_h \subseteq C \quad (h = 1, 2).$$

Proof. Put $\omega((W, I), J) = |\Omega((W, I), J)|$, where

$$\Omega((W, I), J) = \{j \in J \mid Q_j \in W^{\perp}, \ T[I, j] = 0\}.$$

Since we have

$$q((W, I), J) = |J| - |\Omega((W, I), J)| - \dim W - |I|, \quad (W, I) \in \mathcal{L}, \quad J \subseteq C,$$

it suffices to show the bisupermodularity of ω as

$$\omega(X_1, J_1) + \omega(X_2, J_2) \leq \omega(X_1 \vee X_2, J_1 \cap J_2) + \omega(X_1 \wedge X_2, J_1 \cup J_2), \\ X_h \in \mathcal{L}, \quad J_h \subseteq C \quad (h = 1, 2).$$

Noting that

$$\Omega(X_1, J_1) \cap \Omega(X_2, J_2) = \Omega(X_1 \vee X_2, J_1 \cap J_2)$$

and

$$\Omega(X_1, J_1) \cup \Omega(X_2, J_2) \subseteq \Omega(X_1 \wedge X_2, J_1 \cup J_2),$$

we obtain

$$|\Omega(X_1, J_1)| + |\Omega(X_2, J_2)| = |\Omega(X_1, J_1) \cap \Omega(X_2, J_2)| + |\Omega(X_1, J_1) \cup \Omega(X_2, J_2)| \\ \leq |\Omega(X_1 \vee X_2, J_1 \cap J_2)| + |\Omega(X_1 \wedge X_2, J_1 \cup J_2)|,$$

which establishes the bisupermodularity of ω. □

Fix $J \subseteq C$ and put $q_J(\,\cdot\,) = q(\,\cdot\,, J)$; then q_J is a submodular function on $\mathcal{L}$. It may be remarked that the function q_C agrees with the submodular function defined

in [8] associated with the transpose of A when it is considered as a partitioned matrix. In this context it should be clear that the transpose of an LM-matrix is no longer an LM-matrix, but it stays in the wider class of partitioned matrices.

As to the submodular function q_C, we have the following identity, which is quite similar to Lemma 2.1.

THEOREM 4.2. *For an $m \times n$ LM-matrix A,*

$$\mathrm{rank}A = \min\{q_C(X) \mid X \in \mathcal{L}\} + m.$$

Proof. For any $X = (W, I) \in \mathcal{L}$, A can be transformed, by taking a basis extended from W, to the form

$$\widehat{A} = \begin{pmatrix} \widehat{Q} \\ T \end{pmatrix} = \begin{array}{c} \\ R'_Q \\ H \\ R'_T \\ I \end{array} \begin{array}{c} \begin{array}{cc} C' & K \end{array} \\ \begin{pmatrix} \widehat{Q}[R'_Q, C'] & \widehat{Q}[R'_Q, K] \\ O & \widehat{Q}[H, K] \\ T[R'_T, C'] & T[R'_T, K] \\ O & T[I, K] \end{pmatrix} \end{array},$$

where $K = \{j \in C \mid \widehat{Q}_j \notin W^{\perp}$ or $\exists i \in I$ s.t. $T_{ij} \neq 0\}$, $C' = C - K$, $|H| = \dim W$, $R'_Q = R_Q - H$, and $R'_T = R_T - I$. Now we have the inequality

$$\mathrm{rank}A = \mathrm{rank}\widehat{A} \leq |R'_Q \cup R'_T| + |K| = q_C(X) + m$$

from the above form of $\widehat{A}$.

Considering the CCF of A with the notation of §2, put $W_\infty = \Upsilon(R_Q \cap R_\infty)$ and $I_\infty = R_T \cap R_\infty$. Since $\mathrm{rank}A = m - |R_\infty| + |C_\infty|$ from (2), $|C_\infty| = \kappa((W_\infty, I_\infty), C)$, and $\dim W_\infty + |I_\infty| = |R_\infty|$, we have

$$\mathrm{rank}A = q((W_\infty, I_\infty), C) + m.$$

Thus we have obtained $\mathrm{rank}A = \min\{q_C(X) \mid X \in \mathcal{L}\} + m$. □

As is evident from the proof of Theorem 4.2, the CCF $\widetilde{A}$ of an LM-matrix A yields a distributive sublattice of $\mathcal{L}$. To be more precise, put

$$\mathcal{L}_{CCF} = \{(W, I) \in \mathcal{L} \mid W = \Upsilon(R_Q - \alpha(C_*)),\ I = R_T - \alpha(C_*),\ C_* \in \mathcal{D}(\widetilde{A})\},$$

where $\alpha(\,\cdot\,)$ denotes the natural correspondence from the blocks $(C_0; C_1, \ldots, C_b; C_\infty)$ to $(R_0; R_1, \ldots, R_b; R_\infty)$, i.e., $\alpha(C_0) = R_0$, $\alpha(C_0 \cup C_1) = R_0 \cup R_1$, and so on, and where $\mathcal{D}(\widetilde{A})$ is defined by (3). Because of the uniqueness of the CCF, $\mathcal{L}_{CCF}$ is determined uniquely from A. Now $\mathcal{L}_{CCF}$ is a distributive sublattice of $\mathcal{L}$, since it is isomorphic to $\mathcal{D}(\widetilde{A})$ as the above expression of $\mathcal{L}_{CCF}$ shows. Note that $\mathcal{L}_{CCF}$ agrees with the set of minimizers of the submodular function q_C.

We denote by $\mathcal{K}_{PS}$ and $\mathcal{L}_{PS}$, respectively, the principal structure and the principal sublattice of $(\mathcal{L}, q_C)$ in the sense of §3. Then $\mathcal{L}_{CCF} \subseteq \mathcal{K}_{PS}$, and in particular, $\mathcal{L}_{CCF} = \mathcal{K}_{PS} = \mathcal{L}_{PS}$ if $C_0 = \emptyset$, i.e., A is of column-full rank.

For $J \subseteq C$, consider the submatrix $A[R, J]$, which is also an LM-matrix. As we define $\mathcal{L}_{CCF}$ from A, we can define a sublattice of $\mathcal{L}$ from the submatrix $A[R, J]$, which will be denoted by $\mathcal{L}_{CCF}(J)$. Put

$$\mathcal{B} = \{B \subseteq C \mid \mathrm{rank}A = \mathrm{rank}A[R, B] = |B|\}.$$

That is, $\mathcal{B}$ is the base family of the matroid $M(A)$ that represents the linear dependency among the column vectors of A.

Now we are ready to state the main result of this paper, the proof of which is postponed to §5.

THEOREM 4.3. *For an LM-matrix A,*

$$\mathcal{K}_{PS} = \bigcup_{B \in \mathcal{B}} \mathcal{L}_{CCF}(B).$$

Let Λ be the family of all the sublattices of $\mathcal{L}$ augmented by the empty set $\emptyset$. As is well known, Λ is a lattice whose minimum element is $\emptyset$ and maximum element is $\mathcal{L}$ itself. Then $\mathcal{L}_{PS} \in \Lambda$ and $\mathcal{L}_{CCF}(B) \in \Lambda$. We have the following corollary to Theorem 4.3.

COROLLARY 4.4. *For an LM-matrix A,*

$$\mathcal{L}_{PS} = \bigvee_{B \in \mathcal{B}} \mathcal{L}_{CCF}(B),$$

where $\bigvee$ designates the join operation in the lattice Λ.

Example 4.1. Consider the LM-matrix in Example 2.1. We denote by V_1 and V_2 the one-dimensional vector spaces spanned by $\begin{pmatrix} 0 & 1 \end{pmatrix} \in V_Q$ and $\begin{pmatrix} 2 & -1 \end{pmatrix} \in V_Q$, respectively.

As is easily verified, we have

$$\mathcal{K}(q_C) = \{(0, \emptyset), (V_1, \emptyset), (V_2, \emptyset), (0, Z_1), (V_2, Z_1), (V_2, R_T), (V_Q, R_T)\},$$

where $Z_1 = \{z_1\}$ and $Z_2 = \{z_2\}$. The principal structure $\mathcal{K}_{PS} = \mathcal{K}(q_C)$ and the principal sublattice $\mathcal{L}_{PS} = \mathcal{L}(q_C)$ are illustrated in Fig. 2.

On the other hand, we have $\mathcal{B} = \{B_h = C - \{x_h\} \mid h = 1, \ldots, 5\}$. Figure 3 illustrates the sublattices $\mathcal{L}_{CCF}(B_h)$ for $h = 1, \ldots, 5$, which correspond to the CCFs $\widetilde{A}_h$ for $h = 1, \ldots, 5$ given in Example 2.1. For example, the height of $\mathcal{L}_{CCF}(B_5)$ is four since A_5 is in a block-triangular form with four diagonal blocks, and a parallelogram appears in $\mathcal{L}_{CCF}(B_5)$ since the $(3,4)$-component of A_5 is zero. We can easily observe that $\mathcal{K}_{PS}$ agrees with $\bigcup_{h=1}^{5} \mathcal{L}_{CCF}(B_h)$. Furthermore $\mathcal{L}_{PS} \neq \mathcal{L}_{CCF}(B)$ for any $B \in \mathcal{B}$.

Remark 4.1. For an LM-matrix A of rank r, what is the coarsest simultaneous decomposition of the row and the column sides that is finer than any decomposition induced by the CCF of an r-dimensional nonsingular submatrix of A? Theorem 4.3 (or Corollary 4.4), together with the main theorem of [16], give a solution to this question. All the diagonal blocks of the CCF of A should remain in the CCF of an r-dimensional nonsingular submatrix. The refinement of the horizontal tail of the CCF of A is given by the principal structure of q, while the principal structure of p gives the refinement of the vertical tail. This is the reason why we name the former "horizontal principal structure" and the latter "vertical principal structure" in this paper.

5. Proof. Consider a submatrix $A[R, B]$ for $B \in \mathcal{B}$. Since $A[R, B]$ is of column-full rank, having no horizontal tail in its CCF, the principal structure $\mathcal{K}(q_B)$ coincides with the distributive lattice induced by the CCF of $A[R, B]$, i.e.,

$$\mathcal{K}(q_B) = \mathcal{L}_{CCF}(B). \tag{4}$$

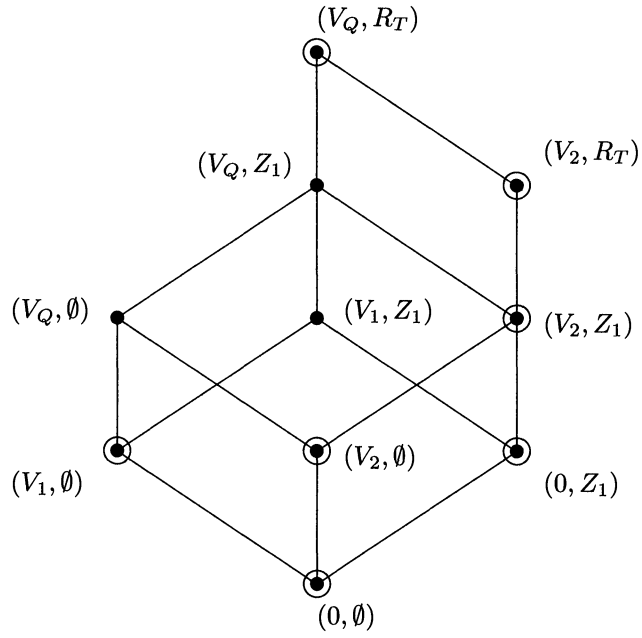


FIG. 2. *The Hasse diagram for the principal sublattice $\mathcal{L}_{PS}$ in Example* 4.1. *The double circles correspond to the elements of $\mathcal{K}_{PS}$.*

Hence, in order to prove Theorem 4.3, we shall reveal the relation between $\mathcal{K}(q_C)$ and $\mathcal{K}(q_B)$, i.e., the relation between $D(q_C; X)$ and $D(q_B; X)$.

LEMMA 5.1. *For any $X \in \mathcal{L}$ and any $J \subseteq C$,*

$$D(q_C; X) \preceq D(q_J; X).$$

Proof. The following argument is the same as that in [16]. Fix $X \in \mathcal{L}$ and put $D_C = D(q_C; X)$ and $D_J = D(q_J; X)$ for notational simplicity. To establish $D_C \preceq D_J$, it suffices to show

$$q_C(D_C) \geq q_C(D_C \wedge D_J).$$

By the bisubmodularity of q, we have

$$q(D_C, C) - q(D_C \wedge D_J, C) \geq q(D_C \vee D_J, J) - q(D_J, J).$$

That is,

$$q_C(D_C) - q_C(D_C \wedge D_J) \geq q_J(D_C \vee D_J) - q_J(D_J),$$

where the right-hand side must be nonnegative since $X \preceq D_C \vee D_J$. □

LEMMA 5.2. *For any $X \in \mathcal{L}$, there exists $B \in \mathcal{B}$ such that*

$$D(q_C; X) = D(q_B; X).$$

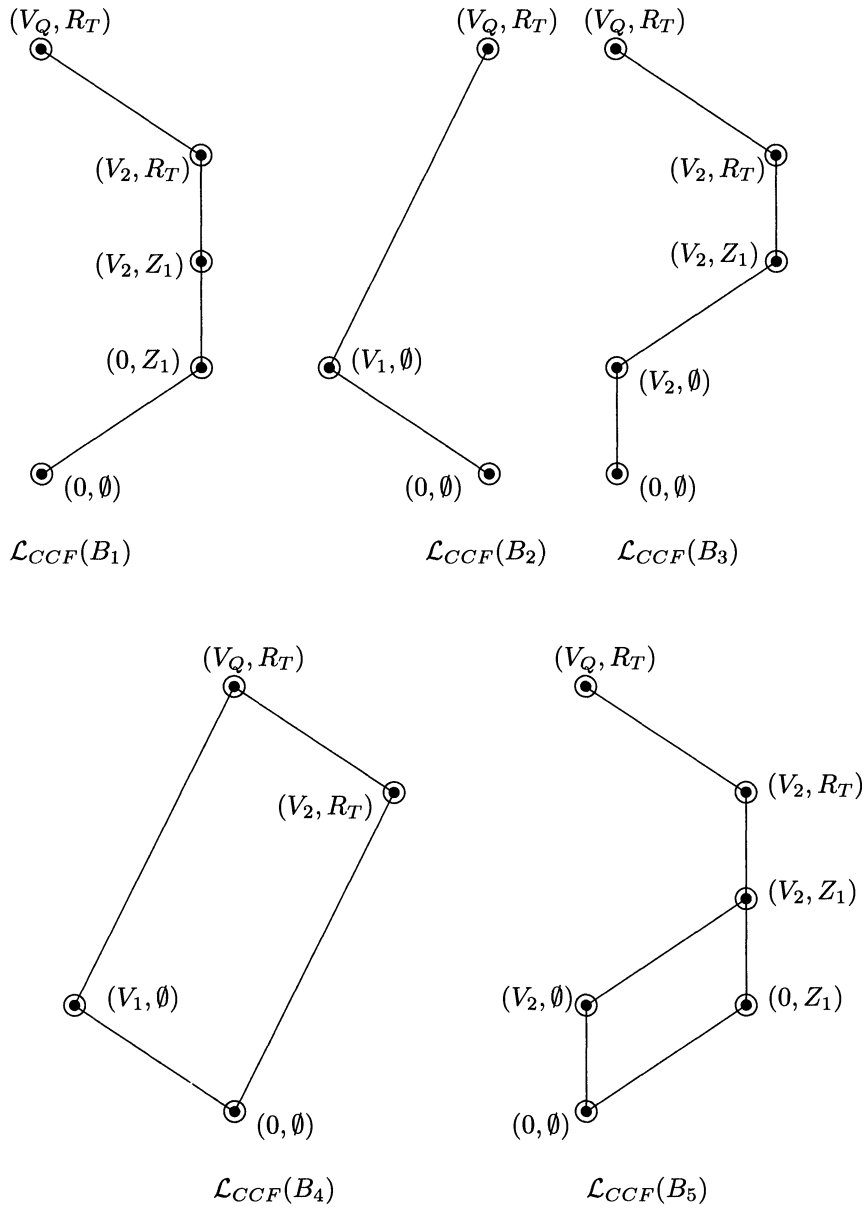


FIG. 3. *The Hasse diagrams for* $\mathcal{L}_{CCF}(B_h)$*'s in Example* 4.1.

Proof. Fix $X \in \mathcal{L}$ and put $D_C = (W_C, I_C) = D(q_C; X)$ for notational simplicity. By taking a basis in V_Q extended from W_C, A can be transformed to the form

$$\widehat{A} = \begin{pmatrix} \widehat{Q} \\ T \end{pmatrix} = \begin{array}{c} \\ R'_Q \\ H \\ R'_T \\ I_C \end{array} \begin{array}{c} \begin{array}{cc} C' & K \end{array} \\ \begin{pmatrix} \widehat{Q}[R'_Q, C'] & \widehat{Q}[R'_Q, K] \\ O & \widehat{Q}[H, K] \\ T[R'_T, C'] & T[R'_T, K] \\ O & T[I_C, K] \end{pmatrix} \end{array},$$

where $K = \{j \in C \mid \widehat{Q}_j \notin W_C{}^{\perp}$ or $\exists i \in I_C$ s.t. $T_{ij} \neq 0\}$, $C' = C - K$, $|H| = \dim W_C$, $R'_Q = R_Q - H$, and $R'_T = R_T - I_C$.

Put $R' = R'_Q \cup R'_T$ and we will claim that the submatrix $\widehat{A}[R', C']$ is of row-full rank, i.e.,

$$\mathrm{rank}\widehat{A}[R', C'] = |R'|. \tag{5}$$

Since $\widehat{A}[H \cup I_C, C'] = O$ and the rank of an LM-matrix is invariant under admissible transformations, it holds that

$$\mathrm{rank}\widehat{A}[R', C'] = \mathrm{rank}\widehat{A}[R, C'] = \mathrm{rank}A[R, C']. \tag{6}$$

Applying Theorem 4.2 to the LM-matrix $A[R, C']$, we have from $Q_j \in W_C{}^{\perp}$ and $T[I_C, j] = 0$ for all $j \in C'$ that

$$\begin{aligned} \mathrm{rank}A[R, C'] &= \min\{q((W, I), C') \mid (W, I) \in \mathcal{L}\} + m \\ &= \min\{q((W, I), C') \mid W \supseteq W_C,\ I \supseteq I_C\} + m. \end{aligned} \tag{7}$$

Assume that $W \supseteq W_C$ and $I \supseteq I_C$; then we have

$$\begin{aligned} \kappa((W, I), C') &= \kappa((W, I), C) - |K| \\ &= \kappa((W, I), C) - \kappa((W_C, I_C), C). \end{aligned}$$

Hence it holds that

$$\begin{aligned} q((W, I), C') &= \kappa((W, I), C') - |I| - \dim W \\ &= q((W, I), C) - q((W_C, I_C), C) - |I_C| - \dim W_C. \end{aligned} \tag{8}$$

It follows from the definition of $D_C = (W_C, I_C)$ and $(W, I) \succeq X$ that

$$q_C(W, I) - q_C(W_C, I_C) \geq 0. \tag{9}$$

Combining (6), (7), (8), (9), and $m - |I_C| - \dim W_C = |R'|$, we obtain (5).

Therefore there exists $J' \subseteq C'$ such that

$$\mathrm{rank}\widehat{A}[R', J'] = |R'| = |J'|.$$

At the same time, there exists $J_K \subseteq K$ such that

$$\mathrm{rank}\widehat{A}[H \cup I_C, K] = \mathrm{rank}\widehat{A}[H \cup I_C, J_K] = |J_K|.$$

Put $B = J' \cup J_K$. Since both $\widehat{A}[R', J']$ and $\widehat{A}[H \cup I_C, J_K]$ are of column-full rank,

$$\widehat{A}[R, B] = \begin{matrix} \\ R' \\ H \cup I_C \end{matrix} \begin{matrix} \begin{matrix} J' & J_K \end{matrix} \\ \begin{pmatrix} \widehat{A}[R', J'] & \widehat{A}[R', J_K] \\ O & \widehat{A}[H \cup I_C, J_K] \end{pmatrix} \end{matrix}$$

is of column-full rank, i.e.,

$$\mathrm{rank}\widehat{A}[R, B] = |B|. \tag{10}$$

On the other hand, since $\widehat{A}[R', C']$ is of row-full rank,

$$\mathrm{rank}\widehat{A} = \mathrm{rank}\widehat{A}[R', C'] + \mathrm{rank}\widehat{A}[H \cup I_C, K] = |J'| + |J_K| = |B|.$$

Thus we obtain $B \in \mathcal{B}$.

Applying Theorem 4.2 to the LM-matrix $A[R, B]$, we have from $\mathrm{rank}A[R, B] = \mathrm{rank}\widehat{A}[R, B]$ and (10) that

$$\min\{q_B(Y) \mid Y \in \mathcal{L}\} = |B| - m,$$

which together with $q_B(D_C) = |J_K| - |I_C| - \dim W_C = |B| - m$ and $X \preceq D_C$ implies

$$q_B(D_C) = \min\{q_B(Y) \mid X \preceq Y \in \mathcal{L}\}.$$

Thus we obtain $D_C \succeq D(q_B; X)$, which completes the proof since we have already shown $D_C \preceq D(q_B; X)$ in Lemma 5.1. □

Now we are ready to prove Theorem 4.3. It follows from Lemma 5.1 that $D(q_J; X) = X$ implies $D(q_C; X) = X$. Hence,

$$\mathcal{K}(q_J) \subseteq \mathcal{K}(q_C)$$

holds for any $J \subseteq C$. On the other hand, from Lemma 5.2, $X = D(q_C; X)$ implies the existence of $B \in \mathcal{B}$ such that $X = D(q_B; X)$. That is to say,

$$\mathcal{K}(q_C) \subseteq \bigcup_{B \in \mathcal{B}} \mathcal{K}(q_B).$$

Thus we have $\mathcal{K}(q_C) = \bigcup_{B \in \mathcal{B}} \mathcal{K}(q_B)$, which establishes Theorem 4.3 when combined with (4).

6. Conclusion. It is shown that the principal structure of the submodular function q defined on a modular lattice associated with an LM-matrix gives the coarsest decomposition of the row side that is finer than any decomposition induced by the CCF of the submatrix consisting of a base of the column vectors of A. This gives a best possible bound on the extent to which the whole system described by an LM-matrix can be decomposed by a suitable choice of design variables. This result can be extended to multilayered matrices [18] under a certain genericity assumption.

Computing the principal structure of the submodular function q_C involves the minimization of submodular functions on nondistributive modular lattices and compact representations of them. Unfortunately, little is known on this matter. However, the projection of the principal sublattice $\mathcal{L}(q_C)$ to the Boolean lattice 2^{R_T} can be computed efficiently. Although this projection has only partial information on the horizontal principal structure of the LM-matrix, it is still interesting because it gives the best possible upper bound on the decompositions of R_T, which corresponds to the set of physical elements of the system in question.

A unification of horizontal and vertical principal structure is given in [10] in terms of independent matchings, and a further extension is presented in [9].

Acknowledgments. The authors are grateful to Professor Satoru Fujishige of the University of Tsukuba for the relevant reference [20] and to an anonymous referee for comments on presentation.

REFERENCES

[1] M. AIGNER, *Combinatorial Theory*, Springer-Verlag, Berlin, 1979.
[2] R. A. BRUALDI AND H. J. RYSER, *Combinatorial Matrix Theory*, Cambridge University Press, London, 1991.

[3] S. F. CHANG AND S. T. MCCORMICK, *A hierarchical algorithm for making sparse matrices sparser*, Math. Programming, 56 (1992), pp. 1–30.
[4] A. L. DULMAGE AND N. S. MENDELSOHN, *A structure theory of bipartite graphs of finite exterior dimension*, Trans. Roy. Soc. Canada, 53 (1959), pp. 1–13.
[5] S. FUJISHIGE, *Principal structures of submodular systems*, Discrete Appl. Math., 2 (1980), pp. 77–79.
[6] ———, *Submodular Functions and Optimization*, North-Holland, Amsterdam, 1991.
[7] M. IRI, *Applications of matroid theory*, Mathematical Programming—The State of the Art, A. Bachem, M. Grötschel, and B. Korte, eds., Springer-Verlag, Berlin, 1983, pp. 158–201.
[8] H. ITO, S. IWATA, AND K. MUROTA, *Block-triangularizations of partitioned matrices under similarity/equivalence transformations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1226–1255.
[9] S. IWATA, *Principal structure of submodular systems and Hitchcock-type independent flows*, Combinatorica, 15 (1995).
[10] S. IWATA AND K. MUROTA, *A theorem on the principal structure for independent matchings*, Discrete Appl. Math., 61 (1995), pp. 229–244.
[11] L. LOVÁSZ AND M. PLUMMER, *Matching Theory*, North-Holland, Amsterdam, 1986.
[12] S. T. MCCORMICK, *A Combinatorial Approach to Some Sparse Matrix Problems*, Technical Report SOL 83–5, Department of Operations Research, Stanford University, 1983.
[13] S. T. MCCORMICK AND S. F. CHANG, *The weighted sparsity problem: complexity and algorithms*, SIAM J. Discrete Math., 6 (1993), pp. 57–69.
[14] K. MUROTA, *Systems Analysis by Graphs and Matroids—Structural Solvability and Controllability*, Springer-Verlag, Berlin, 1987.
[15] ———, *Some recent results in combinatorial approaches to dynamical systems*, Linear Algebra Appl., 122/123/124 (1989), pp. 725–759.
[16] ———, *Principal structure of layered mixed matrices*, Discrete Appl. Math., 27 (1990), pp. 221–234.
[17] ———, *Mixed matrices—Irreducibility and decomposition*, Combinatorial and Graph-Theoretical Problems in Linear Algebra, R. A. Brualdi, S. Friedland, and V. Klee, eds., The IMA Volumes in Mathematics and Its Applications, Springer-Verlag, New York, 1993, pp. 39–71.
[18] K. MUROTA, M. IRI, AND M. NAKAMURA, *Combinatorial canonical form of layered mixed matrices and its application to block-triangularization of systems of equations*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 123–149.
[19] A. RECSKI, *Matroid Theory and Its Applications in Electric Network Theory and in Statics*, Springer-Verlag, Berlin, 1989.
[20] N. TOMIZAWA AND S. FUJISHIGE, *Theory of hyperspace (XIV)—Principal decompositions and principal structures of metric lattices with respect to supermodular functions*, CAS 82-2, Institute of Electronics and Communication Engineers of Japan, 1982. (In Japanese.)

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 87–100, February 1996

009

DATA SECURITY EQUALS GRAPH CONNECTIVITY*

MING-YANG KAO†

Abstract. To protect sensitive information in a cross-tabulated table, it is a common practice to suppress some of the cells in the table. This paper investigates four levels of data security of a two-dimensional table concerning the effectiveness of this practice. These four levels of data security protect the information contained in, respectively, individual cells, individual rows and columns, several rows or columns as a whole, and a table as a whole. The paper presents efficient algorithms and NP-completeness results for testing and achieving these four levels of data security. All these complexity results are obtained by means of fundamental equivalences between the four levels of data security of a table and four types of connectivity of a graph constructed from that table.

Key words. statistical tables, linear algebra, graph theory, mixed graphs, strong connectivity, bipartite-$(k+1)$-connectivity, bipartite completeness

AMS subject classifications. 68Q22, 62A99, 05C99

1. Introduction.

Cross-tabulated tables are used in a wide variety of documents to organize and exhibit information. The values of sensitive cells in such tables are routinely suppressed to conceal sensitive information. There are two fundamental issues concerning the effectiveness of this practice [1], [4], [5], [6], [7], [8], [9], [20], [21], [22], [23]. One is whether an adversary can deduce significant information about the suppressed cells from the published data of a table. The other is how a table maker can suppress a small number of cells in addition to the sensitive ones so that the resulting table does not leak significant information.

This paper investigates how to protect the information in a two-dimensional table that publishes three types of data (see [16] for examples): (1) the values of all cells except a set of sensitive ones, which are *suppressed*, (2) an upper bound and a lower bound for each cell, and (3) all row sums and column sums of the complete set of cells. The cells may have real or integer values. They may have different bounds, and the bounds may be finite or infinite. The upper bound of a cell should be strictly greater than its lower bound; otherwise, the value of that cell is immediately known even if that cell is suppressed. The cells that are not suppressed also have upper and lower bounds. These bounds are necessary because some of the unsuppressed cells may later be suppressed to protect the information in the sensitive cells.

The focus of this paper is how to protect the type of information defined here. A *bounded feasible assignment* to a table is an assignment of values to the suppressed cells such that each row or column adds up to its published sum and the bounds of the suppressed cells are all satisfied. A linear combination of the suppressed cells is a *linear invariant* if it has the same value at all bounded feasible assignments (see [16] for examples). Intuitively, the information contained in a linear invariant is unprotected because its value can be uniquely deduced from the published data. Five classes of linear invariants are of special significance. A *positive* invariant is one whose coefficients are all nonnegative with at least one positive coefficient. A

* Received by the editors January 21, 1993; accepted for publication (in revised form) February 22, 1995. A preliminary version of this work appeared in *Proc. 2nd International Workshop on Discrete Mathematics and Algorithms*, Guanzhou, China, December 18–20, 1994, pp. 134–147.

† Department of Computer Science, Duke University, Durham, NC 27708 (kao@cs.duke.edu). This research was supported in part by National Science Foundation grants MCS-8116678, DCR-8405478, and CCR-9101385. Part of this research was performed while the author was at the Department of Computer Science, Yale University, New Haven, CT 06520.

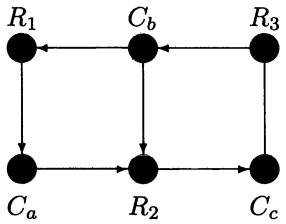
unitary invariant is one whose coefficients are +1, 0, or −1. A *sum* invariant is one whose coefficients are +1 or 0. A *rectangular* sum invariant is one that sums over all suppressed cells shared by a set of rows and a set of columns. An *invariant cell* is a suppressed cell that forms a linear invariant all by itself.

Four levels of data security of a table are discussed in this paper. To motivate the discussion, suppose that a given table tabulates the quantities of several products made by different factories. A row represents a factory, a column records the quantities of a product, and a cell contains the quantity of a product made by a factory. Level 1 protects the suppressed cells individually. A factory wishes to conceal the quantity of a particular product. Naturally, that quantity should be suppressed and its precise value should not be uniquely determined from the published data of the table. Thus, a suppressed cell is *protected* if it is not an invariant cell [12]. Level 2 protects a row (or column) as a whole. After all suppressed cells are protected, an adversary may still be able to obtain useful information by combining the suppressed cells. If a factory wishes to protect the information about the quantities of all its products as a whole, it must ensure that no information of a sensitive type can be extracted by combining the suppressed cells in the row representing that factory. Hence, a row is *protected* if there is no linear invariant of a desired type that combines the suppressed cells in that row. Level 3 protects a set of k rows (or k columns) as a whole. Suppose that a company owns k factories. It wishes to conceal aggregate information about all its factories, not just the information about each individual factory. It should require that no information of an important type can be derived by combining the suppressed cells in the k rows for its k factories. Thus, a set of k rows is *protected* if there is no linear invariant of a desired type that combines the suppressed cells in those rows. Level 4 protects the given table as a whole. Further suppose that the above company owns all the factories tabulated in the table. It wishes to protect aggregate information about all its factories and all their products. It stipulates that only trivial information may be found among a desired class of combinations of the suppressed cells. Thus, a table is *protected* if it has no linear invariant of a desired type that combines its suppressed cells.

The key contribution of this paper is to establish that the latter three levels of data security of a table are equivalent to three types of connectivity of a graph called the *suppressed graph* of that table. Previously, Gusfield showed that the first level of data security is equivalent to a certain type of connectivity of the suppressed graph [12]. The paper further uses these fundamental equivalences to obtain three sets of complexity results. First, the second and fourth levels of data security of a table can be tested in optimal linear time and the third level can be tested in polynomial time. Previously, Gusfield showed how to find all invariant cells of a table and test for its first level of data security in optimal linear time [12]. Second, for each of the four levels of data security, it is an NP-complete problem to compute and suppress the minimum number of additional cells in a table in order to achieve the desired level of data security. Third, for a large and practical class of tables, the above optimal suppression problem for the second and fourth levels of data security can be solved in optimal linear time. For the first level of data security, Gusfield showed that the optimal suppression problem can be solved in optimal linear time [11]. For the third level of data security the optimal suppression problem remains open.

We review basics of graphs and tables in §2, discuss the four levels of data security in §3 through §5, and compare them in §6.

row column index	a	b	c	row sum
1	[0]	[9]	1	10
2	[9]	[9]	[0]	18
3	6	[0]	[5]	11
column sum	15	18	6	



The number in a table cell is its value. A cell with a box is suppressed. The lower and upper bounds of the cells are 0 and 9. The graph is the suppressed graph of the table. Vertex R_p corresponds to row p and vertex C_q to column q.

FIG. 1. *A table and its suppressed graph.*

2. Preliminaries. Every graph in this paper is a *mixed* graph, i.e., it may contain both undirected and directed edges with at most one edge between two vertices. Let $\mathcal{T}$ be a table. The *suppressed graph* $\mathcal{H} = (A, B, E)$ and the *total graph* $\mathcal{H}' = (A, B, E')$ of $\mathcal{T}$ are the bipartite graphs constructed here (see Fig. 1 for an example). For each row (respectively, column) of $\mathcal{T}$, there is a vertex in A (respectively, B); this vertex is called a *row* (respectively, *column*) vertex. For each cell x at row i and column j, there is an edge $e \in E'$ between the vertices of row i and column j. If the value of x is strictly between its lower and upper bounds, then e is undirected. Otherwise, if the value equals the lower (respectively, upper) bound, then e points from the row endpoint to the column endpoint (respectively, from the column endpoint to the row endpoint). E consists of the edges corresponding to the suppressed cells. Note that $\mathcal{H}$ is a subgraph of $\mathcal{H}'$ and $\mathcal{H}'$ is *complete* (i.e., for all $u \in A$ and $v \in B$, E' has exactly one edge between u and v). Also, given an arbitrary complete bipartite graph and a subgraph on the same vertices, it takes only linear time to construct a table with these two graphs as its total and suppressed graphs.

A *traversable* cycle or path is one that can be traversed along its edge directions. A *direction-blind* cycle or path is one that can be traversed if its edge directions are disregarded; we often omit the word direction-blind for brevity. A graph is *connected* if each pair of vertices are in a path. A *connected component* is a maximal connected subgraph. A *nonsingleton* connected component is one with two or more vertices. A graph is *strongly connected* if each pair of vertices are in a traversable cycle. A *strong component* is a maximal strongly connected subgraph.

The *effective area* of a linear invariant F of $\mathcal{T}$, denoted by $EA(F)$, is the set of

suppressed cells in the nonzero terms of F. $EA(F)$ is also regarded as a set of edges in $\mathcal{H}$. F is *nonzero* if $EA(F) \neq \emptyset$. F is *minimal* if it is nonzero and $\mathcal{T}$ has no nonzero linear invariant whose effective area is a proper subset of $EA(F)$. Note that given a minimal linear invariant F, if F' is a nonzero linear invariant with $EA(F') \subseteq EA(F)$, then F' is also minimal and is a multiple of F. Thus, a minimal linear invariant is unique up to a multiplicative factor with respect to its effective area.

An edge set of a graph is an *edge cut* if its removal disconnects a connected component. An edge cut is *minimal* if no proper subset of it is an edge cut.

FACT 1. *Let Z be an edge set of a strong component $\mathcal{H}'$ of $\mathcal{H}$. Z is a minimal edge cut of $\mathcal{H}'$ if and only if $\mathcal{H}' - Z$ has exactly two connected components, say, $\mathcal{H}_1$ and $\mathcal{H}_2$, and each edge of Z is between $\mathcal{H}_1$ and $\mathcal{H}_2$.*

Assume that Z is a minimal edge cut of $\mathcal{H}'$. Z is *bipartite* if the endpoints of Z in $\mathcal{H}_1$ are all row vertices or all column vertices. An edge set of $\mathcal{H}$ is a (respectively, *bipartite*) *basic set* if it consists of an edge not in any strong component of $\mathcal{H}$ or is a (respectively, bipartite) minimal edge cut of some strong component.

THEOREM 2.1 (see [14]). 1. *A linear invariant of $\mathcal{T}$ is minimal if and only if its effective area is a basic set of $\mathcal{H}$. Also, for each basic set Z of $\mathcal{H}$, there is a minimal linear invariant F of $\mathcal{T}$ with $EA(F) = Z$.*

2. *Every minimal linear invariant is a multiple of a unitary invariant. Furthermore, a minimal linear invariant F of $\mathcal{T}$ is a multiple of a sum invariant if and only if $EA(F)$ is a bipartite basic set of $\mathcal{H}$.*
3. *For each nonzero linear invariant F of $\mathcal{T}$, there exist unitary minimal linear invariants $F_1, \ldots, F_k$ of $\mathcal{T}$ such that $F = \sum_{i=1}^{k} c_i \cdot F_i$ for some $c_i > 0$, $EA(F) = \cup_{i=1}^{k} EA(F_i)$, and for each F_i and each $e \in EA(F_i)$, the coefficients of e in F and F_i are either both positive or both negative.*

Remark. A referee has indicated that a different proof for Theorem 2.1 from that in [14] can be constructed by means of conformal vector decomposition [18], [19].

3. Protection of a cell. A suppressed cell of $\mathcal{T}$ is *protected* if it is not an invariant cell.

3.1. Cell protection and bridge-freeness. A graph is *bridge free* if it has no edge cut consisting of a single edge.

THEOREM 3.1 (see [12]). 1. *A suppressed cell of $\mathcal{T}$ is protected if and only if it is an edge in an edge-simple traversable cycle of $\mathcal{H}$.*

2. *The suppressed cells of $\mathcal{T}$ are all protected if and only if each connected component of $\mathcal{H}$ is strongly connected and bridge free.*

COROLLARY 3.2 (see [12]). *Given $\mathcal{H}$, the unprotected cells of $\mathcal{T}$ can be found in $O(|\mathcal{H}|)$ time.*

3.2. Optimal suppression problems for cell protection. The problem below is concerned with suppressing the minimum number of additional cells in $\mathcal{T}$ such that the original and the new suppressed cells in the resulting table are all protected.

PROBLEM 1 (protection of all cells).

- Input: $\mathcal{T}$ and an integer $p \geq 0$.
- Output: Is there a set P consisting of at most p published cells of $\mathcal{T}$ such that all suppressed cells are protected in the table formed by $\mathcal{T}$ with the cells in P also suppressed?

Problem 1 can be reformulated as the graph augmentation problem below.

PROBLEM 2.

- Input: A complete bipartite graph $\mathcal{H}'$, a subgraph $\mathcal{H}$, and an integer $p \geq 0$.

- Output: Is there a set P of at most p edges in $\mathcal{H}' - \mathcal{H}$ such that each connected component of $\mathcal{H} \cup P$ is strongly connected and bridge free?

LEMMA 3.3. *Problems* 1 *and* 2 *can be reduced to each other in linear time.*

Proof. The proof follows from Theorem 3.1(2). □

The next problem is NP-complete [10]. It is used here to prove that Problems 1 and 2 are hard.

PROBLEM 3 (hitting set).

- Input: A finite set S, a nonempty set $W \subseteq 2^S$, and an integer $h \geq 0$.
- Output: Is there a subset S' of S such that $|S'| \leq h$ and S' contains at least one element in each set in W?

THEOREM 3.4. *Problems* 1 *and* 2 *are NP-complete.*

Proof. Problems 1 and 2 are both in NP. To prove their completeness, by Lemma 3.3, it suffices to reduce Problem 3 to Problem 2.

Given an instance $S = \{s_1, \ldots, s_\alpha\}$, $W = \{S_1, \ldots, S_\beta\}$, h of Problem 3, an instance $\mathcal{H}' = (A, B, E'), \mathcal{H} = (A, B, E), p$ of Problem 2 is constructed as follows:

- Rule 1: Let $A = \{a_0, a_1, \ldots, a_\alpha\}$. The vertices $a_1, \ldots, a_\alpha$ correspond to $s_1, \ldots, s_\alpha$, but a_0 corresponds to no s_i.
- Rule 2: Let $B = \{b_0, b_1, \ldots, b_\beta\}$. The vertices $b_1, \ldots, b_\beta$ correspond to $S_1, \ldots, S_\beta$ of S, but b_0 corresponds to no S_j.
- Rule 3: Let E' consist of the following edges:
  1. The edge between a_0 and b_0 is $b_0 \to a_0$.
  2. For all j with $1 \leq j \leq \beta$, the edge between a_0 and b_j is $a_0 \to b_j$.
  3. For all i with $1 \leq i \leq \alpha$, the edge between a_i and b_0 is $a_i \to b_0$.
  4. For each s_i and each S_j, if $s_i \in S_j$, then the edge between a_i and b_j is $b_j \to a_i$; otherwise it is $a_i \to b_j$.
- Rule 4: Let $E = \{b_0 \to a_0\} \cup \{a_0 \to b_1, \ldots, a_0 \to b_\beta\}$.
- Rule 5: Let $p = h + \beta$.

The above construction can be easily computed in polynomial time. The next two claims show that it is indeed a desired reduction from Problem 3 to Problem 2.

CLAIM 1. *If some $S' \subseteq S$ with $|S'| \leq h$ has at least one element in each S_j, then some $P \subseteq E' - E$ consists of at most p edges such that every connected component of $\mathcal{H} \cup P$ is strongly connected and bridge free.*

To prove this claim, observe that for each S_j, some $s_{i_j} \in S' \cap S_j$ exists. By Rule 3(4), $P_1 = \{b_1 \to a_{i_1}, \ldots, b_\beta \to a_{i_\beta}\}$ exists. By Rule 3(3), $P_2 = \{a_{i_1} \to b_0, \ldots, a_{i_\beta} \to b_0\}$ exists. Let $P = P_1 \cup P_2$. Note that P_1 consists of β edges. P_2 consists of at most $|S'|$ edges. Thus P has at most $p = \beta + h$ edges. For all j with $1 \leq j \leq \beta$, the edges $b_0 \to a_0, a_0 \to b_j, b_j \to a_{i_j}$, and $a_{i_j} \to b_0$ form a vertex-simple traversable cycle. Because $E \cup P$ consists of the edges in these cycles, every connected component of $\mathcal{H} \cup P$ is strongly connected and bridge free. This finishes the proof of Claim 1.

CLAIM 2. *If some $P \subseteq E' - E$ consists of at most p edges such that every connected component of $\mathcal{H} \cup P$ is strongly connected and bridge free, then some $S' \subseteq S$ with $|S'| \leq h$ has at least one element in each S_j.*

To prove this claim, observe that for all j with $1 \leq j \leq \beta$, by Rule 4, E contains $a_0 \to b_j$ but no edge pointing from b_j. Because every connected component of $\mathcal{H} \cup P$ is strongly connected, P contains an edge $b_j \to a_{i_j}$ for some i_j. By Rule 3(4), $s_{i_j} \in S_j$. Let $S' = \{s_{i_1}, \ldots, s_{i_\beta}\}$. Note that P contains $b_1 \to a_{i_1}, \ldots, b_\beta \to a_{i_\beta}$ but E contains no edges pointing from $\{a_{i_1}, \ldots, a_{i_\beta}\}$. Because every connected component of $\mathcal{H} \cup P$ is strongly connected, P must also contain at least one edge pointing from each vertex in $\{a_{i_1}, \ldots, a_{i_\beta}\}$. Thus P contains at least $|S'| + \beta$ edges. Then $|S'| \leq h$ because

$|P| \leq \beta + h$. This finishes the proof of Claim 2 and thus that of Theorem 3.4. □

The next two problems are optimization versions of Problems 1 and 2 for undirected graphs and tables whose total graphs are undirected.

PROBLEM 4 (protection of all cells).

- Input: The suppressed graph of a table $\mathcal{T}$ whose total graph is undirected.
- Output: A set P consisting of the smallest number of published cells of $\mathcal{T}$ such that all suppressed cells are protected in the table formed by $\mathcal{T}$ with the cells in P also suppressed.

PROBLEM 5.

- Input: A bipartite undirected graph $\mathcal{H} = (A, B, E)$.
- Output: A set P consisting of the smallest number of undirected edges between A and B but not in E such that every connected component of $(A, B, E \cup P)$ is bridge free.

Note that Problem 5 need not specify $\mathcal{H}'$ because it is undirected and thus is unique for $\mathcal{H}$. Similarly, $\mathcal{H} \cup P$ is always strongly connected.

LEMMA 3.5. *Problems* 4 *and* 5 *can be reduced to each other in linear time.*

Proof. The proof is similar to that of Lemma 3.3. □

THEOREM 3.6 (see [11]). *Problem* 5 *is solvable in linear time*; *thus so is Problem* 4.

4. Protection of rows and columns. This section discusses the data security of a table at levels 2 and 3 in a unified framework. Let $EA(R)$ denote the set of suppressed cells in a row or column R. Let $\overline{R} = \sum_{e \in EA(R)} e$. Let $R_1, \ldots, R_k$ be k rows or k columns of $\mathcal{T}$, but no mixed case. For level 3 data security, $\{R_1, \ldots, R_k\}$ is *protected* with respect to the linear invariants (respectively, the positive invariants, the unitary invariants, the sum invariants, or the rectangular sum invariants) if the conditions below hold:

1. Each linear invariant (respectively, positive invariant, unitary invariant, sum invariant, or rectangular sum invariant) F of $\mathcal{T}$ with $EA(F) \subseteq \cup_{i=1}^{k} EA(R_i)$ is a linear combination of $\overline{R}_1, \ldots, \overline{R}_k$.
2. No suppressed cell of $R_1, \ldots, R_k$ is an invariant cell.

Level 2 data security is a special case of level 3 with $k = 1$, and its definitions can be simplified. A row or column R is *protected* with respect to the linear invariants (respectively, the positive invariants, the unitary invariants, or the sum invariants) if the conditions below hold:

1. Each linear invariant (respectively, positive invariant, unitary invariant, or sum invariant) F with $EA(F) \subseteq EA(R)$ is a multiple of $\overline{R}$.
2. No suppressed cell in R is an invariant cell.

We do not explicitly consider the protection of R with respect to the rectangular sum invariants because for $k = 1$ these invariants are the same as the sum invariants. Also, the five types of invariants here are implicitly considered for cell protection because a linear invariant with exactly one nonzero term is essentially an invariant cell.

The two conditions in the definitions are based on technical considerations. No matter how many cells in $\mathcal{T}$ are suppressed, $\overline{R}_1, \ldots, \overline{R}_k$ and their linear combinations are always linear invariants. Thus the first condition gives the best possible protection for $R_1, \ldots, R_k$ as a whole. If R_i has either no suppressed cell or at least two, the first condition implies the second one; otherwise, the first condition holds trivially but the only suppressed cell in R_i is an invariant. The second condition is adopted to avoid this undesirable situation.

These definitions also require that $R_1, \ldots, R_k$ be all rows or all columns. In these two pure cases, $EA(R_1), \ldots, EA(R_k)$ are pairwise disjoint. Therefore, a linear combination of $\overline{R}_1, \ldots, \overline{R}_k$ has a very simple structure and encodes essentially the same information as $\overline{R}_1, \ldots, \overline{R}_k$. In contrast, if at least one R_i is a row and at least one R_j is a column, then a linear combination of $\overline{R}_1, \ldots, \overline{R}_k$ may have a very complex structure and may encode very different information from that contained in $\overline{R}_1, \ldots, \overline{R}_k$. Furthermore, unlike in the two pure cases, these definitions do not seem to have useful characterizations in the mixed case.

The importance of the first four types of invariants considered in the definitions are evident. The fifth type, a rectangular sum invariant, is motivated by a popular technique for protecting information in a table. Let e be an invariant cell at row i and column j. To protect e, row i can be split into several rows and column j into several columns. Correspondingly, e is split into four or more cells. Then enough of these refined cells can be suppressed to ensure that each suppressed refined cell is protected. However, the sum of the suppressed refined cells of e is a rectangular sum invariant. This property can be used to uniquely determine the value of e. Thus the consideration of rectangular sum invariants renders this refinement approach useless at the third level of data security.

4.1. Equivalence of k row-column protection. This section shows that the five definitions of k row-column protection are all equivalent.

LEMMA 4.1. *Every sum minimal invariant is rectangular.*

Proof. Let F be a sum minimal invariant of $\mathcal{T}$. If $EA(F)$ consists of an edge not in any strong component of $\mathcal{H}$, then F is trivially rectangular. Otherwise, by Theorem 2.1, $EA(F)$ is a bipartite minimal cut set of a strong component $\mathcal{H}'$ of $\mathcal{H}$. By Fact 1, $\mathcal{H}' - EA(F)$ has two connected components $\mathcal{H}'_1$ and $\mathcal{H}'_2$. Let U_1 and U_2 be the sets of endpoints of $EA(F)$ in $\mathcal{H}'_1$ and $\mathcal{H}'_2$, respectively. By the bipartiteness of $EA(F)$, without loss of generality the vertices in U_1 are rows in $\mathcal{T}$ and those in U_2 are columns. Then F is rectangular because $EA(F)$ consists of the edges between U_1 and U_2 in $\mathcal{H}$. □

LEMMA 4.2. *If no $EA(R_i)$ is empty, the statements below are equivalent:*

1. *Every positive invariant F with $EA(F) \subseteq \cup_{i=1}^{k} EA(R_i)$ is a linear combination of $\overline{R}_1, \ldots, \overline{R}_k$.*
2. *Every sum invariant F with $EA(F) \subseteq \cup_{i=1}^{k} EA(R_i)$ is a linear combination of $\overline{R}_1, \ldots, \overline{R}_k$.*
3. *Every rectangular sum invariant F with $EA(F) \subseteq \cup_{i=1}^{k} EA(R_i)$ is a linear combination of $\overline{R}_1, \ldots, \overline{R}_k$.*
4. *$\overline{R}_1, \ldots, \overline{R}_k$ are the only sum minimal invariants of $\mathcal{T}$ whose effective areas are subsets of $\cup_{i=1}^{k} EA(R_i)$.*

Proof. The directions $1 \Rightarrow 2 \Rightarrow 3$ are straightforward. The direction $4 \Rightarrow 1$ follows from the fact that by statement 4, $\overline{R}_1, \ldots, \overline{R}_k$ are the only factors in the decomposition in Theorem 2.1(3) for a positive invariant F with $EA(F) \subseteq \cup_{i=1}^{k} EA(R_i)$. To prove $3 \Rightarrow 4$, note that because $\overline{R}_j$ is a positive invariant for all R_j, by Theorem 2.1(3) there is a sum minimal invariant F with $EA(F) \subseteq EA(\overline{R}_j)$. Since F is also rectangular, by statement 3, $F = \sum_{i=1}^{k} c_i \cdot \overline{R}_i$ for some c_i. Because $\overline{R}_1, \ldots, \overline{R}_k$ share no variable, by the minimality of F and coefficient comparison $\overline{R}_j$ equals F and thus is a sum minimal invariant. To prove the desired uniqueness of $\overline{R}_1, \ldots, \overline{R}_k$, let F' be a sum minimal invariant with $EA(F') \subseteq \cup_{i=1}^{k} EA(R_i)$. By Lemma 4.1, F' is rectangular. By statement 3, $F' = \sum_{i=1}^{k} c'_i \cdot \overline{R}_i$ for some c'_i. Because F' is nonzero, some $c'_h \neq 0$. Because $\overline{R}_1, \ldots, \overline{R}_k$ do not share variables, $EA(\overline{R}_h) \subseteq EA(F')$. Then, $F' = \overline{R}_h$ by

coefficient comparison and the minimality of F'. □

LEMMA 4.3. *If no $EA(R_i)$ is empty, the statements below are equivalent:*

1. *Every linear invariant of $\mathcal{T}$ whose effective area is a subset of $\cup_{i=1}^{k} EA(R_i)$ is a linear combination of $\overline{R}_1, \ldots, \overline{R}_k$.*
2. *Every unitary invariant whose effective area is a subset of $\cup_{i=1}^{k} EA(R_i)$ is a linear combination of $\overline{R}_1, \ldots, \overline{R}_k$.*
3. *$\overline{R}_1, \ldots, \overline{R}_k$ and their nonzero multiples are the only minimal linear invariants of $\mathcal{T}$ whose effective areas are subsets of $\cup_{i=1}^{k} EA(R_i)$.*

Proof. The proof is similar to that of Lemma 4.2. □

LEMMA 4.4. *$\{R_1, \ldots, R_k\}$ is protected with respect to the positive invariants (respectively, the linear invariants) if and only if the following statements hold:*

1. *For each strong component D of $\mathcal{H}$ and each vertex R_i contained in D, the component D contains all edges incident to R_i in $\mathcal{H}$.*
2. *The nonempty sets among $EA(R_1), \ldots, EA(R_k)$ are the only bipartite minimal edge cuts (respectively, the only minimal edge cuts) of the strong components of $\mathcal{H}$ among the subsets of $\cup_{i=1}^{k} EA(R_i)$.*
3. *Each vertex R_i is either isolated or incident to two or more edges in $\mathcal{H}$.*

Proof. The proofs of the lemma for the positive invariants and of that for the general invariants are similar; only the former is detailed here. For the direction $\Rightarrow$, statement 3 follows from the second condition of the definition of $\{R_1, \ldots, R_k\}$ being protected. Then statements 1 and 2 follow from Lemma 4.2(1), 4.2(4) and Theorem 2.1(1), 2.1(2). For the direction $\Leftarrow$, by statements 1 and 2, Theorem 2.1, and Lemma 4.2, the first condition of $\{R_1, \ldots, R_k\}$ being protected is satisfied. The second condition then follows from statement 3. □

A set of vertices in a connected graph is a *vertex cut* if its removal disconnects the graph.

FACT 2. *If each $EA(R_i)$ is included in the strong component of $\mathcal{H}$ that contains R_i, then the following statements are equivalent:*

1. *Among the subsets of $\cup_{i=1}^{k} EA(R_i)$, the nonempty sets $EA(R_i)$ are the only minimal edge cuts of the strong components of $\mathcal{H}$.*
2. *Among the subsets of $\cup_{i=1}^{k} EA(R_i)$, the nonempty sets $EA(R_i)$ are the only bipartite minimal edge cuts of the strong components of $\mathcal{H}$.*
3. *$\{R_1, \ldots, R_k\}$ includes no vertex cut of any strong component of $\mathcal{H}$.*

THEOREM 4.5. *The five definitions of a set of k rows or k columns being protected are all equivalent.*

Proof. If some $EA(R_i) = \emptyset$, then $\{R_1, \ldots, R_k\}$ is protected if and only if $\{R_1, \ldots, R_k\} - \{R_i\}$ is protected. Thus, without loss of generality, assume that no $EA(R_i)$ is empty. Then, by Lemma 4.2, the protection definitions with respect to the positive, sum, and rectangular invariants are all equivalent. Similarly, by Lemma 4.3, those with respect to the general and unitary invariants are also equivalent. This theorem then follows directly from Lemma 4.4 and Fact 2. □

4.2. k row-column protection and bipartite-$(k+1)$-connectivity. A connected bipartite graph $\mathcal{G} = (X, Y, I)$ is *bipartite-$(k+1)$-connected* if $|X| \geq k+1$, $|Y| \geq k+1$, and neither X nor Y includes a vertex cut of at most k vertices. $\mathcal{G}$ is *$(k+1)$-connected* if $|X \cup Y| \geq k+1$ and there is no vertex cut of at most k vertices.

LEMMA 4.6. *$\{R_1, \ldots, R_k\}$ is protected if and only if the statements below hold:*

1. *For each strong component D of $\mathcal{H}$ and each vertex $R_i \in D$, D contains all the edges incident to R_i in $\mathcal{H}$.*
2. *$\{R_1, \ldots, R_k\}$ includes no vertex cut of any strong component of $\mathcal{H}$.*

3. *Each R_i is either isolated or incident to two or more edges in $\mathcal{H}$.*

Proof. This lemma follows from Theorem 4.5, Lemma 4.4, and Fact 2. □

THEOREM 4.7. *Every set of at most k rows or k columns of $\mathcal{T}$ is protected if and only if every nonsingleton connected component of $\mathcal{H}$ is strongly connected and bipartite-$(k+1)$-connected.*

Proof. This theorem follows directly from Lemma 4.6. □

COROLLARY 4.8.

1. *Given $\mathcal{H}$ and $\{R_1, \ldots, R_k\}$, whether $\{R_1, \ldots, R_k\}$ is protected can be determined in $O(|\mathcal{H}|)$ time.*
2. *Given $\mathcal{H}$ and k, whether $\mathcal{T}$ has any unprotected set of at most k rows or k columns can be answered in $O(k^4n^2)$ time, where n is the number of vertices in $\mathcal{H}$.*

Proof. Statement 1 follows from Lemma 4.6 in a straightforward manner using linear-time algorithms for connectivity and strong connectivity [3]. Statement 2 follows from Theorem 4.7. The key step is to test the bipartite-$(k+1)$-connectivity of $\mathcal{H}$ within the stated time bound. We first construct two auxiliary graphs $\mathcal{H}_A$ and $\mathcal{H}_B$. For each vertex $u \in A$, replace u with $k+1$ copies in $\mathcal{H}_A$. For each $u \in A$ and each edge e in $\mathcal{H}$ between u and a vertex $v \in B$, replace e with $k+1$ copies between v and the $k+1$ copies of u in $\mathcal{H}_A$. $\mathcal{H}_B$ is obtained by exchanging A and B in the construction. Because $\mathcal{H}$ is connected and each vertex in A is duplicated $k+1$ times, $\mathcal{H}_A$ has a vertex cut U of at most k vertices if and only if U is a subset of B and is a vertex cut of $\mathcal{H}$. A symmetrical statement for B also holds. Thus $\mathcal{H}$ is bipartite-$(k+1)$-connected if and only if both $\mathcal{H}_A$ and $\mathcal{H}_B$ are $(k+1)$-connected. This corollary then follows from the fact [2], [17] that the $(k+1)$-connectivity of an m-vertex graph can tested in $O(k^2m^2)$ time if $k \leq \sqrt{m}$. □

COROLLARY 4.9. *Given $\mathcal{H}$, it takes $O(|\mathcal{H}|)$ time to find the unprotected rows and columns of $\mathcal{T}$ and decide whether all individual rows and columns of $\mathcal{T}$ are protected.*

Proof. This corollary follows from Lemma 4.6 in a straightforward manner using linear-time algorithms for strong connectivity and 2-connectivity [3]. □

4.3. Optimal suppression problems for k row-column protection.

PROBLEM 6 (protection of all sets).

- Input: $\mathcal{T}$ and two integers $k > 0$ and $p \geq 0$.
- Output: Is there a set P consisting of at most p published cells of $\mathcal{T}$ such that every set of at most k rows or k columns is protected in the table formed by $\mathcal{T}$ with the cells in P also suppressed?

Problem 6 can be reformulated as the following graph augmentation problem.

PROBLEM 7.

- Input: A complete bipartite graph $\mathcal{H}'$, a subgraph $\mathcal{H}$, and integers $k > 0$ and $p \geq 0$.
- Output: Is there a set P of at most p edges in $\mathcal{H}' - \mathcal{H}$ such that each nonsingleton connected component of $\mathcal{H} \cup P$ is strongly connected and bipartite-$(k+1)$-connected?

LEMMA 4.10. *Problems 6 and 7 can be reduced to each other in linear time.*

Proof. The proof follows from Theorem 4.7. □

THEOREM 4.11. *For $k = 1$, Problems 6 and 7 are NP-complete. Thus, both problems are NP-complete for general k.*

Proof. Problems 6 and 7 are both in NP. To prove their completeness for $k = 1$, by Lemma 4.10, it suffices to reduce Problem 3 to Problem 7 with $k = 1$. Given an instance $S = \{s_1, \ldots, s_\alpha\}$, $W = \{S_1, \ldots, S_\beta\}$, h of Problem 3, let $\mathcal{H}' = (A, B, E'), \mathcal{H} =$

$(A, B, E), p$ be the instance constructed for Theorem 3.4. The next two claims show that this transformation is indeed a desired reduction.

CLAIM 3. *If some $S' \subseteq S$ with $|S'| \leq h$ has at least one element in each S_j, then some $P \subseteq E' - E$ consists of at most p edges such that every nonsingleton connected component of $\mathcal{H} \cup P$ is strongly connected and bipartite-2-connected.*

To prove this claim, observe that for each S_j, some $s_{i_j} \in S' \cap S_j$ exists. Let $P_1 = \{b_1 \to a_{i_1}, \ldots, b_\beta \to a_{i_\beta}\}$, which exists by Rule 3(4) of the construction of $\mathcal{H}'$, $\mathcal{H}$, and p. By Rule 3(3), $P_2 = \{a_{i_1} \to b_0, \ldots, a_{i_\beta} \to b_0\}$ exists. Let $P = P_1 \cup P_2$. Note that P_1 consists of β edges. P_2 consists of at most $|S'|$ edges. Thus P has at most $p = \beta + h$ edges. For each j with $1 \leq j \leq \beta$, the edges $b_0 \to a_0, a_0 \to b_j, b_j \to a_{i_j}$, and $a_{i_j} \to b_0$ form a vertex-simple traversable cycle. These cycles all go through $b_0 \to a_0$ and form the only nonsingleton connected component of $\mathcal{H} \cup P$. This component is clearly strongly connected and bipartite-2-connected. This finishes the proof of Claim 3.

CLAIM 4. *If some $P \subseteq E' - E$ consists of at most p edges such that every nonsingleton connected component of $\mathcal{H} \cup P$ is strongly connected and bipartite-2-connected, then some $S' \subseteq S$ with $|S'| \leq h$ has at least one element in each S_j.*

The proof of this claim is the same as that of Claim 2 and uses only the componentwise strong connectivity of $\mathcal{H} \cup P$. This, then, finishes the proof of Theorem 4.11. □

The next two problems are variants of Problems 6 and 7.

PROBLEM 8 (protection of all sets).

- Input: The suppressed graph of a table $\mathcal{T}$ whose total graph is undirected and a positive integer k.
- Output: A set P consisting of the smallest number of published cells of $\mathcal{T}$ such that every set of at most k rows or k columns is protected in the table formed by $\mathcal{T}$ with the cells in P also suppressed.

PROBLEM 9.

- Input: A bipartite undirected graph $\mathcal{H} = (A, B, E)$ and a positive integer k.
- Output: A set P consisting of the smallest number of undirected edges between A and B but not in E such that every nonsingleton connected component of $(A, B, E \cup P)$ is bipartite-$(k+1)$-connected.

LEMMA 4.12. *Problems 8 and 9 can be reduced to each other in linear time.*

Proof. The proof is similar to that of Lemma 4.10. □

THEOREM 4.13 (see [13]). *For $k = 1$, Problem 9 can be solved in linear time.*

THEOREM 4.14. *For $k = 1$, Problem 8 can be solved in linear time.*

Proof. The proof follows from Lemma 4.12 and Theorem 4.13. □

5. Protection of a table. Let $R_1, \ldots, R_n$ be the rows and columns of $\mathcal{T}$. $\mathcal{T}$ is *protected* with respect to the positive invariants (respectively, the sum invariants or the rectangular sum invariants) if it holds the conditions below:

1. Every positive invariant (respectively, nonzero sum invariant or nonzero rectangular sum invariant) of $\mathcal{T}$ is a positive linear combination of $\overline{R}_1, \ldots, \overline{R}_n$, where a *positive* linear combination is one that has no negative coefficients and at least one positive coefficient.
2. $\mathcal{T}$ has no invariant cell.

These definitions allow only positive linear combinations because general linear combinations of $\overline{R}_1, \ldots, \overline{R}_n$ generate all linear invariants and leave nothing for protection. This restriction excludes the protection with respect to the general linear invariants.

As a result, the protection with respect to the unitary invariants is also not considered, because by Theorem 2.1, these invariants have the same structures as the general linear invariants do.

THEOREM 5.1. *The three definitions of a table being protected are all equivalent.*

Proof. Because a protected table has no invariant cells, each row or column has either no suppressed cell or at least two suppressed cells. It suffices to prove that if $\mathcal{T}$ holds this condition, then the following statements are equivalent:

1. Every positive invariant is a positive linear combination of $\overline{R}_1, \ldots, \overline{R}_n$.
2. Every nonzero sum invariant is a positive linear combination of $\overline{R}_1, \ldots, \overline{R}_n$.
3. Every nonzero rectangular sum invariant of $\mathcal{T}$ is a positive linear combination of $\overline{R}_1, \ldots, \overline{R}_n$.
4. The nonzero linear invariants among $\overline{R}_1, \ldots, \overline{R}_n$ are the only sum minimal invariants of $\mathcal{T}$.

The directions $1 \Rightarrow 2$ and $2 \Rightarrow 3$ are straightforward. The direction $4 \Rightarrow 1$ follows from Theorem 2.1(3). To prove the direction $3 \Rightarrow 4$, note that for each R_j with $EA(R_j) \neq \emptyset$, $\overline{R}_j$ is a nonzero sum invariant. By Theorem 2.1 there is a sum minimal invariant F with $EA(F) \subseteq EA(\overline{R}_j)$. F is also rectangular. By statement 3, $F = \sum_{i=1}^{k} c_i \cdot \overline{R}_i$, where $c_i \geq 0$. By coefficient comparison there is some $c_h > 0$ with $EA(R_h) \neq \emptyset$. Because $c_i \geq 0$, $\emptyset \neq EA(\overline{R}_h) \subseteq EA(F) \subseteq EA(\overline{R}_j)$. Then $R_h = R_j$ because two distinct R_i's cannot share more than one cell and each nonempty $EA(R_i)$ contains at least two cells. Thus $\overline{R}_j$ equals F and is a sum minimal invariant. To prove the desired uniqueness of $\overline{R}_1, \ldots, \overline{R}_n$, let F' be a sum minimal invariant with $EA(F') \subseteq \cup_{i=1}^{k} EA(R_i)$. By Lemma 4.1, F' is rectangular. By statement 3, $F' = \sum_{i=1}^{k} c'_i \cdot \overline{R}_i$, where $c'_i \geq 0$. By coefficient comparison there is some $c'_j > 0$ with $EA(\overline{R}_j) \neq \emptyset$. Because $c'_i \geq 0$, $EA(\overline{R}_j) \subseteq EA(F')$. Then $F' = \overline{R}_j$ by coefficient comparison and the minimality of F'. □

5.1. Table protection and bipartite completeness. A graph $\mathcal{G} = (X, Y, I)$ is *bipartite complete* if it is complete, $|X| \geq 2$, and $|Y| \geq 2$.

FACT 3. *Let $u_1, \ldots, u_g$ be the vertices in $\mathcal{G}$. Let $EA(u_i)$ be the set of edges incident to u_i. Then $\mathcal{G}$ is bipartite complete if and only if it is bridge free and has more than one vertex and the sets $EA(u_i)$ are its only bipartite minimal edge cuts.*

THEOREM 5.2. *$\mathcal{T}$ is protected if and only if each nonsingleton connected component of $\mathcal{H}$ is strongly connected and bipartite complete.*

Proof. By Fact 3, it suffices to prove that the following statments are equivalent:

1. $\mathcal{T}$ is protected.
2. The nonzero invariants among $\overline{R}_1, \ldots, \overline{R}_n$ are the only sum minimal invariants of $\mathcal{T}$. Also, each R_i contains either no suppressed cell or at least two suppressed cells.
3. Each connected component of $\mathcal{H}$ is strongly connected and bridge free. Also, the nonempty sets among $EA(R_1), \ldots, EA(R_n)$ are the only bipartite minimal edge cuts of the strong components of $\mathcal{H}$.

The equivalence $1 \Leftrightarrow 2$ follows from the proof of Theorem 5.1. The equivalence $2 \Leftrightarrow 3$ follows from Theorems 2.1 and 3.1. □

COROLLARY 5.3. *Given $\mathcal{H}$, it takes linear time in the size of $\mathcal{H}$ to determine whether $\mathcal{T}$ is protected.*

Proof. This is an immediate corollary of Theorem 5.2. □

5.2. Optimal suppression problems for table protection.

PROBLEM 10 (protection of a table).

- Input: $\mathcal{T}$ and a nonnegative integer p.
- Output: Is there a set P consisting of at most p published cells of $\mathcal{T}$ such that the table formed by $\mathcal{T}$ with the cells in P also suppressed is protected?

Problem 10 can be reformulated as the following graph augmentation problem.

PROBLEM 11.

- Input: A complete bipartite graph $\mathcal{H}'$, a subgraph $\mathcal{H}$, and an integer $p \geq 0$.
- Output: Is there a set P of at most p edges in $\mathcal{H}' - \mathcal{H}$ such that each nonsingleton connected component $\mathcal{H} \cup P$ is strongly connected and bipartite complete?

LEMMA 5.4. *Problems* 10 *and* 11 *can be reduced to each other in linear time.*

Proof. The proof follows from Theorem 5.2. □

THEOREM 5.5. *Problems* 10 *and* 11 *are NP-complete.*

Proof. Problems 10 and 11 are both in NP. To prove their completeness, by Lemma 5.4, it suffices to reduce Problem 3 to Problem 11. Given an instance $S = \{s_1, \ldots, s_\alpha\}$, $W = \{S_1, \ldots, S_\beta\}$, h of Problem 3, let $\mathcal{H}' = (A, B, E'), \mathcal{H} = (A, B, E), p$ be the instance constructed for Theorem 3.4 with the modification below:

- Rule 5′: Let $p = (\beta + 1) \cdot h$.

This construction can be computed in polynomial time. The next two claims show that it is a desired reduction from Problem 3 to Problem 11.

CLAIM 5. *If some $S' \subseteq S$ with $|S'| \leq h$ has at least one element in each S_j, then some $P \subseteq E' - E$ consists of at most p edges such that every nonsingleton connected component of $\mathcal{H} \cup P$ is strongly connected and bipartite complete.*

To prove this claim, observe that for each S_j, some $s_{i_j} \in S' \cap S_j$ exists. Let $A' = \{a_{i_1}, \ldots, a_{i_\beta}\}$. Let $B' = \{b_1, \ldots, b_\beta\}$. Let P_1 be the set of edges in E' from B' to A'. Let P_2 be the set of edges in E' from A' to b_0. Let $P = P_1 \cup P_2$. Note that P has at most $p = (\beta + 1) \cdot h$ edges because A' has at most $|S'| \leq h$ vertices. For each j with $1 \leq j \leq \beta$, the edge $b_j \to a_{i_j}$ is in P_1 by Rule 3(4) of the construction of $\mathcal{H}'$, $\mathcal{H}$, and p. Also, $b_0 \to a_0$, $a_0 \to b_j$, and $a_{i_j} \to b_0$ are in $\mathcal{H} \cup P$. These four edges form a vertex-simple traversable cycle. These cycles form the only nonsingleton connected component in $\mathcal{H} \cup P$. Because these cycles all go through a_0, this component is strongly connected. By the choice of P, this component is bipartite complete. This finishes the proof of Claim 5.

CLAIM 6. *If some $P \subseteq E' - E$ consists of at most p edges such that every nonsingleton connected component of $\mathcal{H} \cup P$ is strongly connected and bipartite complete, then some $S' \subseteq S$ with $|S'| \leq h$ has at least one element in each S_j.*

To prove this claim, observe that because every connected component of $\mathcal{H} \cup P$ is strongly connected, for each j with $1 \leq j \leq \beta$, the set P contains some edges $b_j \to a_{i_j}$ and $a_{i_j} \to b_{j'}$. Then $i_j \neq 0$ and s_{i_j} exists in S_j by Rule 3 of the construction of $\mathcal{H}'$, $\mathcal{H}$, and p. Let $S' = \{s_{i_1}, \ldots, s_{i_\beta}\}$. Let D be the connected component of $\mathcal{H} \cup P$ that contains a_0. Then D also contains $a_{i_1}, \ldots, a_{i_\beta}$ and $b_0, \ldots, b_\beta$. By the completeness of D, the set P has at least $(\beta+1) \cdot |S'|$ edges. Thus $|S'| \leq h$ because $|P| \leq p = (\beta+1) \cdot h$. This finishes the proof of Claim 6 and thus that of Theorem 5.5. □

The next two problems are variants of Problems 10 and 11.

PROBLEM 12 (protection of a table).

- Input: The suppressed graph $\mathcal{H}$ of a table $\mathcal{T}$ whose total graph is undirected.
- Output: A set P consisting of the smallest number of published cells of $\mathcal{T}$ such that the table formed by $\mathcal{T}$ with the cells in P also suppressed is protected.

PROBLEM 13.

- Input: A bipartite undirected graph $\mathcal{H} = (A, B, E)$.

- Output: A set P consisting of the smallest number of undirected edges between A and B but not in E such that every nonsingleton connected component of $(A, B, E \cup P)$ is bipartite complete.

LEMMA 5.6. *Problems* 12 *and* 13 *can be reduced to each other in linear time.*

Proof. The proof is similar to that of Lemma 5.4. □

THEOREM 5.7 (see [15]). *Problem* 13 *can be solved in optimal* $O(|\mathcal{H}| + p)$ *time, where* p *is the output size.*

THEOREM 5.8. *Problem* 12 *can be solved in optimal* $O(|\mathcal{H}| + p)$ *time, where* p *is the output size.*

Proof. This theorem follows from Lemma 5.6 and Theorem 5.7. □

6. Discussions. The relationship between the data security of $\mathcal{T}$ and the connectivity of $\mathcal{H}$ is summarized and compared in Table 1.

TABLE 1

Levels of data security	Degrees of graph connectivity
all cells	strongly connected, bridge free
all rows and columns	strongly connected, bipartite-2-connected
all sets of k rows or k columns	strongly connected, bipartite-$(k+1)$-connected
the whole table	strongly connected, bipartite-complete

LEMMA 6.1. *Let* R *be a row or column of* $\mathcal{T}$. *Let* k *be the smallest number of row vertices or column vertices in any nonsingleton connected component of* $\mathcal{H}$.

1. *If* R *is protected, then every suppressed cell in* R *is also protected.*
2. *If a set of* k *rows or* k *columns of* $\mathcal{T}$ *is protected, then every subset of that set is also protected.*
3. *If* $\mathcal{T}$ *is protected, then every set of* $k-1$ *rows or* $k-1$ *columns is also protected.*

Note that the converses of the above statements are all false.

Proof. Statements 1 and 2 are straightforward. Statement 3 follows from Theorems 4.7 and 5.2. □

Acknowledgments. The author is deeply grateful to Dan Gusfield for his help. The author wishes to thank the anonymous referees for very helpful and thorough comments.

REFERENCES

[1] G. J. BRACKSTONE, L. CHAPMAN, AND G. SANDE, *Protecting the confidentiality of individual statistical records in Canada*, in Proc. of the Conference of the European Statisticians 31st Plenary Session, Geneva, 1983.

[2] J. CHERIYAN, M. Y. KAO, AND R. THURIMELLA, *Scan-first search and sparse certificates: An improved parallel algorithm for k-vertex connectivity*, SIAM J. Comput., 22 (1993), pp. 157–174.

[3] T. H. CORMEN, C. L. LEISERSON, AND R. L. RIVEST, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1991.

[4] L. H. COX, *Disclosure analysis and cell suppression*, in Proc. of the American Statistical Association, Social Statistics Section, 1975, pp. 380–382.

[5] ———, *Suppression methodology in statistics disclosure*, in Proc. of the American Statistical Association, Social Statistics Section, 1977, pp. 750–755.

[6] ———, *Automated statistical disclosure control*, in Proc. of the American Statistical Association, Survey Research Method Section, 1978, pp. 177–182.

[7] ———, *Suppression methodology and statistical disclosure control*, J. Amer. Statist. Assoc., Theory and Method Section, 75 (1980), pp. 377–385.

[8] L. H. Cox and G. Sande, *Techniques for preserving statistical confidentiality*, in Proc. of the 42nd Session of the International Statistical Institute, the International Association of Survey Statisticians, 1979.
[9] D. Denning, *Cryptography and Data Security*, Addison–Wesley, Reading, MA, 1982.
[10] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, NY, 1979.
[11] D. Gusfield, *Optimal mixed graph augmentation*, SIAM J. Comput., 16 (1987), pp. 599–612.
[12] ———, *A graph theoretic approach to statistical data security*, SIAM J. Comput., 17 (1988), pp. 552–571.
[13] T. S. Hsu and M. Y. Kao, *Optimal augmentation for componentwise bipartite biconnectivity in linear time*, 1994, manuscript.
[14] M. Y. Kao, *Efficient detection and protection of information in cross tabulated tables* II: *Minimal linear invariants*, in Proc. of the 1995 Asian Computing Science Conference, Pathumthani, Thailand, December 11–13, 1995.
[15] ———, *Linear-time optimal augmentation for componentwise bipartite-completeness of graphs*, Inform. Process. Lett., (1995), pp. 59–63.
[16] M. Y. Kao and D. Gusfield, *Efficient detection and protection of information in cross tabulated tables* I: *Linear invariant test*, SIAM J. Discrete Math., 6 (1993), pp. 460–476.
[17] H. Nagamochi and T. Ibaraki, *A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph*, Algorithmica, (1992), pp. 583–596.
[18] R. T. Rockafellar, *The elementary vectors of R^n*, in Combinatorial Mathematics and its Applications, R. C. Bose and T. A. Dowling, eds., University of North Carolina Press, Chapel Hill, NC, 1969.
[19] ———, *Network Flows and Monotropic Optimization*, Wiley, New York, NY, 1984.
[20] G. Sande, *Towards automated disclosure analysis for establishment based statistics*, Tech. report, Statistics Canada, 1977.
[21] ———, *A theorem concerning elementary aggregations in simple tables*, Tech. report, Statistics Canada, 1978.
[22] ———, *Automated cell suppression to preserve confidentiality of business statistics*, Statist. J. United Nations, 2 (1984), pp. 33–41.
[23] ———, *Confidentiality and polyhedra, an analysis of suppressed entries on cross tabulations*, Tech. report, Statistics Canada, unknown date.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 101–117, February 1995

010

ON LINEAR RECOGNITION OF TREE-WIDTH AT MOST FOUR*

DANIEL P. SANDERS†

Abstract. A graph G has tree-width at most k if the vertices of G can be decomposed into a tree-like structure of sets of vertices, each set having cardinality at most $k+1$. An alternate definition of tree-width is stated in terms of a k-elimination sequence, which is an order to eliminate the vertices of the graph such that each vertex, at the time it is eliminated from the graph, has degree at most k. Arnborg and Proskurowski showed that if a graph has tree-width at most a fixed k, then many NP-hard problems can be solved in linear time, provided this k-elimination sequence is part of the input. These algorithms are very efficient for small k, such as 2, 3, or 4, but may be impractical for large k as they depend exponentially on k. A reduction process is developed, and reductions are shown that can be applied to a graph of tree-width at most four without increasing its tree-width. Further, each graph of tree-width at most four contains one of these reductions. The reductions are then used in a linear-time algorithm that generates a 4-elimination sequence, if one exists.

Key words. tree-width, partial k-tree, graph algorithms

AMS subject classifications. 05C85, 05C75, 68Q25, 68R10

1. Introduction.

Many NP-hard problems can be solved in polynomial time on restricted classes of graphs. An important class of graphs for which this is the case is the class of series-parallel graphs. A graph is *series-parallel* if it can be reduced to the null graph by a finite sequence of deleting loops, deleting vertices of degree at most one, series-reductions, and parallel-reductions. (*Remark.* Series-parallel was originally defined in terms of networks, or 2-connected graphs. A 2-connected graph is *series-parallel* if it can be reduced to the graph consisting of one vertex and one loop by a finite sequence of series-reductions and parallel-reductions.) This same class of graphs can be decribed as all graphs with no minor isomorphic to K_4. A third characterization is important here: a graph is series-parallel if and only if it has tree-width at most two [21] (equivalently, is a partial 2-tree). This is where the generalizations arise. Now tree-width at most k, for any fixed k, will be defined.

A *tree-decomposition* of a graph G is a pair (T, B), where T is a tree and B is a collection of *bags* satisfying $B := \{B_t \subset V(G) : t \in V(T)\}$, with the following properties.

1. $\bigcup_{t \in V(T)} B_t = V(G)$.
2. If $vw \in E(G)$, then for some $t \in V(T)$, $\{v, w\} \subset B_t$.
3. For every $x \in V(G)$, the induced subgraph of $\{t \in V(T) : x \in B_t\}$ with respect to T is a tree.

The *width* of a tree-decomposition is $\max\{|B_t| - 1 : t \in V(T)\}$. The *tree-width* of a graph G is the minimum width of a tree-decomposition of G.

The flexibility of tree-decompositions is important for proofs, as will be shown later. But an alternate definition of tree-width is more useful for algorithms. For a graph G, a *clique* of G is a complete subgraph of G. For $x \in V(G)$, the graph of G with x *eliminated*, in symbols $G * x$, is defined to be the graph obtained from G by

*Received by the editors January 19, 1993; accepted for publication (in revised form) February 22, 1995.

†Department of Mathematics, The Ohio State University, Columbus, OH 43210–1174 (dsanders@math.ohio-state.edu).

first adding a clique on the neighborhood of x and then deleting x (see [19]). A *k-elimination sequence* for a graph G is a labeling of its vertices, say, $V(G) = \{v_1, \ldots, v_n\}$ such that for each i satisfying $0 \leq i < n$, in the graph $G_i := G * v_1 * \cdots * v_i$, the degree of v_{i+1} is at most k. It is easy to show [23] that a graph has tree-width at most k if and only if it has a k-elimination sequence.

Extensive research has been performed in the area of fast algorithms for graphs of bounded tree-width. Over 200 papers in this area are listed in Hedetniemi's bibliography [13]. A special issue of *Discrete Applied Mathematics* (volume 54, issues 2–3, pages 97–291) was dedicated to the subject. Arnborg and Proskurowski [7] first showed that several NP-hard problems such as maximum independent set, minimum dominating set, chromatic number, Hamilton circuit, and network reliability can be solved in linear time for graphs of bounded tree-width. This paper sparked a rash of papers by these and others, showing linearity for more NP-hard problems on graphs of bounded tree-width [2], [5], [16], [22], eventually leading to a logical consideration of exactly what kind of problems can be solved on graphs in linear time [11]. Other problems, such as isomorphism [8], [18], have been shown to be polynomial for graphs of bounded tree-width. All of these algorithms, however, require a tree-decomposition or a k-elimination sequence to be part of the input.

Thus research has focused on finding a practical algorithm to produce the k-elimination sequence. A probabilistic algorithm was found by Matoušek and Thomas [17]. Deterministic algorithms of successively better complexity were found by Arnborg, Corneil, and Proskurowski [3] ($O(n^{k+2})$); Bodlaender and Kloks [10] ($O(n \log^2 n)$); Reed [20] ($O(n \log n)$); and Bodlaender [9] ($O(n)$). Although Bodlaender's linear-time algorithm is theoretically the best possible, the constants of his algorithm and the others mentioned are too large to be used in practice.

For small k, a k-elimination sequence can be found efficiently. This was first shown for series-parallel graphs ($k = 2$). It is not hard to construct a 2-elimination sequence in linear time. For $k = 3$, Arnborg and Proskurowski [6] (independently [14]) found a set of reductions—the series-parallel reductions and three others—to characterize graphs of tree-width at most three. Matoušek and Thomas [17] (see also [14]) were able to use a modification of these reductions to create a practical quadratic algorithm to produce 3-elimination sequences.

This paper extends and improves this work to the case $k = 4$. A set of reductions is first presented such that a graph has tree-width at most four if and only if it can be reduced to the null graph by a finite sequence of these reductions. Then these reductions are used to produce a simple, practical linear algorithm that finds a 4-elimination sequence (or a 3-elimination sequence). This case is of particular interest for a number of reasons. First, graphs of tree-width at most four arise naturally in several applications where a fast algorithm is required. The reliability graph of a battleship, constructed for weapons-effect simulations, has several thousand vertices but tree-width at most four [1]. At Bellcore, determining whether a SONET ring is present in certain communication networks is solvable in linear time, as the networks have small tree-width [12]. Also, in [7] it is admitted that the linear-time optimization algorithms mentioned previously contain constants that grow exponentially or super-exponentially in k. Thus algorithms of this type can be expected to be impractical for large k.

2. Reductions. Sections 2–7 of this paper consider only simple graphs. Thus any multiple edges created by performing the union of two graphs are assumed to be automatically deleted. The *addition* of an edge xy to a graph G, in symbols $G + xy$,

is the graph on the same vertices as G with the edge xy added to $E(G)$ if not present already. The *contraction* of an edge xy in a graph G, in symbols $G \cdot xy$, is the graph formed by adding a new vertex z not in $V(G)$, adding edges from z to each vertex that is a neighbor of either x or y, and then deleting the vertices x and y.

A *separation* of a graph G is a triple $(A, B, (v_1, \ldots, v_k))$, where A, B are subgraphs of G with $A \cup B = G$, $E(A) \cap E(B) = \emptyset$, and $V(A) \cap V(B) = \{v_1, \ldots, v_k\}$. Let a *structure* S be a pair $(G(S), (u_1, \ldots, u_j))$, where $G(S)$ is a graph and $u_1, \ldots, u_j$ are distinct vertices of $G(S)$, called the *vertices of attachment* of S. For a structure $S := (G(S), (u_1, \ldots, u_j))$, define $V(S) := V(G(S))$. Two structures $S := (G, (u_1, \ldots, u_j))$ and $T := (H, (v_1, \ldots, v_k))$ are *isomorphic* if $j = k$, and there is an isomorphism from G to H taking u_i to v_i for all $i \leq j$. A graph G *has* a structure S if G has a separation $(A, B, (v_1, \ldots, v_k))$ where $(B, (v_1, \ldots, v_k))$ is isomorphic to S. For graphs G and H and structures S and T, the graph H is obtained from G by *replacing* S by T if G has a separation $(A, B, (v_1, \ldots, v_k))$ and H has a separation $(A, C, (v_1, \ldots, v_k))$ such that $(B, (v_1, \ldots, v_k))$ is isomorphic to S and $(C, (v_1, \ldots, v_k))$ is isomorphic to T. Thus, loosely speaking, a copy of S in G is replaced by a copy of T.

A *reduction* R is a pair of structures, S_R and T_R, with the same sequence of vertices of attachment and $|V(S_R)| > |V(T_R)|$. For graphs G, H and reduction R, the graph H is obtained from G by *performing* R if H is obtained from G by replacing S_R by T_R. For each reduction R, define the following partial order: $H \leq_R G$ if there is a sequence of graphs $H = G_1, G_2, \ldots, G_k = G$ such that for every $i < k$, G_i is obtained from G_{i+1} by performing R. If A is a set of graphs, a reduction R is *A-safe* if for all graphs G, H satisfying $H \leq_R G$, $G \in A$ if and only if $H \in A$. By saying a graph *has* or *contains* a reduction R, it is meant that the graph has an S_R such that every edge of S_R between two of its vertices of attachment is incident to at least one vertex whose image in G has degree at most six in G (this restriction is essential for the linear algorithm). A set Q of reductions is *A-complete* if every nonnull graph G in A has some R in Q.

As an example, consider $TW2$, the set of all simple series-parallel graphs or simple graphs of tree-width at most two. In the following figures, some reductions are represented. For each reduction represented, the vertices of attachment are precisely the solid vertices, ordered in a consistent geometric fashion. Define the reductions Zero, One, Series by Fig. 1. Note T_{Zero} is meant to be the null graph. A well-known result is that {Zero, One, Series} is a $TW2$-complete set of $TW2$-safe reductions.

The next example is for $TW3$, the set of all simple graphs of tree-width at most three. Define the reductions Triangle, Buddy, Cube by Fig. 2. Arnborg and Proskurowski [6] showed that {Zero, One, Series, Triangle, Buddy, Cube} is a $TW3$-complete set of $TW3$-safe reductions.

Let $TW4$ be the set of all simple graphs with tree-width at most four. This paper will determine a $TW4$-complete set of $TW4$-safe reductions. A graph G is a *minor* of H if G can be obtained from a subgraph of H by a finite sequence of edge contractions. Two partial orders are helpful here. Let $G \leq_m H$ if G is isomorphic to a minor of H. Let $G \leq_* H$ if G is isomorphic to a graph that can be obtained from H by a finite sequence of eliminations of vertices of degree at most four. The following lemmas are easy to prove.

LEMMA 1. *If $G \in TW4$ and $H \leq_m G$, then $H \in TW4$.*

LEMMA 2. *If $G \in TW4$ and $G \leq_* H$, then $H \in TW4$.*

LEMMA 3. *If $G \leq_R H$ implies $G \leq_m H$ and $G \leq_* H$, then R is $TW4$-safe.*

It follows as a corollary that Zero, One, Series, Triangle, and Buddy are $TW4$-safe. To proceed further, reductions must be found that are not so straightforwardly

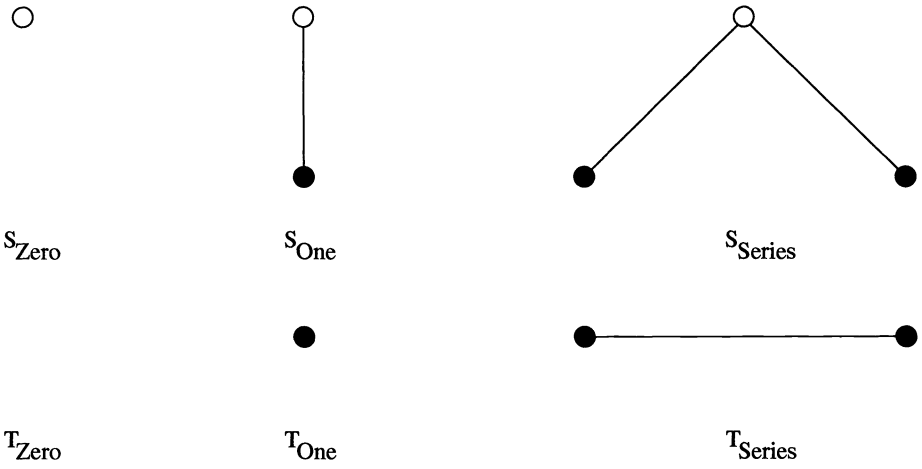


FIG. 1. *Reductions for TW2.*

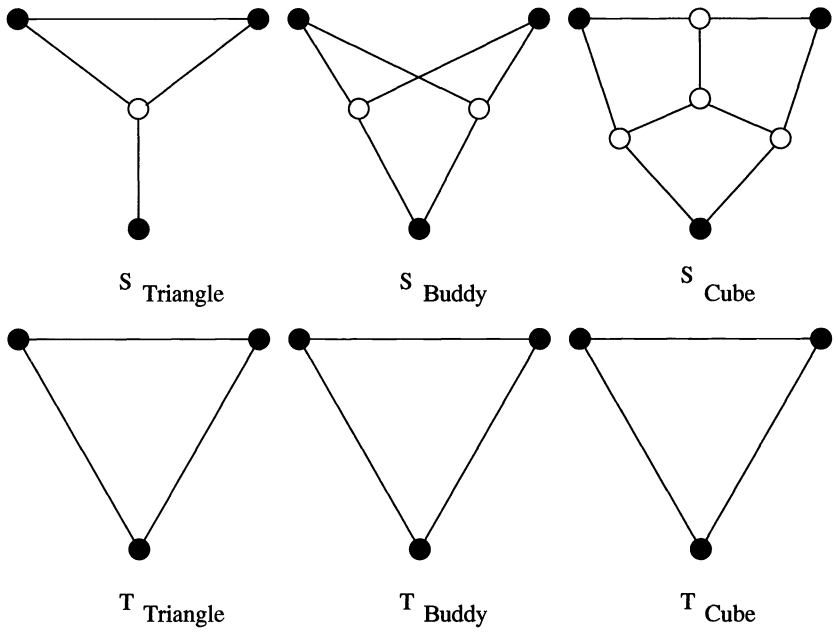


FIG. 2. *Reductions for TW3.*

$TW4$-safe. This can be done by the use of the following lemma, which is proven in various places.

LEMMA 4. *Every graph in $TW4$ is a subgraph of a chordal graph in $TW4$.*

The short proof of Lemma 5 demonstrates how Lemma 4 is used to help show that certain reductions are $TW4$-safe. Let Star-O (SO) be the reduction defined as follows. Let $S_{SO} := (W_5, (a, b, c, d))$, where W_5 is the wheel on the five vertices x, a, b, c, d with

hub x and circuit $abcd$. Let $T_{SO} := (K_4, (a, b, c, d))$, where K_4 is the complete graph on the vertices a, b, c, d.

LEMMA 5. *Star-O (SO) is a TW4-safe reduction.*

Proof. Let G, H be given satisfying $G \leq_{SO} H$. Without loss of generality, assume G is obtained from H by performing SO. Let a', b', c', d', x' be the vertices of G corresponding to a, b, c, d, x.

Assume $H \in TW4$. Since H is isomorphic to $G * x'$, by Lemma 2, $G \in TW4$.

Assume $G \in TW4$. Since $a'b'c'd'$ is a circuit in G, by Lemma 4, either $G + a'c'$ or $G + b'd'$ is in $TW4$. If $G + a'c' \in TW4$, since $(G + a'c') \cdot b'x'$ is isomorphic to H, by Lemma 1, $H \in TW4$. If $G + b'd' \in TW4$, since $(G + b'd') \cdot a'x'$ is isomorphic to H, by Lemma 1, $H \in TW4$. □

3. Simple Y-Δ reductions are $TW4$-safe. Let a reduction R be a *4-star* reduction if $T_R \leq_* S_R$. This type of reduction is important algorithmically. If a sequence of 4-star reductions takes a graph G to the empty graph, a 4-elimination sequence for G is created, showing that $G \in TW4$. Every reduction used in the linear algorithm in this paper will be a 4-star reduction. The first two groups of reductions in this paper are performed by eliminating vertices of degree three and, thus, are called Y-Δ reductions. This section discusses simple Y-Δ reductions, while the following section discusses Ladder Y-Δ reductions. Let YO, $H7$, TO, YI be defined by Fig. 3. This section shows that the simple Y-Δ reductions—YO, $H7$, TO, YI—are $TW4$-safe by a proof similar to that of Lemma 5.

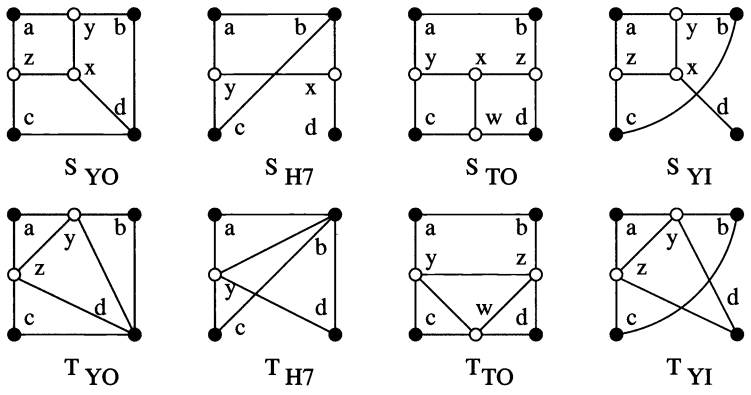


FIG. 3. *The simple Y-Δ reductions.*

For a graph G and a vertex x of G, G *has* a structure S *at* x if G has S such that the isomorphic copy of S in G has x as one of its vertices. For the ease of presentation of the following lemmas, assume that the structures for the reductions are equal to, instead of isomorphic to, one side of a separation of the indicated graphs.

LEMMA 6. *The reduction YO is a $TW4$-safe 4-star reduction.*

Proof. Clearly YO is a 4-star reduction. Let G, H be given satisfying $G \leq_{YO} H$. Without loss of generality, assume G is obtained from H by performing YO.

Assume $G \in TW4$. Since $G = H * x$, by Lemma 2, $H \in TW4$.

Assume $H \in TW4$. By Lemma 4, a chord of $ayxz$ may be added to H without increasing the tree-width. Assume first that $H + ax \in TW4$. Then since $J :=$

$(H + ax) \cdot by \cdot cz$ has an S_{SO} structure at x, $J * x \in TW4$ by Lemmas 1 and 5. Since $G * y * z = J * x$, by Lemma 2, $G \in TW4$.

Assume next that $H + yz \in TW4$. In this case, since $H + yz$ has an S_{Triangle} structure at x, $(H + yz) * x \in TW4$, by Lemma 3. But clearly, $(H + yz) * x = H * x = G$. □

LEMMA 7. *The reduction H7 is a TW4-safe 4-star reduction.*

Proof. Clearly $H7$ is a 4-star reduction. Let G, H be given satisfying $G \leq_{H7} H$. Without loss of generality, assume G is obtained from H by performing $H7$.

Assume $G \in TW4$. Since $G = H * x$, by Lemma 2, $H \in TW4$.

Assume $H \in TW4$. By Lemma 4, a chord of $abxy$ may be added to H without increasing the tree-width. Assume first that $H + ax \in TW4$. Then $J := (H + ax) \cdot cy \cdot dx \in TW4$ by Lemma 1. Since $G * y = J$, by Lemma 2, $G \in TW4$.

Assume next that $H + by \in TW4$. In this case, since $H + by$ has an S_{Triangle} structure at x, $(H + by) * x \in TW4$, by Lemma 3. But clearly, $(H + by) * x = H * x = G$. □

LEMMA 8. *The reduction TO is a TW4-safe 4-star reduction.*

Proof. Clearly TO is a 4-star reduction. Let G, H be given satisfying $G \leq_{TO} H$. Without loss of generality, assume G is obtained from H by performing TO.

Assume $G \in TW4$. Since $G = H * x$, by Lemma 2, $H \in TW4$.

Assume $H \in TW4$. By Lemma 4, a chord of $cyxw$ may be added to H without increasing the tree-width. Assume first that $H + cx \in TW4$. Then $J := (H + cx) \cdot ay \cdot bz \cdot dw \in TW4$, by Lemma 1. Since J has an S_{SO} structure at x, $J * x \in TW4$, by Lemma 5. But since $G * w * y * z = J * x$, by Lemma 2, $G \in TW4$.

Assume next that $H + yw \in TW4$. Here, since $H + yw$ has an S_{Triangle} structure at x, $(H + yw) * x \in TW4$, by Lemma 3. But clearly, $(H + yw) * x = H * x = G$. □

LEMMA 9. *The reduction YI is a TW4-safe 4-star reduction.*

Proof. Clearly YI is a 4-star reduction. Let G, H be given satisfying $G \leq_{YI} H$. Without loss of generality, assume G is obtained from H by performing YI.

Assume $G \in TW4$. Since $G = H * x$, by Lemma 2, $H \in TW4$.

Assume $H \in TW4$. By Lemma 4, a chord of $ayxz$ may be added to H without increasing the tree-width.

Assume first that $H + ax \in TW4$. Then $J := (H + ax) \cdot by \cdot cz \cdot dx \in TW4$, by Lemma 1. But since $G * y * z = J$, by Lemma 2, $G \in TW4$.

Assume next that $H + yz \in TW4$. In this case, since $H + yz$ has an S_{Triangle} structure at x, $(H + yz) * x \in TW4$, by Lemma 3. But clearly, $(H + yz) * x = H * x = G$. □

4. Ladder reductions are $TW4$-safe. This section shows some safe reductions of a special nature. A graph is taken to a smaller graph with a similar structure, although the reduction itself cannot be realized through minors or eliminations. These Triangle Ladder reductions are then used to show four other Y-Δ reductions, the Ladder reductions, are safe. Let the reductions DL, $XL1$, $XL2$, SL be defined by Fig. 4.

LEMMA 10. *Let G be a graph with tree-decomposition (T, B). If A is a set of vertices of a clique of G, then there is a $t \in V(T)$ with $A \subset B_t$.*

Lemma 10 is well known in the study of tree-width. This lemma will be used in the proofs of Lemmas 11, 13, and 15. Some definitions will also be useful. Given a tree T and distinct $r, s, t \in V(T)$, let the *centroid* of r, s, t be the unique vertex of T that is on the path from r to s, on the path from r to t, and on the path from s to

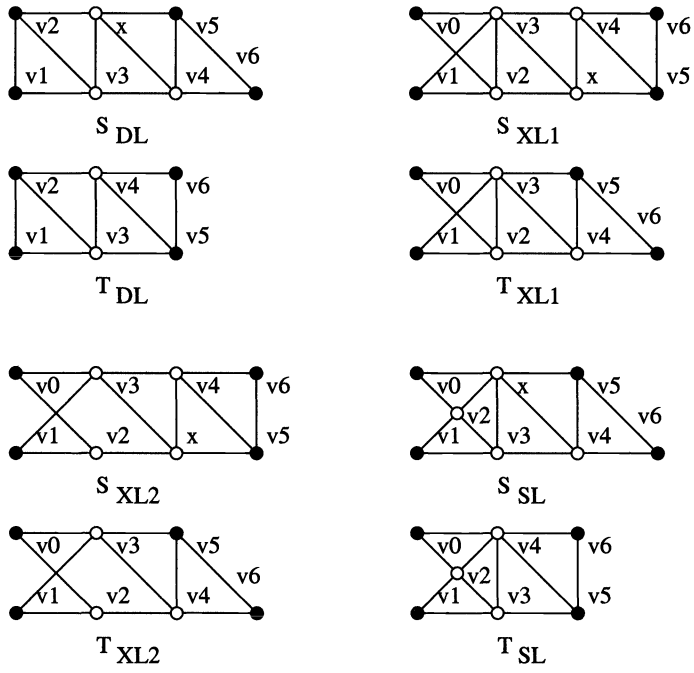


FIG. 4. *The Triangle Ladder reductions.*

t. Given a tree decomposition (T, B) of a graph G, let $G(T, B)$ be the graph with vertices $V(G)$ and edges $\{xy : \exists t \in V(T)$ such that $\{x, y\} \subset B_t\}$.

LEMMA 11. *The reduction Double Ladder (DL) is TW4-safe.*

Proof. Let G, H be given satisfying $G \leq_{DL} H$. Without loss of generality, assume G is obtained from H by performing DL.

Assume $G \in TW4$. Let a tree-decomposition (T, B) of G of width four be given.

If there is a bag B_t with $|B_t \cap \{v_1, \ldots, v_6\}| \geq 4$, then $J := G * v_3 * v_4 \leq_m G(T, B)$. This gives $J \in TW4$ by Lemma 1. Also, $H * v_3 * x * v_4 = J$, so $J \leq_* H$, and $H \in TW4$, by Lemma 2. Thus assume there is no such bag. By Lemma 10, for $i := 1, \ldots, 4$, there are vertices s_i of T such that $\{v_i, v_{i+1}, v_{i+2}\} \subset B_{s_i}$. Let $t_1 := s_1$, but for $i := 2, 3, 4$ let t_i be the closest vertex to t_1 with $\{v_i, v_{i+1}, v_{i+2}\} \subset B_{t_i}$. From the assumption, t_1, t_2, t_3, t_4 are distinct. Let c_2 be the centroid of t_1, t_2, t_3. Notice that $\{v_2, v_3\} \subset (B_{t_1} \cap B_{t_2})$ and that c_2 is on the path from t_1 to t_2. Also note that $v_4 \in (B_{t_2} \cap B_{t_3})$ and c_2 is on the path from t_2 to t_3. By the third condition of the definition of tree-decomposition, $\{v_2, v_3, v_4\} \subset B_{c_2}$. Since t_2 is the closest vertex to t_1 with this property, $c_2 = t_2$. Similarly, t_3 is the centroid of t_2, t_3, t_4. This gives that there is a path P in T from t_1 first through t_2, then through t_3, and ending at t_4.

Let t be the neighbor of t_3 in T such that t_1 and t_3 are in different components of $T - tt_3$. Let t_{new} be a vertex not in $V(T)$. Let $S := T - tt_3 + t_{new} + tt_{new} + t_{new}t_3$. Let P_1 be the component of $P - t_2$ containing t_1. Let P_2 be the component of $(P \setminus P_1) - t_3$ containing t_2. Let P_4 be the component of $P - t_3$ containing t_4. For every vertex $s \in V(T \setminus P)$, let $A_s := B_s \setminus \{v_3, v_4\}$. For every vertex $s \in V(P_1)$, let $A_s := B_s$. For every vertex $s \in V(P_2)$, let $A_s := (B_s \setminus \{v_4\}) \cup \{x\}$. Let $A_{t_{new}} := (B_{t_3} \setminus \{v_5\}) \cup \{x\}$. Let $A_{t_3} := (B_{t_3} \setminus \{v_3\}) \cup \{x\}$. For every vertex $s \in V(P_4)$, let $A_s := B_s \setminus \{v_3\}$. Notice

that for each edge of H incident to one of v_3, x, v_4, both of its ends are contained in one of the bags $A_{t_1}, A_{t_2}, A_{t_{new}}, A_{t_3}, A_{t_4}$. It is not hard to check, then, that (S, A) is a tree-decomposition of H of width at most four and that $H \in TW4$.

Assume $H \in TW4$. Let $M := H - (V(H) \setminus \{v_1, v_2, v_3, x, v_4, v_5\})$. Let J be obtained from H by replacing T_{DL} by S_{DL}, using $(M, (v_1, v_2, v_4, v_5))$ as the copy of T_{DL}. Thus $H \leq_{DL} J$, and from the three paragraphs above $J \in TW4$. Notice then that $G \leq_m J$, and $G \in TW4$, by Lemma 1. □

Now that the first Triangle Ladder reduction has been shown to be $TW4$-safe; this can be used to prove that its corresponding Ladder reduction is also $TW4$-safe. Let the Ladder reductions $L1$, $L2$, $L3$, $L4$ be defined by Fig. 5.

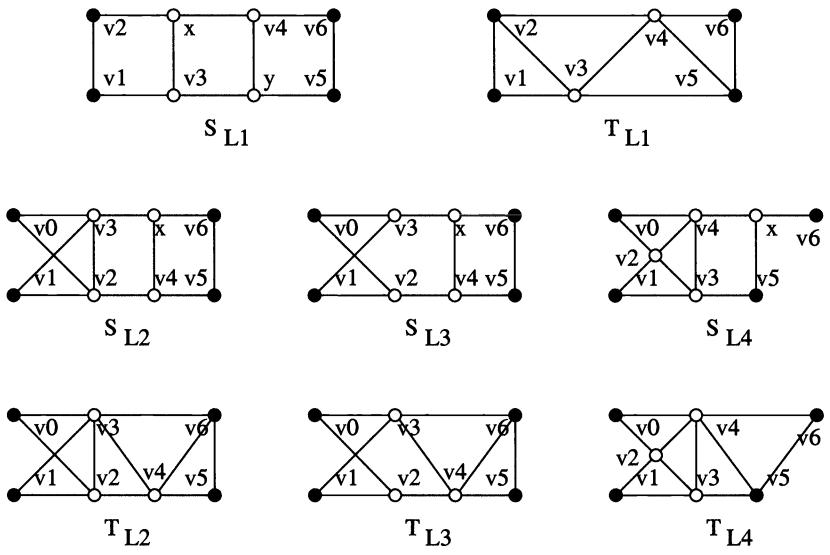


FIG. 5. *The Ladder reductions.*

LEMMA 12. *The reduction Ladder* 1 *(L1) is a TW4-safe* 4*-star reduction.*

Proof. Clearly $L1$ is a 4-star reduction. Let G, H be given satisfying $G \leq_{L1} H$. Without loss of generality, assume G is obtained from H by performing $L1$.

Assume $G \in TW4$. Since $G = H * x * y$, by Lemma 2, $H \in TW4$.

Assume $H \in TW4$. By Lemma 4, a chord of xv_3yv_4 may be added to H without increasing the tree-width. If $H + v_3v_4 \in TW4$, then $(H + v_3v_4) \cdot v_2x \cdot v_5y = G \in TW4$ by Lemma 1. If $H + xy \in TW4$, then $J := (H + xy) \cdot v_1v_3 \cdot v_4v_6 \in TW4$, and there is a K such that $J \leq_{DL} K$ (performing DL twice) and $G \leq_m K$; this shows $G \in TW4$ in this case, by Lemmas 1 and 11. □

LEMMA 13. *The reductions X Ladder* 1 *and* 2 *(XL1, XL2) are TW4-safe.*

Proof. Let G, H be given satisfying $G \leq_{XL1} H$ or $G \leq_{XL2} H$. Without loss of generality, assume G is obtained from H by performing $XL1$ or $XL2$. Let $k \in \{1, 2\}$ be such that G is obtained from H by performing XLk.

Assume $G \in TW4$. Let a tree-decomposition (T, B) of G of width four be given.

If there is a bag B_t with $|B_t \cap \{v_0, \ldots, v_6\}| \geq 4$, then $J := G * v_2 * v_3 * v_4 \leq_m G(T, B)$. This gives $J \in TW4$, by Lemma 1. Also, $H * v_2 * v_3 * x * v_4 = J$, so $J \leq_* H$,

and $H \in TW4$, by Lemma 2. Thus assume there is no such bag. By Lemma 10, there is a vertex s_0 of T such that $\{v_0, v_2, v_3\} \subset B_{s_0}$. By the same lemma, for $i := 1, \ldots, 4$, there are vertices s_i of T such that $\{v_i, v_{i+1}, v_{i+2}\} \subset B_{s_i}$. Let $t_0 := s_0$ and $t_1 := s_1$, but for $i := 2, 3, 4$ let t_i be the closest vertex to t_1 with $\{v_i, v_{i+1}, v_{i+2}\} \subset B_{t_i}$. From the assumption, t_0, t_1, t_2, t_3, t_4 are distinct. Let c_2 be the centroid of t_1, t_2, t_3. Notice that $\{v_2, v_3\} \subset (B_{t_1} \cap B_{t_2})$ and that c_2 is on the path from t_1 to t_2. Also note that $v_4 \in (B_{t_2} \cap B_{t_3})$ and c_2 is on the path from t_2 to t_3. By the third condition of the definition of tree-decomposition, $\{v_2, v_3, v_4\} \subset B_{c_2}$. Since t_2 is the closest vertex to t_1 with this property, $c_2 = t_2$. Similarly, t_3 is the centroid of t_2, t_3, t_4. This gives that there is a path P in T from t_1 first through t_2, then through t_3, and ending at t_4. Let Q be the path in T from t_0 to t_4.

Let T be the neighbor of t_3 in T such that t_1 and t_3 are in different components of $T - tt_3$. Let t_{new} be a vertex not in $V(T)$. Let $S := T - tt_3 + t_{new} + tt_{new} + t_{new}t_3$. Let $P_0 := Q \backslash P$. Let P_1 be the component of $P - t_2$ containing t_1. Let P_2 be the component of $(P \setminus P_1) - t_3$ containing t_2. Let P_4 be the component of $P - t_3$ containing t_4. For every vertex $s \in V(T \backslash (P \cup Q))$, let $A_s := B_s \backslash \{v_3, v_4\}$. For every vertex $s \in V(P_0 \cup P_1)$, let $A_s := B_s \setminus \{v_4\}$. For every vertex $s \in V(P_2)$, let $A_s := (B_s \setminus \{v_4\}) \cup \{x\}$. Let $A_{t_{new}} := (B_{t_3} \setminus \{v_5\}) \cup \{x\}$. Let $A_{t_3} := (B_{t_3} \setminus \{v_3\}) \cup \{x\}$. For every vertex $s \in V(P_4)$, let $A_s := B_s \setminus \{v_3\}$. Notice that for each edge of H incident to one of v_3, x, v_4, both of its ends are contained in one of the bags $A_{t_0}, A_{t_1}, A_{t_2}, A_{t_{new}}, A_{t_3}, A_{t_4}$. It is not hard to check, then, that (S, A) is a tree-decomposition of H of width at most four and that $H \in TW4$.

Assume $H \in TW4$. Let $M := G - (V(G) \setminus \{v_0, v_1, v_2, v_3, x, v_4, v_5\})$. Let J be obtained from H by replacing T_{XLk} by S_{XLk}, using $(M, (v_0, v_1, v_4, v_5))$ as the copy of T_{XLk}. Thus $H \leq_{XLk} J$, and from the three paragraphs above, $J \in TW4$. Notice then that $G \leq_m J$, and $G \in TW4$, by Lemma 1. □

LEMMA 14. *The reductions Ladder 2 and 3 (L2, L3) are TW4-safe 4-star reductions.*

Proof. Clearly $L2$ and $L3$ are 4-star reductions. Let G, H be given satisfying $G \leq_{L2} H$ or $G \leq_{L3} H$. Without loss of generality, assume G is obtained from H by performing $L2$ or $L3$.

Assume $G \in TW4$. Since $G = H * x$, by Lemma 2, $H \in TW4$.

Assume $H \in TW4$. By Lemma 4, or trivally, a chord of $v_0v_2v_1v_3$ may be added to H without increasing the tree-width. If $H + v_0v_1 \in TW4$, since $(H + v_0v_1) \cdot v_3x \cdot v_2v_4 \cdot xv_6 \cdot v_4v_5 = G * v_2 * v_3 * v_4$, it follows that $G \in TW4$ from Lemmas 1 and 2. Thus without loss of generality, $H + v_2v_3 \in TW4$. By Lemma 4, a chord of $v_3v_2v_4x$ may be added to H without increasing the tree-width. If $H + v_3v_4 \in TW4$, then let $J := (H + v_3v_4) \cdot v_6x$. If $H + v_2x \in TW4$, then let $J := (H + v_2x) \cdot v_4v_5$. In either case, there is a K such that $J \leq_{XL1} K$ and $G \leq_m K$. Thus $G \in TW4$, by Lemmas 1 and 13. □

LEMMA 15. *The reduction Star Ladder (SL) is TW4-safe.*

Proof. Let G, H be given satisfying $G \leq_{SL} H$. Without loss of generality, assume G is obtained from H by performing SL.

Assume $G \in TW4$. Let a tree-decomposition (T, B) of G of width four be given.

If there is a bag B_t with $|B_t \cap \{v_0, \ldots, v_6\}| \geq 4$, then $J := G * v_2 * v_3 * v_4 \leq_m G(T, B)$. This gives $J \in TW4$, by Lemma 1. Also, $H * v_2 * v_3 * x * v_4 = J$, so $J \leq_* H$, and $H \in TW4$, by Lemma 2. Thus assume there is no such bag. By Lemma 10, there is a vertex s_0 of T such that $\{v_0, v_2, v_4\} \subset B_{s_0}$. By the same lemma, for $i := 1, \ldots, 4$, there are vertices s_i of T such that $\{v_i, v_{i+1}, v_{i+2}\} \subset B_{s_i}$. Let $t_0 := s_0$ and $t_1 := s_1$, but for $i := 2, 3, 4$ let t_i be the closest vertex to t_1 with $\{v_i, v_{i+1}, v_{i+2}\} \subset B_{t_i}$. From

the assumption, t_0, t_1, t_2, t_3, t_4 are distinct. Let c_2 be the centroid of t_1, t_2, t_3. Notice that $\{v_2, v_3\} \subset (B_{t_1} \cap B_{t_2})$ and c_2 is on the path from t_1 to t_2. Also note that $v_4 \in (B_{t_2} \cap B_{t_3})$ and c_2 is on the path from t_2 to t_3. By the third condition of the definition of tree-decomposition, $\{v_2, v_3, v_4\} \subset B_{c_2}$. Since t_2 is the closest vertex to t_1 with this property, $c = t_2$. Similarly, t_3 is the centroid of t_2, t_3, t_4. This gives that there is a path P in T from t_1 first through t_2, then through t_3, and ending at t_4. Let Q be the path in T from t_0 to t_4.

Let t be the neighbor of t_3 in T such that t_1 and t_3 are in different components of $T - tt_3$. Let t_{new} be a vertex not in $V(T)$. Let $S := T - tt_3 + t_{new} + tt_{new} + t_{new}t_3$. Let $P_0 := Q \setminus P$. Let P_1 be the component of $P - t_2$ containing t_1. Let P_2 be the component of $(P \setminus P_1) - t_3$ containing t_2. Let P_4 be the component of $P - t_3$ containing t_4. For every vertex $s \in V(T \setminus (P \cup Q))$, let $A_s := B_s \setminus \{v_3, v_4\}$. For every vertex $s \in V(P_0)$, let $A_s := B_s \setminus \{v_3\}$. For every vertex $s \in V(P_1)$, let $A_s := B_s$. For every vertex $s \in V(P_2)$, let $A_s := (B_s \setminus \{v_4\}) \cup \{x\}$. Let $A_{t_{new}} := (B_{t_3} \setminus \{v_5\}) \cup \{x\}$. Let $A_{t_3} := (B_{t_3} \setminus \{v_3\}) \cup \{x\}$. For every vertex $s \in V(P_4)$, let $A_s := B_s \setminus \{v_3\}$. Notice that for each edge of H incident to one of v_3, x, v_4, both of its ends are contained in one of the bags $A_{t_0}, A_{t_1}, A_{t_2}, A_{t_{new}}, A_{t_3}, A_{t_4}$. It is not hard to check, then, that (S, A) is a tree-decomposition of H of width at most four and that $H \in TW4$.

Assume $H \in TW4$. Let $M := G - (V(G) \setminus \{v_0, v_1, v_2, v_3, x, v_4, v_5\})$. Let J be obtained from H by replacing T_{SL} by S_{SL}, using $(M, (v_0, v_1, v_4, v_5))$ as the copy of T_{SL}. Thus $H \leq_{SL} J$, and from the three paragraphs above, $J \in TW4$. Notice then that $G \leq_m J$, and $G \in TW4$, by Lemma 1. □

LEMMA 16. *The reduction Ladder 4 (L4) is a TW4-safe 4-star reduction.*

Proof. Clearly $L4$ is a 4-star reduction. Let G, H be given satisfying $G \leq_{L4} H$. Without loss of generality, assume G is obtained from H by performing $L4$.

Assume $G \in TW4$. Since $G = H * x$, by Lemma 2, $H \in TW4$.

Assume $H \in TW4$. By Lemma 4, a chord of $v_4v_3v_5x$ may be added to H without increasing the tree-width. If $H + v_4v_5 \in TW4$, then, since $G = (H + v_4v_5) \cdot v_6x$, this gives $G \in TW4$, by Lemma 1. If $H + v_3x \in TW4$, then let $J := (H + v_3x) \cdot v_6x$; thus, $J \in TW4$, by Lemma 1. In this case, there is a K such that $J \leq_{SL} K$ and $G \leq_m K$. Thus $G \in TW4$, by Lemmas 1 and 15. □

5. Superstructures. This section describes the more complicated reductions that are used in the linear algorithm. The reductions are described in terms of leaf structures, or the possible structures at the leaves of a tree-decomposition that are not reducible. The *leaf structures* are 60 simple structures, $LS_0, \ldots, LS_{59}$, and four infinite families of structures, $L_{1,n}, L_{2,n}, L_{3,n}, L_{4,n}$. Of the 60 simple structures, only 20 are central to the description of all 60. Of these 20, 13 are substructures of the Y-Δ reductions described in §§3 and 4. Thus, in some sense, the algorithm must search for only 11 new structures besides the eight structures for the Y-Δ reductions. Let K_2 be the complete graph on x, y. Let $LS_0 := (K_2, (x, y))$. Let $LS_1, \ldots, LS_{13}$, LS_{16}, LS_{25}, LS_{37}, LS_{46}, LS_{50}, LS_{55} be defined by Fig. 6.

For a structure S with vertex of attachment u_j of degree 2, define $S + j$ as follows: Let x, y be the neighbors of u_j. Let q_j be a vertex not in $V(S)$. Let G_S be the graph associated with S, and let P_S be the ordered list of vertices of attachment. Then $S + j := ((G_S + q_j) - u_jx - u_jy + q_ju_j + q_jx + q_jy, P_S)$. Then the remaining LS_i are defined as follows: $LS_{14} := LS_{13} + 1$, $LS_{15} := LS_{14} + 3$. $LS_{17} := LS_{16} + 1$, $LS_{18} := LS_{16} + 2$, $LS_{19} := LS_{17} + 3$, $LS_{20} := LS_{18} + 4$, $LS_{21} := LS_{18} + 1$, $LS_{22} := LS_{19} + 2$, $LS_{23} := LS_{20} + 1$, $LS_{24} := LS_{22} + 4$. $LS_{26} := LS_{25} + 1$, $LS_{27} := LS_{25} + 2$, $LS_{28} := LS_{25} + 4$, $LS_{29} := LS_{26} + 3$, $LS_{30} := LS_{27} + 1$, $LS_{31} := LS_{26} + 4$, $LS_{32} :=$

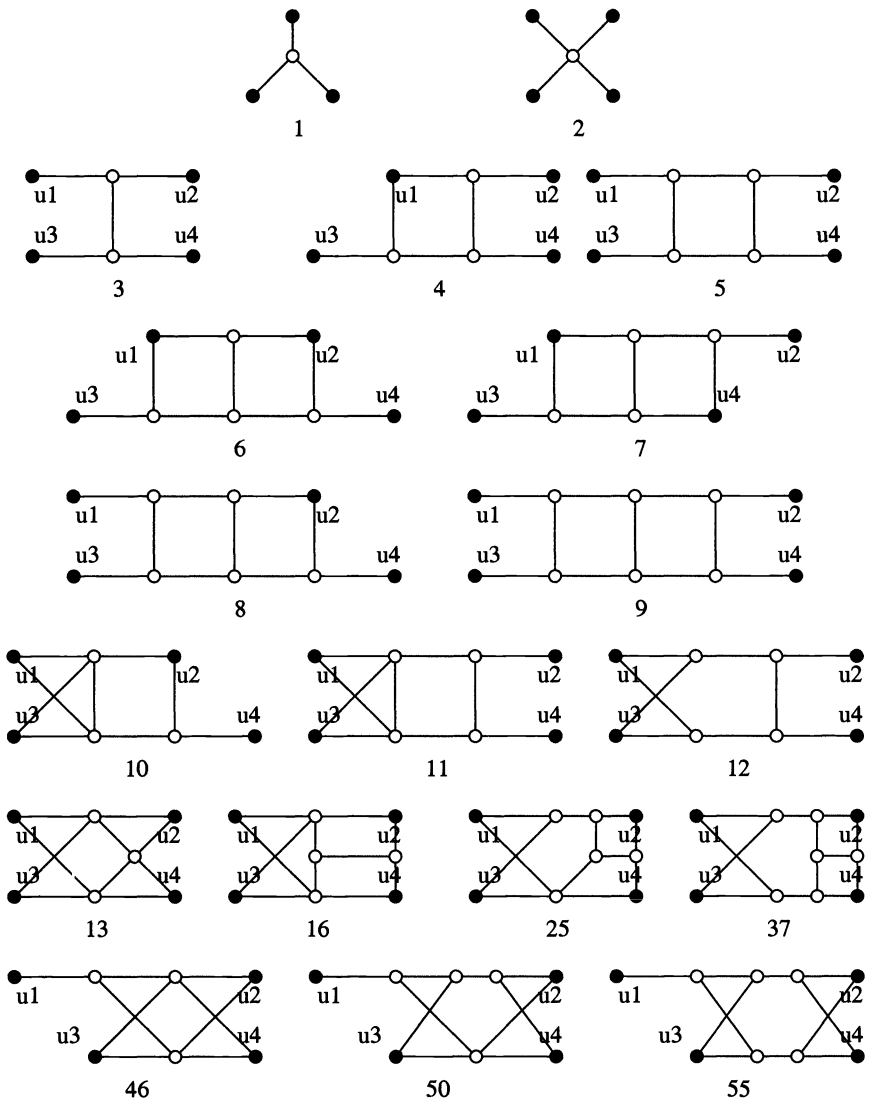


FIG. 6. *The central leaf structures.*

$LS_{27}+4$, $LS_{33} := LS_{29}+2$, $LS_{34} := LS_{29}+4$, $LS_{35} := LS_{30}+4$, $LS_{36} := LS_{33}+4$. $LS_{38} := LS_{37}+1$, $LS_{39} := LS_{37}+2$, $LS_{40} := LS_{38}+3$, $LS_{41} := LS_{38}+2$, $LS_{42} := LS_{39}+4$, $LS_{43} := LS_{40}+2$, $LS_{44} := LS_{41}+4$, $LS_{45} := LS_{43}+4$. $LS_{47} := LS_{46}+2$, $LS_{48} := LS_{47}+3$, $LS_{49} := LS_{48}+4$. $LS_{51} := LS_{50}+2$, $LS_{52} := LS_{50}+3$, $LS_{53} := LS_{51}+3$, $LS_{54} := LS_{53}+4$. $LS_{56} := LS_{55}+2$, $LS_{57} := LS_{55}+3$, $LS_{58} := LS_{56}+3$, $LS_{59} := LS_{58}+4$.

The four infinite families are indicated in Fig. 7. The n tells how many vertices are in the structure: $|V(L_{1,n})| = n+5$, $|V(L_{2,n})| = n+5$, $|V(L_{3,n})| = n+6$, $|V(L_{4,n})| = n+6$. Notice that each leaf structure has no edge between any two of its vertices of attachment. This is important for the linear algorithm.

Let a *superstructure* be a structure S satisfying the following. The set A of vertices

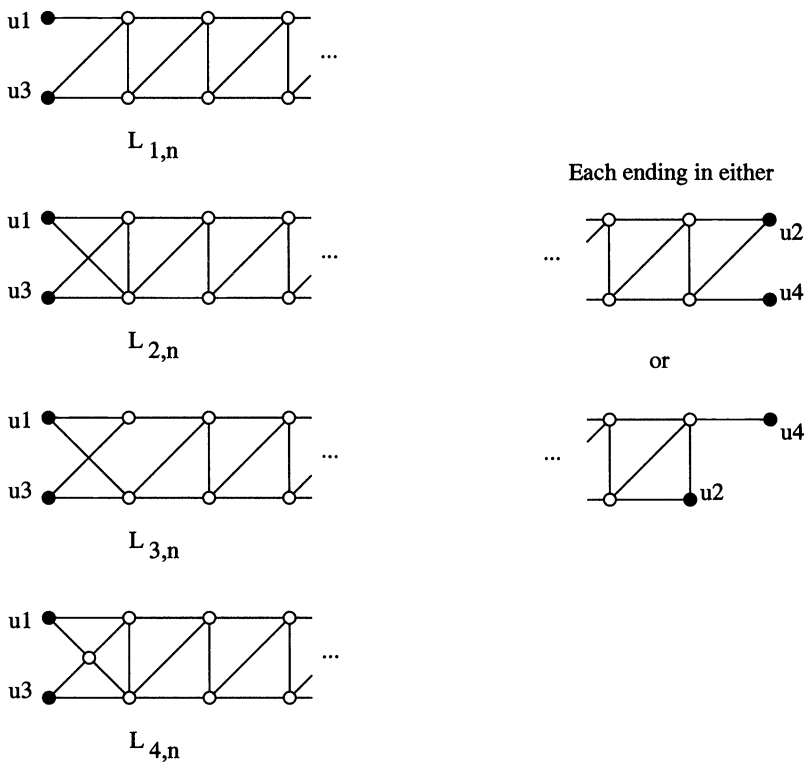


FIG. 7. *The infinite classes of leaf structures.*

of attachment of S is of size at most four. There is a *center* vertex x of $G(S)$ that is not in A. The graph of S is the union of the graphs of a finite set L of *substructures* of S, which are leaf structures, such that for each $M \in L$, the set B of vertices of attachment of M satisfies $\{x\} \subset B \subset A \cup \{x\}$.

Let CM (Clique Minor) be the set of all reductions R that satisfy the following. For each $R \in CM$, the structure S_R is a superstructure, T_R is a complete graph with each of its vertices a vertex of attachment, and either $G(T_R) \leq_m G(S_R)$ or there is some J such that $G(T_R) \leq_{SO} J \leq_m G(S_R)$. Notice that Cube $\in CM$. Let Triple be the set of all reductions satisfying the following. For each $R \in$ Triple, $T_R = (K_4, (a, b, c, d))$. The graph $G(S_R)$ is the union of the graphs of three leaf structures, each having vertices of attachment a subset of $\{a, b, c, d\}$, none of which is LS_0 and no two of which are LS_1 with the same three vertices of attachment. Let BCM (bounded CM) be the set of all reductions satisfying the following. For each $R \in BCM$, then $R \in CM$, and S_R does not contain a Buddy or a reduction in Triple. Note that the center of each $R \in BCM$ has degree at most 20. This is important for the linear algorithm.

LEMMA 17. *Each reduction in* $CM \cup$ Triple *is a* $TW4$*-safe* 4*-star reduction.*

Proof. Let $R \in CM \cup$ Triple be given. Let graphs G, H be given such that $G \leq_R H$. All the leaf structures are such that it is easy to see $G \leq_* H$. Thus R is a 4-star reduction. If $R \in CM$, then the conclusion follows from the definition of

CM and Lemmas 3 and 5. If $R \in$ Triple, then notice first that $G \leq_m H$; then the conclusion follows from Lemma 3. □

This concludes the list of reductions needed for the linear algorithm. Each reduction is a $TW4$-safe 4-star reduction; thus, given a graph G, if there is a sequence of reductions that takes G to the empty graph, then G has tree-width at most four. What remains is to show the converse, that is, that for every graph G of tree-width at most four, there is a sequence of these reductions that takes G to the empty graph. Each of these statements is contained in the following theorem, which is proved in [23] by a long case analysis that shows every superstructure is either a leaf structure or contains a reduction in $CS4$ (Complete and Safe – 4) := $EZ \cup CM \cup \{\text{Buddy}\}$, where $EZ := \{\text{Zero}, \text{One}, \text{Series}, \text{Triangle}, YO, H7, TO, YI, L1, L2, L3, L4\}$.

THEOREM 1. *The set $CS4$ of reductions is a $TW4$-complete set of $TW4$-safe 4-star reductions.*

Proof. That each of the reductions is a $TW4$-safe 4-star reduction follows from Lemmas 3, 6, 7, 8, 9, 12, 14, 16, and 17. To see that the set is $TW4$-complete, see [23]. □

The set $CS4$ is used to show a practical linear algorithm in §6. The list $CS4$ is infinite, and certainly one would hope to have a finite list of reductions to use. If 4-star reductions are abandoned, a finite $TW4$-complete set of $TW4$-safe reductions can be found (see [23]) simply by adding the reduction DL to the list. This eliminates the need for the infinite families of leaf structures, as all sufficiently large structures in these families contain a DL. The reduction DL, on the other hand, does not apply well to algorithms. The following theorem shows that in this case, the best of both worlds is not possible. In the statement of the theorem, TWk and k-star reductions are the natural generalizations of $TW4$ and 4-star reductions.

THEOREM 2 (Lagergren [15]). *For each integer $k \geq 4$, there is no TWk-complete finite set of TWk-safe k-star reductions.*

6. A linear algorithm to find 4-elimination sequences. This section presents an algorithm that, given a graph G with n vertices, either finds a 4-elimination sequence for G or proves that the tree-width of G is greater than four. The algorithm proceeds by finding one of the reductions in $CS4$. Note that if G has a reduction in CM, then it either has a reduction in BCM or Triple or a Buddy. The algorithm will find one of the latter and obtain a smaller graph by performing that reduction and then find a reduction in that graph, and so forth. Since the reductions in $CS4$ require adding edges to the graph, for the algorithm to be linear, it must address multigraphs. The problems associated with this are addressed in the following lemma.

LEMMA 18. *Given a multigraph G and a vertex x of G, there is an algorithm that determines whether x has degree at most k in the underlying simple graph of G, deletes some number m of multiple edges incident with x, and takes time $O(k^2 + km)$.*

Proof. Clear an array A of k elements. Scan the adjacency list of x. For the next edge α in the adjacency list, do the following: Let y be the vertex incident with α such that $x \neq y$. If y is already in A, then xy is a multiple edge; delete α. Else, if A is full, then x has degree greater than k in the underlying simple graph of G; halt. If neither of the previous cases occur, add y to A. If there are no more edges in the adjacency list, then x has degree at most k in G; halt. □

The reason that the lemma is useful is the following. First, at most n reductions are performed on G. Next, note that each reduction adds at most six edges to G. Thus the sum of the ms mentioned in the lemma over the whole algorithm is at most $6n$. Finally, the lemma will only be applied when k is at most six. The edges that

will be sought in G will have at least one end of degree at most six and, thus, using multigraphs will not be a problem for this algorithm.

The algorithm will need a subroutine that determines whether a vertex is the center of a reduction in CM. This subroutine will either determine that the vertex cannot be the center of a reduction in CM or find a Buddy or a reduction in BCM or Triple. Before the statement of the subroutine, some variables need to be defined. Let VS be the Vertex Stack, used to keep track of vertices that need to be examined. For each vertex x in G, let $L(x)$ be the list of all leaf structures found that contain x as an internal vertex (not a vertex of attachment), where only the $L_{i,n}$ for the largest n is stored. It can be shown that for every vertex x in G, $|L(x)| \leq 4$. For a structure S, let $N(S)$ be the set of vertices of attachment of S. The parameters of the algorithm are as follows: a is the possible center, A is a set of possible vertices of attachment, and S is a set of possible substructures. The statement **return** sends the algorithm out of the most recent call; the statement **master-return** breaks the algorithm out of every recursive call.

ALGORITHM 1 **CENTER_CHECK**(a, A, S).

begin
if $|A| > 4$
then return
if every edge in a's adjacency list has been scanned
then if $\bigcup S$ is not a leaf structure
then perform the reduction in BCM
push its vertices of attachment onto VS
master-return
else mark every edge in a's adjacency list as not scanned
return
repeat
let α be the next unscanned edge in a's adjacency list, incident with $b \neq a$
if an edge from a to b has been scanned before
then delete the multiple edge
until α is not a multiple edge
mark α scanned
let K be the graph with vertices a, b and edge α
CENTER_CHECK$(a, A \cup \{b\}, S \cup \{(K, (a, b))\})$
mark α unscanned
for each $L \in L(b)$ with $a \in N(L)$ **do**
if L is isomorphic to LS_1 **and** there is an $M \in S$ with $N(M) = N(L)$
then perform the Buddy
push $N(L)$ onto VS
master-return
for each pair P, Q of nontrivial leaf structures (not L_0) in S **do**
if $|N(L) \cup N(P) \cup N(Q)| = 4$
then perform the reduction in Triple
push its vertices of attachment onto VS
master-return
mark the edges of $G(L)$ incident with a as scanned
CENTER_CHECK$(a, A \cup (N(L)\{a\}), S \cup \{L\})$
mark the edges of $G(L)$ incident with a as not scanned
return
end

The main algorithm must perform a number of functions. It must search for the reductions in EZ. For each vertex x, it must keep track of what leaf structures contain x as an internal vertex. Whenever a reduction R is performed, these lists, the $L(x)$'s, must be updated for the vertices of attachment of R. It is not hard to see that this may be done in constant time amortized, including possibly replacing an old $L_{i,n}$ with the currently known maximal such leaf structures. Finding the maximal $L_{i,n}$ may require looking at a large number j of vertices initially, but just note that the algorithm is at the same time determining the set of maximal leaf structures for those j vertices as well. The main algorithm is Algorithm 2. An easy proposition that follows from the definition of tree-width is that if the tree-width of G is at most k, then $|E(G)| \leq k|V(G)|$.

ALGORITHM 2 **TREE-WIDTH-FOUR?**(G).
begin
if $|E(G)| > 4|V(G)|$
then output "The tree-width of G is greater than four"
halt
$H := G$.
initialize VS to the empty stack
for each vertex v of H **do**
push v onto VS
$L(v) := \emptyset$
for each edge α of H **do**
mark α as not scanned
repeat
pop a vertex x from VS.
if $\deg(x) \leq 6$
then if x is an interior vertex of some reduction in EZ
then perform the reduction to H
push its vertices of attachment onto VS.
else determine the leaf structures $LS_1, \ldots, LS_{59}$ and the maximal structures $L_{i,n}$ with x an interior vertex by examining the neighbors of any $L_{i,n}$ in $L(x)$, and then of x if necessary
for each new leaf structure L found which is not in $L(x)$ **do**
$L(x) := L(x) \cup \{L\}$
for each vertex of attachment z of L **do**
mark the edges from z into L as scanned
CENTER_CHECK$(z, N(L) \setminus \{z\}, \{L\})$
mark the edges from z into L as not scanned
CENTER_CHECK$(x, \emptyset, \emptyset)$
until $VS = \emptyset$
if $H = K_0$ (the empty graph)
then $G \in TW4$; output the order that vertices were eliminated
else $G \notin TW4$; output "The tree-width of G is greater than 4"
end

THEOREM 3. *The algorithm TREE-WIDTH-FOUR? determines a 4-elimination sequence or correctly says the graph has tree-width greater than four in linear time.*

Proof. The proof of the correctness of the algorithm is basically Theorem 1, which gives that every graph in $TW4$ has a reduction in $CS4$. It must be shown that the algorithm is able to find such a reduction at each stage of the algorithm. If the

reduction existed in G itself, it will be found, as every vertex is pushed onto VS at the beginning of the algorithm. Thus the reduction was created after the performance of some other reduction. Each time a reduction is performed, however, the affected vertices are pushed onto VS. Thus the algorithm can always find a reduction if one is present. The proof of linearity remains.

First, each main call (not recursive call) to CENTER_CHECK will be shown to take constant time. This is seen by noticing that at most 20 edges are scanned. This is clear from the definitions of Buddy, Triple, and BCM and from Theorem 1. If the vertex a examined has had 20 edges scanned, no Buddy or reduction in Triple has been found, and edges still remain to be examined, then a cannot be the center of a reduction in CM. Also, for each edge scanned, the edge is in a bounded number of appropriate leaf structures.

Updating $L(x)$ is constant time amortized. For all but the leaf structures $L_{i,j}$, Lemma 18 shows that it takes constant time to look for them. The initial instance where the $L_{i,j}$ is found requires $O(j)$ time. This cost can be split among the $O(j)$ vertices in the structure, for the algorithm is simultaneously finding $L(y)$ for each of the j vertices y in this $L_{i,j}$. Reductions that do not use the entire $L_{i,j}$ affect only a bounded number of its vertices. Thus subsequent updates require only analysis at the vertices of attachment of the structure. Again, this is a constant-time operation.

Searching for a reduction in EZ takes constant time per vertex by Lemma 18. Performing a reduction takes only a constant amount of work on the vertices of attachment of that reduction. The rest of the work is deleting the interior vertices in an elimination sequence, which clearly only requires a total linear time for the whole algorithm, as each vertex is only eliminated once. As mentioned in the discussion after Lemma 18, the total amount of work required to delete multiple edges is linear.

All that remains is to put a linear bound on the number of iterations of the main algorithm, i.e., or the number of vertices that are pushed onto VS. Vertices are only pushed onto VS in two instances: At initialization each vertex is pushed onto VS once. After each reduction, at most four vertices are pushed onto VS. Clearly, at most n reductions are performed; thus at most $5n$ vertices are pushed onto VS. The total cost of the algorithm is then linear, noting that the test to determine if $|E(G)| > 4|V(G)|$ is linear as well. □

7. Conclusion. This paper presented a linear algorithm to find a 4-elimination sequence or to prove that no such sequence exists. This algorithm has no large hidden constants and, thus, should be practical. This was made possible through the reductions found that were safe and complete for all graphs of tree-width at most four. This algorithm then, combined with the algorithms of Arnborg and Proskurowski, show that the NP-hard problems of Hamilton circuit, maximum independent set, as well as others, can be solved practically in linear time for graphs of tree-width at most four.

Arnborg, Corneil, and Proskurowski [4] (independently [24]) used the reductions found by Arnborg and Proskurowski to determine the four minimal forbidden minors for tree-width at most three. Clearly there is a relationship between the reductions presented in this paper and the minimal forbidden minors for tree-width at most four. In [23], over 75 minimal forbidden minors for tree-width at most four of widely varying structures are presented. This gives some evidence that using a large number of reductions is necessary for this type of algorithm. Conversely, the complete list of minimal forbidden minors could be determined from the list of reductions in this paper.

REFERENCES

[1] S. ARNBORG, *Reduced state enumeration: Another algorithm for reliability evaluation*, IEEE Trans. Reliability, 27 (1978), pp. 101–105.

[2] ———, *Efficient algorithms for combinatorial problems on graphs with bounded decomposability —a survey*, BIT, 25 (1985), pp. 2–23.

[3] S. ARNBORG, D. G. CORNEIL, AND A. PROSKUROWSKI, *Complexity of finding embeddings in a k-tree*, SIAM J. Alg. Discrete Methods, 8 (1987), pp. 277–284.

[4] ———, *Forbidden minors characterization of partial 3-trees*, Discrete Math., 80 (1990), pp. 1–19.

[5] S. ARNBORG, J. LAGERGREN, AND D. SEESE, *Problems easy for tree-decomposable graphs*, J. Algorithms, 12 (1991), pp. 308–340.

[6] S. ARNBORG AND A. PROSKUROWSKI, *Characterization and recognition of partial 3-trees*, SIAM J. Alg. Discrete Methods, 7 (1986), pp. 305–314.

[7] ———, *Linear time algorithms for NP-hard problems restricted to partial k-trees*, Discrete Appl. Math., 23 (1989), pp. 11–24.

[8] H. L. BODLAENDER, *Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees*, J. Algorithms, 11 (1990), pp. 631–643.

[9] ———, *A linear time algorithm for finding tree-decompositions of small treewidth*, submitted, September 1992.

[10] H. L. BODLAENDER AND T. KLOKS, *Better algorithms for the pathwidth and treewidth of graphs*, 18th International Colloquium on Automata, Languages, and Programming, 1991, pp. 544–555; for a more complete version (40 pages): *Efficient and constructive algorithms for the pathwidth and treewidth of graphs*, submitted.

[11] B. COURCELLE, *The monadic second-order logic of graphs* III: *Tree-decompositions, minors and complexity issues*, Inform. Théorique et Appl., 26 (1992), pp. 257–286.

[12] N. DEAN, private communication.

[13] S. T. HEDETNIEMI, *Bibliography of algorithms on partial k-trees*, Ann. Discrete Math., to appear.

[14] A. ISHIZUKA, Y. KAJITANI, AND S. UENO, *Characterization of partial 3-trees in terms of three structures*, Graphs Combin., 2 (1986), pp. 233–246.

[15] J. LAGERGREN, *The nonexistence of reduction rules giving an embedding into a k-tree*, Discrete Appl. Math., 54 (1994), pp. 219–223.

[16] E. MATA-MONTERO, *Resilience of partial k-tree networks with edge and node failures*, Networks, 21 (1991), pp. 321–344.

[17] J. MATOUŠEK AND R. THOMAS, *Algorithms finding tree-decompositions of graphs*, J. Algorithms, 12 (1991), pp. 1–22.

[18] ———, *On the complexity of finding iso- and other morphisms for partial k-trees*, Discrete Math., 108 (1992), pp. 343–364.

[19] S. PARTER, *The use of linear graphs in Gauss elimination*, SIAM Rev., 3 (1961), pp. 119–130.

[20] B. A. REED, *Finding approximate separators and computing tree-width quickly*, Proc. Sympos. Theory Computing, 24 (1992), pp. 221–228.

[21] N. ROBERTSON AND P. D. SEYMOUR, *Graph Minors.* II. *Algorithmic aspects of tree-width*, J. Algorithms, 7 (1986), pp. 309–322.

[22] ———, *An outline of a disjoint paths algorithm*, in Paths, Flows, and VLSI-Layout, B. Korte, L. Lovasz, H. J. Promel, and A. Schrijver, eds., Algorithms and Combinatorics 9, Springer-Verlag, New York, 1990, pp. 267–292.

[23] D. P. SANDERS, *Linear Algorithms for Graphs of Tree-width at Most Four*, Ph.D. thesis, Program of Algorithms, Combinatorics, and Optimization, Georgia Institute of Technology, Atlanta, Georgia, June 1993.

[24] A. SATYANARAYANA AND L. TUNG, *A characterization of partial 3-trees*, Networks, 20 (1990), pp. 299–322.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 118–128, February 1996

011

CHIP-FIRING GAMES ON MUTATING GRAPHS*

KIMMO ERIKSSON†

Abstract. We investigate a generalization of a chip-firing game on a graph of Björner, Lovász, and Shor [*European J. Combin.*, 1 (1992), pp. 305–328]. In our version, the graph mutates during play. We show that some known results about the game length and period length of the earlier game hold for the mutating version as well, and some completely new bounds are also obtained. In a small detour, we treat an orientability concept for eulerian graphs.

Key words. chip firing, eulerian graph, period length, game length

AMS subject classifications. 90D43, 90D46, 05C20

1. Introduction.

This paper considers a generalization of the chip-firing game treated by Björner and Lovász [2]. For an account of the earlier history of this game we refer to [1], [2]. The game is played on a directed graph with some initial distribution of chips on its nodes. Whenever a node has at least as many chips as it has outgoing edges it may send one chip along each of these edges. This process is called *firing* the node.

Here we will consider a game where the graph is not fixed, but *mutating*, by which is meant that the outgoing edges of a node may be erased and new ones may be added after every firing of the node. These mutations happen in a predetermined order. This is a common generalization of the chip firing of Björner and Lovász and another game of Diaconis and Fulton [4], and we show that our new game also has the special convergence property shared by the earlier two.

A fruitful restriction is to demand that the predetermined order of mutations at every node has a finite period. We can extend several of the results in [2] to these periodically mutating games (PMG).

Section 3 covers finite termination. Björner and Lovász gave a lower bound on the number of chips needed for a game to be infinite, in terms of the maximal number of edge-disjoint cycles in the graph. We characterize the graphs for which this bound is sharp as eulerian graphs that are *orientable* in a sense we define.

In §4 we discuss period length and game length of the nonmutating game. A new bound on the maximal length of a terminating game in terms of the period length is proved. Here is also discussed parallel firings of the nodes, a topic initiated by Bitar and Goles [1] for the undirected chip-firing game.

Finally, §5 contains the generalizations to the PMG, the periodically mutating games.

1.1. Preliminaries.

In this paper, vectors are written boldfaced, while their entries are in italics, e.g., $\mathbf{v} = (v_1, v_2, \ldots)$. We adopt the following terminology from Björner and Lovász [2]. Graphs are assumed to be finite and directed with loops and multiple edges allowed. Undirected graphs are regarded as directed by replacing every undirected edge (i, j) by the two directed edges $(\overrightarrow{i,j})$ and $(\overrightarrow{j,i})$. For a digraph $G = (V, E)$ we let $n = |V|$ and $m = |E|$. Let d_{ij} denote the number of directed edges from i to j. For a node i, denote the outdegree by $d^+(i) = \sum_{j \in V} d_{ij}$ and the indegree by $d^-(i) = \sum_{j \in V} d_{ji}$. Let D be the maximum outdegree in G.

* Received by the editors November 16, 1992; accepted for publication (in revised form) March 8, 1995.

† Department of Mathematics, Royal Institute of Technology, S-100 44 Stockholm, Sweden.

G is *eulerian* if $d^+(i) = d^-(i)$ for every i. G is *strongly connected* if there is a directed path from i to j for every ordered pair of nodes i and j. A strongly connected component H that is not connected to the outside by any edge leaving H is a *sink component.*

2. Chip firing on mutating graphs. Let us first give an exact definition of the mutating game.

DEFINITION 2.1. *The* mutating game *is a solitary game played with chips distributed on V, the nodes of a* mutating graph. *A position in the game consists of*

- *a directed graph on V,*
- *on each node $i \in V$ a nonnegative number c_i of chips,*
- *for every node $i \in V$ an infinite sequence M_i of multisets of nodes in V—the* mutation sequence *of i.*

A node i may be fired if the number of chips on i is at least $d^+(i)$, the number of outgoing edges from i in the current directed graph, in which case one chip is sent from i along each outgoing edge. This may be written as

$$c_i := c_i - d^+(i)$$

and then

$$c_j := c_j + d_{ij} \quad \text{for all } j \in V.$$

Suppose that the mutation sequence is $M_i = (S_1, S_2, S_3, \ldots)$. After the chips have moved, the mutation of the graph takes place. This means that the outgoing edges from i are erased and a new edge $(\overrightarrow{i,j})$ is drawn for each j in S_1, the first multiset of M_i. Finally, this multiset is removed from the sequence, i.e., $M_i := (S_2, S_3, \ldots)$. In computer science, this is often referred to as "popping" M_i, so we denote this operation by

$$M_i := Pop(M_i)$$

The steps above constitute one firing *of i. The game continues from the new position that is obtained.*

Obviously, if every mutation adds the same edges that it erases, so that the graph does not change, then this is equivalent to the chip-firing game of Björner and Lovász. Another chip-firing game, motivated by algebraic considerations, was defined by Diaconis and Fulton [4] as follows: to every node i is assigned a threshold $l(i)$ (positive integer) and an unlimited deck of cards, such that whenever a node i has at least $l(i)$ chips it may send one chip to the node indicated by the top card of its deck, and then this card is thrown away. Also this game can be interpreted as a mutating game, where every node i has a fixed number $l(i) - 1$ of loops and one additional outgoing edge mutating according to the deck.

Known for both these earlier games is a special convergence property, for which we now give a simple combinatorial proof for the mutating game in general. A position is *terminal* if no node can be fired.

THEOREM 2.2. *Given an initial distribution of chips in a mutating game, either every firing sequence can be continued indefinitely or else every firing sequence must terminate after the same number of moves with the same terminal position. The number of times a given node has been fired in a terminating game is independent of the particular firing sequence.*

We shall prove this via a basic lemma. First observe that since the mutation sequence at any node is predetermined, the flow of chips from any node is determined solely by how many times it is fired. Thus, the resulting position of a firing sequence is dependent only on the number of times that each particular node is fired. For a firing sequence α, define the *score vector* $[\alpha] \in \mathbb{N}^V$ where each entry $[\alpha]_i$ is the number of times that node i is fired in α.

LEMMA 2.3. *If α and β are firing sequences, then α can be extended by some subword of β to some sequence α' such that $[\alpha']_i = \max([\alpha]_i, [\beta]_i)$ for every i.*

Proof. Let $\alpha' := \alpha$. If $[\alpha']_i \geq [\beta]_i$ for all i, we are done. Otherwise, let i be the first node when playing β that is played for the $([\alpha']_i + 1)$th time. Since i is firable at this point in β, it must also be firable after α' since the flow of chips *from* i has been the same while the flow of chips *to* i has been at least as great in α'. Extend α' by i, i.e., $\alpha' := \alpha' i$, and repeat the argument. □

Proof of Theorem 2.2. Suppose there is some firing sequence α leading to a terminal position. Then α cannot be extended, so it follows from Lemma 2.3 that every legal firing sequence β must have $[\beta]_i \leq [\alpha]_i$ for every node i and that every such β can be extended to some β' such that $[\beta'] = [\alpha]$. This completes the proof. □

Remark 2.4. A similar result with a similar proof was found independently by Thorup [7].

In the rest of this paper we will be interested in questions connected with periodicity of the game. Since a position in the mutating game is defined by the current graph, the chips distribution, and the mutation sequence of every node, the game can have a finite period length only if every node's mutation sequence has a finite period. If this is the case, then we say that the game is *periodically mutating* (a PMG). Denote by p_i the mutation period length at node i, so p_i is the smallest positive integer such that after i is fired p_i times the remaining mutation sequence at i is the same as it was from the beginning. In other words, p_i is the smallest positive integer such that M_i satisfies $M_i = \mathrm{Pop}^{p_i}(M_i)$. Concretely, M_i looks like $(S_1, S_2, \ldots, S_{p_i}, S_1, S_2, \ldots, S_{p_i}, \ldots)$.

To any PMG we can associate a *great graph GG* with the same node set as the PMG and where the number of edges $(\overrightarrow{i,j})$ is the total number of such edges occurring during one complete mutation period at i, i.e., the number of occurrences of j in the multiset union $S_1 \cup S_2 \cup \cdots \cup S_{p_i}$. The motivation for this is of course that the flow of chips from i during p_i firings of i in the PMG is equal to the flow of chips from i when firing i once in the ordinary chip-firing game (CFG) played on GG. So the CFG on the great graph GG will be a tool in the analysis of the PMG.

Note that a CFG played on a directed graph G, when viewed as a PMG, has all $p_i = 1$ and great graph G. A way to look at a PMG is by marking each outgoing edge from every node i in GG by an element of $\mathbb{Z} \bmod p_i$ and letting i have a counter $\kappa_i \in \mathbb{Z} \bmod p_i$, such that firing node i means sending one chip along each edge marked κ_i and then setting $\kappa_i := \kappa_i + 1 \bmod p_i$.

3. Finite termination. Björner and Lovász [2] studied the amount of chips possible for terminating games. The following lemma of theirs can be lifted directly to PMG.

LEMMA 3.1. *In every infinite game on a* PMG, *every node in some sink component of GG is fired infinitely often.*

Proof. In an infinite game, some node i must be fired infinitely often. Hence an infinite number of mutation periods are played at i, so every j such that there is

an edge $(\overrightarrow{i,j})$ in GG must also be fired infinitely many times, because otherwise an infinite number of chips would pile up on j. Hence, all nodes reachable from i, which must include some sink component, are fired infinitely often. □

From [2] we can also borrow the following bound: if GG has n nodes and m edges and we have $N > m - n$ chips, then the game is infinite, since by the pigeon-hole principle some node can always be fired. One might thus be tempted to believe that just about anything carries over trivially from CFG to PMG. This is false, though. For example, Björner and Lovász argue that any CFG with fewer than h chips is finite, where h is maximal such that every sink component of G has h edge-disjoint directed cycles. However, one can construct a PMG with arbitrary great graph GG such that the game is infinite with one single chip! Just choose the mutations so that every node has only one outgoing edge at any time.

Remark 3.2. One can make a nice observation in this context. Suppose we have a PMG where exactly one edge is active at every firing and whose great graph is eulerian, so $p_i = d^+(i) = d^-(i)$ for every i. Place one chip on some node j and play the game. Then the first node i to be fired for the $(p_i + 1)$th time must be j, because any other node i must first receive the chip for the $(p_i + 1)$th time, which implies that some of its in-edges $(\overrightarrow{i',i})$ have been active twice, and hence i' must have been fired $p_{i'} + 1$ times already.

This property will be generalized to any great graph in §5.

We now discuss in more detail the result in [2] that says that a strongly connected graph G with h edge-disjoint directed cycles cannot have an infinite CFG with less than h chips. The proof is based on the idea that every cycle may "capture" any chip that travels along any of its edges so that in the future the captured chip only travels along the cycle that has captured it. This is in accordance with the rules, since the cycles are edge-disjoint. Every node is fired infinitely often in an infinite CFG on a strongly connected graph G, so every edge is used infinitely often and hence every cycle must capture at least one chip.

If G is not eulerian, then there must be some edges that are not contained in any of the h cycles, and hence it takes *more* than h chips to get an infinite game on such a G, since these excess edges cannot use the captured chips (except possibly a finite number of times before the chips have been captured). We shall investigate the possibility of an infinite game with h chips when G really is eulerian.

DEFINITION 3.3. *For a connected eulerian graph G, let $h = h(G)$ be the maximal number of edge-disjoint directed cycles in G. If $\mathcal{C} = \{C_1, C_2, \ldots, C_h\}$ is such a maximal family of edge-disjoint directed cycles, then G is* orientable relative $\mathcal{C}$ *if there is an ordering $x_1, x_2, \ldots, x_n$ of the nodes of G—an* orientation *of G—such that every $C \in \mathcal{C}$ is a cycle $(x_{i_1} x_{i_2} \cdots x_{i_k})$ with $i_1 < i_2 < \cdots < i_k$.*

Observe that every edge of G must be contained in some $C \in \mathcal{C}$ since G is eulerian and $\mathcal{C}$ is maximal and that every $C \in \mathcal{C}$ visits any node at most one time. Thus, for every node $x \in G$ the outdegree $d^+(x)$ is equal to the number of cycles $C \in \mathcal{C}$ that contain x.

LEMMA 3.4. *Let G be a connected eulerian graph and $\mathcal{C} = \{C_1, C_2, \ldots, C_h\}$ some maximal family of edge-disjoint directed cycles. Then G has an infinite* CFG *with h chips if and only if G is orientable relative $\mathcal{C}$.*

Proof. ($\Leftarrow$): Suppose that $x_1, x_2, \ldots, x_n$ make up an orientation relative $\mathcal{C}$. For every $i = 1, 2, \ldots, n$, let B_i be the subset of $\mathcal{C}$ consisting of all cycles for which x_i is the vertex of least index and let A_i be the set of all other cycles in $\mathcal{C}$ containing x_i. Thus, every cycle in $\mathcal{C}$ containing x_i is either in A_i or in B_i, so $d^+(x_i) = |A_i| + |B_i|$. Define an

initial position on G where every node x_i has $|B_i|$ chips. Since $|B_1|+|B_2|+\cdots+|B_n| = |C| = h$, this is a position with h chips. We shall see that it is possible to fire the nodes in the order $x_1, x_2, \ldots, x_n$. Since such a firing sequence would give back the initial position, we will then have proved that the game is infinite.

First, x_1 is firable because $d^+(x_1) = |B_1|$ since $A_1 = \emptyset$. Now suppose that $x_1, x_2, \ldots, x_{i-1}$ has been fired. Then x_i has received $|A_i|$ chips, one for each $C \in A_i$ due to the edge $(\overrightarrow{x_j, x_i}) \in C, j < i$. Hence, on the node x_i are now $|A_i|+|B_i| = d^+(x_i)$ chips, so x_i is firable.

($\Rightarrow$): Suppose that there is an infinite firing sequence α on G with h chips. Let $x_1, x_2, \ldots, x_n$ be the order in which the nodes occur for the first time in this sequence. Following [2], let every cycle $C \in \mathcal{C}$ capture any chip that uses any of its edges so that in the rest of the game the captured chip must travel along the edges of C. Since there are h chips and the game is infinite, every cycle must capture exactly one chip. Let $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$, where $i_1 < i_2 < \cdots < i_k$, be the nodes of some $C \in \mathcal{C}$. We claim that C in fact is the cycle $(x_{i_1} x_{i_2} \ldots x_{i_k})$. If not, there would be some smallest $j > 1$ such that $(\overrightarrow{x_{i_{j-1}}, x_{i_j}})$ is not an edge of C. But then x_{i_j} would not be firable the first time it should be fired during α because the chip captured by C would be someplace else, and a node x can be fired only if the chip of every cycle containing x currently is on x. Consequently, every $C \in \mathcal{C}$ is of the form $(x_{i_1} x_{i_2} \ldots x_{i_k}), i_1 < i_2 < \cdots < i_k$, so by definition G is orientable relative $\mathcal{C}$. □

PROPOSITION 3.5. *If $\mathcal{C}_1$ and $\mathcal{C}_2$ are two maximal families of edge-disjoint directed cycles in a connected eulerian graph G, then G is orientable relative $\mathcal{C}_1$ if and only if G is orientable relative $\mathcal{C}_2$.*

Proof. The proposition follows immediately from Lemma 3.4. □

DEFINITION 3.6. *By Proposition* 3.5, *we can speak of a connected eulerian graph G as* orientable *if it is orientable relative any, and hence every, maximal family of edge-disjoint directed cycles.*

THEOREM 3.7. *A connected eulerian graph G has an infinite* CFG *with $h(G)$ chips if and only if G is orientable.*

Proof. The theorem follows from Lemma 3.4 and Proposition 3.5. □

So, which connected eulerian graphs are orientable? One might conjecture that all of them are, but unfortunately this is false. The smallest counterexample has seven nodes and $h = 4$; see Fig. 1.

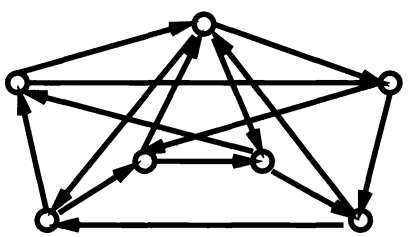


FIG. 1. *A nonorientable eulerian graph.*

4. Period length and game length of CFG. Let $G = (V, E)$ be a digraph. Following [2], a nonzero vector $\mathbf{v} \in \mathbb{N}^V$ is called a *period vector* for G if, in a CFG played on G, a firing sequence where every node i is fired v_i times leads back to the beginning position. A period vector is *primitive* if its entries have no common divisor greater than one.

PROPOSITION 4.1 (Björner and Lovász [2]). (i) *A* CFG *whose graph* G *is strongly connected has a unique primitive period vector* $\mathbf{v}$. *It is strictly positive, and all period vectors are of the form* $t \cdot \mathbf{v}, t = 1, 2, \ldots$.

(ii) *If* G *is connected eulerian, then* $\mathbf{v}$ *has all entries* $v_i = 1$.

(iii) *In general, the period vectors of a* CFG *are exactly the vectors of the form* $\lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \cdots + \lambda_k \mathbf{v}_k$, *where* $\lambda_i \in \mathbb{N}$ *and* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ *are the primitive vectors of the sink components of* G.

The *period length* of a CFG on G is denoted $\mathrm{per}(G)$ and is defined as the sum of all the entries of the primitive vectors of the sink components of G.

A vector $\mathbf{a} \in \mathbb{N}^V$ is *reduced* if for every period vector $\mathbf{v}$ of the CFG there exists a node i with $a_i < v_i$. Björner and Lovász proved the following.

PROPOSITION 4.2 (Björner and Lovász [2]). *Every score vector of a terminating firing sequence is reduced.*

The main theorem on period length and game length of CFG in [2] says that $\mathrm{game}(G) \leq 2nND\,\mathrm{per}(G)$, where $\mathrm{game}(G)$ is the maximal length of a terminal game on G and N is the total number of chips. We will prove another bound on $\mathrm{game}(G)$, in Theorem 4.6.

PROPOSITION 4.3. *Let* V_0 *be a node set containing at least one node of every sink component of a digraph* G. *In a* CFG *on* G *where no node in* V_0 *is ever fired, the total number of firings is no more than* $N(D^{n-2} + D^{n-3} + \cdots + D + 1)$.

Proof. Define a distance function l on nodes, such that $l(i)$ is the length of the shortest directed path from i to V_0. By the assumption on V_0, every node in G has a directed path to some node in V_0, and of course $l(i) < n$ for all i. We can thus partition the node set V in subsets $V_0, V_1, \ldots, V_{n-1}$ where $V_k \stackrel{\text{def}}{=} \{i \in V : l(i) = k\}$. Then there is no directed edge $(\overrightarrow{i,j})$ where $i \in V_k, j \in V_{k-x}$ for $x > 1$. Suppose that in the start position there are c_i chips on any node i. The total number of chips is $N = \sum c_i, i \in V$. Let $\mathbf{m} = [\alpha]$ be the score vector of a firing sequence α that does not fire any node in V_0. Then the amount of chips that $W_k \stackrel{\text{def}}{=} V_0 \cup V_1 \cup \cdots \cup V_k$ has received from the outside during α can be at most the original amount on $V \backslash W_k$ plus the amount that has been sent out from W_k. Since chips can only enter W_k through edges from V_{k+1} to V_k, we get

$$\sum_{j \in V_{k+1}} m_j \sum_{i \in V_k} d_{ji} \leq \sum_{j \in V \backslash W_k} c_j + \sum_{i \in W_k} m_i \sum_{j \in V \backslash W_k} d_{ij}. \tag{1}$$

Now, we have $\sum_{j \in V \backslash W_k} c_j \leq N$. Further, for $j \in V_{k+1}$ we have $\sum_{i \in V_k} d_{ji} \geq 1$, since every node in V_{k+1} has at least one edge to V_k. For the same reason we have $\sum_{j \in V \backslash W_k} d_{ij} \leq D - 1$ for any $i \in W_k$, except possibly if $i \in V_0$, but in that case $m_i = 0$. Hence (1) implies

$$\sum_{j \in V_{k+1}} m_j \leq N + \sum_{i \in W_k} m_i (D - 1). \tag{2}$$

Let $s_k \stackrel{\text{def}}{=} \sum_{i \in W_k} m_i$. Adding s_k to both sides of (2) yields

$$s_{k+1} \leq N + s_k D \tag{3}$$

and $s_0 = \sum_{i \in V_0} m_i = 0$. Solving this recurrence inequality yields $s_k \leq N(1 + D + \cdots + D^{k-1})$, and the number of firings in α is s_{n-1}. □

Remark 4.4. The bound given by Proposition 4.3 is essentially sharp, because the following game is suggested by the proof:

Construct a digraph G by first taking a directed path $i_{n-1}, i_{n-2}, \dots, i_0$ (so we have only one edge $(\overrightarrow{i_k, i_{k-1}})$ for each $k = 1, 2, \dots, n-1$) and then adding $D-1$ edges $(\overrightarrow{i_k, i_{n-1}})$ for every $k = 1, 2, \dots, n-2$. Put N chips on i_{n-1} and play the game without ever firing i_0. Then the game terminates when there are fewer than D chips on each node $i_k, k = 1, 2, \dots, n-2$ and no chips on i_{n-1}, so the bulk of the chips, at least $N' \stackrel{\text{def}}{=} N-(n-2)(D-1)$, is on i_0. Then i_1 must have fired N' times, so it must have received $N'D$ chips from i_2, so i_2 must have fired $N'D$ times. Continuing this type of argument, we get that every i_k must have fired $N'D^{k-1}$ times, so the total number of firings must have been at least $N'(1+D+\cdots+D^{n-2})$, so the error in the estimate of Proposition 4.3 is for this game bounded by $(n-2)D^{n-1}$, which is independent of N.

LEMMA 4.5. *If α is a firing sequence in a* CFG *on a directed graph G and $[\alpha]$ is reduced, then the number of firings in α is less than* $\operatorname{per}(G)+N(D^{n-2}+\cdots+D+1)$.

Proof. Suppose α is a firing sequence from position P_1 to P_2 such that its score vector, say $\mathbf{a}$, is reduced. Let V_0 contain every node i of every sink component H of G such that $a_i < v_i$, where v_i is the ith entry of the primitive period vector $\mathbf{v}_H$ of H. There must be at least one such node i in every sink component H, by the definition of reducedness and Proposition 4.1(iii), with $\lambda_H = 1$ and all other $\lambda_j = 0$. Hence V_0 intersects every sink component. Now illegally fire (i.e., we allow negative numbers of chips on the nodes) every $i \in V_0$ exactly $v_i - a_i$ times. We get some position P_3. P_3 has a nonnegative number of chips at each node because every node in V_0 must have no less chips than in P_1, $\mathbf{v}$ being a period vector, while $V \backslash V_0$ must have no less chips than in P_2 since these nodes have not been fired since P_2.

Examining the proof of Proposition 4.3 one easily verifies that it is applicable also on illegal firing sequences as long as they result in a nonnegative ending position. Since firing every node $i \notin V_0$ exactly $a_i - v_i$ times from P_1 while not firing nodes in V_0 also results in position P_3, this firing sequence must by Proposition 4.3 use at most $N(1+D+\cdots+D^{n-2})$ firings. Thus the original firing sequence between P_1 and P_3 uses less than $N(1+D+\cdots+D^{n-2})+\sum v_i = N(1+D+\cdots+D^{n-2})+\operatorname{per}(G)$ firings, and α was shorter than this sequence. □

We can now obtain a new bound on the game length in terms of the period length.

THEOREM 4.6. *For the* CFG *on any directed graph G,*

$$\operatorname{game}(G) < \operatorname{per}(G) + nD^{n-1}.$$

Proof. For any terminating game we know (see §3) that $N \le m-n$ where m is the number of edges. We must always have $m \le nD$. Hence $N \le n(D-1)$, and by Proposition 4.2 the score vector of a terminating game is reduced, so by Lemma 4.5 the game length is less than $\operatorname{per}(G)+N(1+D+\cdots+D^{n-2}) < \operatorname{per}(G)+nD^{n-1}$. □

To obtain a general bound on $\operatorname{game}(G)$ we need a general bound on $\operatorname{per}(G)$. Björner and Lovász proved that $\operatorname{per}(G) \le n(\sqrt{2}D)^{n-1}$ by applying Hadamard's inequality to the minors of the Laplacian of G. A better bound can be proved by pure combinatorics.

THEOREM 4.7. *For every digraph G,*

$$\operatorname{per}(G) < nD^{n-1}.$$

Proof. It is enough to prove the inequality for strongly connected G. Let u_i be the number of *i-rooted spanning trees* of G, i.e., spanning trees where every node has a directed path in the tree to i. We shall see that the vector $\mathbf{u}$ with entries u_i is an adequate period vector. It is clearly nonzero and integral, but in order to be a period vector it must satisfy

$$d^+(i)u_i = \sum_{j \in V} d_{ji} u_j$$

for every node i. The left side counts pairs consisting of one edge $(\overrightarrow{i,k}), k \in V$, and one i-rooted spanning tree. The right side counts pairs consisting of one edge $(\overrightarrow{j,i})$ and one j-rooted spanning tree, $j \in V$. It is easily verified that a bijection between these sets of pairs is obtained by, in the left pair, adding the edge $(\overrightarrow{i,k})$ to the i-rooted tree, thereby forming one directed cycle, and then removing the unique edge $(\overrightarrow{j,i})$ from this cycle, thus obtaining a j-rooted spanning tree. This bijection proves that $\mathbf{u}$ satisfies the period vector criterion.

Since $\mathbf{u}$ is an integral, nonzero period vector, but possibly not primitive, we have $\mathrm{per}(G) \leq \sum_{i \in V} u_i$. Any i-rooted spanning tree has exactly one outgoing edge from each node except from i, so

$$u_i \leq \prod_{j \neq i} d^+(j) \leq D^{n-1}.$$

Hence,

$$\mathrm{per}(G) \leq \sum_{i \in V} u_i \leq \sum_{i \in V} D^{n-1} = nD^{n-1}. \qquad \square$$

COROLLARY 4.8. *The* CFG *on a digraph G satisfies* $\mathrm{game}(G) < 2nD^{n-1}$.

Finally, let us discuss parallel firings. One move in a parallel chip-firing game consists of parallel firings of every node that is firable at the beginning of the move. Bitar and Goles [1] showed that the period length of a parallel CFG on an undirected tree is either 1 or 2 and noted that for general undirected graphs the period length could grow linearly with n, the number of nodes. For general directed graphs, since at most n firings can be done in one move, we of course have

$$\mathrm{game}(G)/n \leq \mathrm{parallelgame}(G) \leq \mathrm{game}(G)$$

and also

$$\mathrm{per}(G)/n \leq \mathrm{parallelper}(G),$$

but it is not clear whether $\mathrm{parallelper}(G) \leq \mathrm{per}(G)$ must hold. Since it is known (see [2], [5]) that the game length and period length may be exponential in n, then so is obviously the case also for parallel firing. Tardos [6] showed that the game length on an undirected graph can be proportional to n^4, so for parallel firing in that case the number of moves grows at least as n^3.

5. Period length and game length of PMG. In this section we will try to generalize as much as possible of the theory of CFG.

For a PMG on V, we analogously define $\mathbf{v} \in \mathbb{N}^V$ to be a period vector if a firing sequence in the PMG where every node i is fired v_i times leads back to the beginning

position. Of course the two notions coincide for a CFG viewed as a PMG. If $\mathbf{v}$ is a period vector for the PMG, then v_i must be a multiple of p_i, since the beginning and ending positions are regarded as equal only if the positions in every mutation sequence are equal. If, in addition, the chips distribution of the two positions shall be equal, the vector $\mathbf{v}'$ defined by $v_i' = v_i/p_i$ must be a period vector for the CFG on the great graph GG of the PMG. Call $\mathbf{v}$ PMG-primitive if $\mathbf{v}'$ is primitive on GG. Then Proposition 4.1 of Björner and Lovász can be reformulated for PMG.

PROPOSITION 5.1. (i) *A* PMG *whose great graph GG is strongly connected has a unique* PMG*-primitive period vector $\mathbf{v}$. It is strictly positive, and all period vectors are of the form $t \cdot \mathbf{v}, t = 1, 2, \ldots$.*

(ii) *If GG is connected eulerian, then $\mathbf{v}$ has entries $v_i = p_i$.*

(iii) *In general, the period vectors of a* PMG *are exactly the vectors of the form $\lambda_1\mathbf{v}_1 + \lambda_2\mathbf{v}_2 + \cdots + \lambda_k\mathbf{v}_k$, where $\lambda_i \in \mathbb{N}$ and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ are the* PMG*-primitive vectors of the sink components of GG.*

Proof. To have a period in the PMG it is necessary that every node i has been fired a multiple of p_i, times, since p_i is the period of the mutation sequence at node i. By the definition of great graph at the end of §2, p_i firings of node i in the PMG is equivalent to *one* firing of node i in the CFG on the great graph GG. Thus, $\mathbf{v}$ is a period vector of the PMG if and only if the vector $\mathbf{v}'$ defined by $v_i' = v_i/p_i$ is a period vector for the CFG on GG. The result then follows directly from Proposition 4.1. □

The *period length* of a PMG is denoted per(PMG) and is defined as the sum of all the entries of the PMG-primitive vectors of the sink components of GG. This is consistent with the CFG definition of per(G) as the sum of all the entries of the primitive vectors of the sink components of G.

Before we continue following the line of events in the previous section, let us make the promised generalization of the remark in §3.

PROPOSITION 5.2. *Given a* PMG *and any period vector $\mathbf{v}$, let $V' \subseteq V$ be the set of nodes that are firable in the initial position. The first node j to be fired more than v_j times, if any such node exists, must be in V'.*

Proof. Suppose $j \notin V'$ has been fired v_j times. The number of chips on j is a monotonic increasing function of the number of times each node $i \neq j$ has been fired. If every node i has been fired exactly v_i times, then the current position is the initial one, so j is not firable. Hence, before j can be fired for the (v_j+1)th time, some node $i \neq j$ must have been fired more than v_i times. □

The aim now is to relate the period length of a PMG to the maximal length of a terminating game playable on the PMG. We had the following proposition stated for CFG in [2]. Our new proof for PMG is somewhat shorter and simpler. Given a PMG, say that a vector $\mathbf{a} \in \mathbb{N}^V$ is *reduced* if for every period vector $\mathbf{v}$ of the PMG there exists a node i with $a_i < v_i$.

PROPOSITION 5.3. *Every score vector of a terminating firing sequence is reduced.*

Proof. Let α be a firing sequence with nonreduced score vector $\mathbf{a} = [\alpha]$. We want to show that α is not a terminating firing sequence, so we want to find some node that is firable after α. Since α is not reduced, some period vector $\mathbf{v}$ has all entries $v_i \leq a_i$. Thus we can split α as $\alpha_1 i \alpha_2$, such that $[i\alpha_2]_i = v_i$ and $[i\alpha_2]_j \geq v_j$ for all $j \neq i$. Let P be the position after α_1. Then i is firable in position P, and by the definition of period vector, the net flow of chips when playing $i\alpha_2$ is equal to the flow of chips when playing every node j exactly $[i\alpha_2]_j - v_j \geq 0$ times, while never playing i, so i has at least as many chips after playing $i\alpha_2$ from P as in P itself. Thus i is

firable after playing α from the original position. □

From Proposition 5.1(ii) we deduce that if a PMG with eulerian great graph has a terminating game, then some node i is fired less than p_i times.

Proposition 4.3 can be generalized to PMG. Let $p_{\max} \stackrel{\text{def}}{=} \max\{p_i : i \in V\}$, the length of the longest mutation period. Let D denote the maximum outdegree of the great graph GG.

PROPOSITION 5.4. *Let $V_0 \subset V$ be a node set containing at least one node of every sink component of the great graph GG of a* PMG. *Playing the* PMG *without firing any node in V_0 can go on for no more than $((N+n)(1+D+\cdots+D^{n-2})+nD^{n-1})p_{\max}$ firings.*

Proof. Partition GG in $V_0, V_1, \ldots, V_{n-1}$ as in the proof of Proposition 4.3. Let m_i be the number of complete mutation periods at node i in a game α that never fires the nodes of V_0, so m_i is the integer part of $[\alpha]_i/p_i$. Following the notation and argument of the other proof, we get

$$\sum_{j \in V_{k+1}} m_j \le N + \sum_{i \in W_k} (m_i + 1)(D-1). \tag{4}$$

Let $t_k \stackrel{\text{def}}{=} \sum_{i \in W_k}(m_i+1)$. Adding $t_k + n$ to both sides of (4) yields

$$t_{k+1} \le N + n + t_k D \tag{5}$$

and $t_0 = |V_0| \le n$, so the recurrence inequality yields $t_k \le (N+n)(1+D+\cdots+D^{k-1})+nD^k$. The number of firings in α is less than $\sum_{i \in V}(m_i+1)p_i \le t_{n-1}p_{\max} \le ((N+n)(1+D+\cdots+D^{n-2})+nD^{n-1})p_{\max}$. □

LEMMA 5.5. *If α is a firing sequence in a* PMG *with great graph GG and $[\alpha]$ is reduced, then the number of firings in α is less than* per(PMG) $+ ((N+n)(D^{n-2}+\cdots+D+1)+nD^{n-1})p_{\max}$.

Proof. We can follow the proof of Lemma 4.5 word for word (well, exchange GG for G and PMG-primitive for primitive), only we must use the bound given by Proposition 5.4 instead of its CFG counterpart. □

THEOREM 5.6. *For a* PMG,

$$\text{game(PMG)} < \text{per(PMG)} + 3nD^{n-1}p_{\max}.$$

Proof. We follow the line of proof as in Theorem 4.6. First, we know from §3 that in a terminating game we have $N + n \le m \le nD$. Second, we know that a terminating game has reduced score vector, so by Lemma 5.5 and the inequality above we have game(PMG) < per(PMG) $+ nD(1+D+\cdots+D^{n-2})p_{\max} + nD^{n-1}p_{\max} =$ per(PMG) $+ n(D+\cdots+D^{n-2})p_{\max} + 2nD^{n-1}p_{\max}$. If $D = 1$, then the game can only terminate if there are no chips at all, in which case game(PMG) $= 0$. Otherwise, we have

$$n(D+\cdots+D^{n-2})p_{\max} = n\left(\frac{D^{n-1}}{D-1} - 1\right)p_{\max} \le nD^{n-1}p_{\max},$$

and the theorem follows. □

From the discussion of period vectors of PMG and CFG in the very beginning of this section, it is evident that per(PMG) $\le p_{\max}\,\text{per}(GG)$. This yields a general bound on the game length of PMG.

COROLLARY 5.7. *For a* PMG, *the game length of terminating games is bounded by*

$$\text{game(PMG)} \leq 4nD^{n-1}p_{\max}.$$

Remark 5.8. Observe that if firings of nodes without any outgoing edges at all are not counted (since they do not do anything), then we have $p_{\max} \leq D$ and hence $\text{game(PMG)} \leq 4nD^n$, where D is the maximal outdegree of the great graph.

Remark 5.9. Björner and L. Lovász [2] also prove that the reachability problem is decidable for CFG, by describing an algorithm that takes a CFG and two positions, p and q, and which in finite time answers whether q can be reached by playing from p. It is not hard (though not entirely trivial) to generalize their algorithm (and the complexity analysis) to PMG, so the reachability problem is decidable also for PMG. We omit the details.

REFERENCES

[1] J. BITAR AND E. GOLES, *Parallel chip firing games on graphs*, Theoret. Comput. Sci., 92 (1992), pp. 291–300.

[2] A. BJÖRNER AND L. LOVÁSZ, *Chip firing games on directed graphs*, J. Algebraic Combin., 1 (1992), pp. 305–328.

[3] A. BJÖRNER, L. LOVÁSZ, AND P. SHOR, *Chip-firing games on graphs*, European J. Combin., 12 (1991), pp. 283–291.

[4] P. DIACONIS AND W. FULTON, *A growth model, a game, an algebra, Lagrange inversion, and characteristic classes*, Rend. Sem. Mat. Univ. Politec. Torino, 49 (1991), pp. 95–119.

[5] K. ERIKSSON, *No polynomial bound for the chip firing game on directed graphs*, Proc. Amer. Math. Soc., 112 (1989), pp. 1203–1205.

[6] G. TARDOS, *Polynomial bound for a chip firing game on graphs*, SIAM J. Discrete Math., 1 (1988), pp. 397–398.

[7] M. THORUP, *Firing games*, Tech. report. DIKU-94-15, Dept. of Computer Science, Univ. of Copenhagen, 1994.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 129–150, February 1996

012

LINEAR ALGORITHMS FOR PARTITIONING EMBEDDED GRAPHS OF BOUNDED GENUS*

L. ALEKSANDROV† AND H. DJIDJEV‡

Abstract. This paper develops new techniques for constructing separators for graphs embedded on surfaces of bounded genus. For any arbitrarily small positive ε we show that any n-vertex graph G of genus g can be divided in $O(n+g)$ time into components whose sizes do not exceed εn by removing a set C of $O(\sqrt{(g+1/\varepsilon)n})$ vertices. Our result improves the best previous ones with respect to the size of C and the time complexity of the algorithm. Moreover, we show that one can cut off from G a piece of no more than $(1-\varepsilon)n$ vertices by removing a set of $O(\sqrt{n\varepsilon(g\varepsilon+1)})$ vertices. Both results are optimal up to a constant factor.

Key words. graph separator, graph genus, algorithm, divide-and-conquer, topological graph theory

AMS subject classifications. 05C10, 05C85, 68R10

1. Introduction.

Let S be a class of graphs closed under the subgraph relation and $f(n)$ be a nonnegative function. An *$f(n)$-separator theorem* for S is defined in [18] as a theorem of the following form: *there exist constants $\alpha < 1$ and $\beta > 0$ such that if G is any n-vertex graph in S, then the vertices of G can be divided into three sets A, B, and C such that no edge joins a vertex in A with a vertex in B,* $\max(|A|, |B|) \leq \alpha n$ *and* $|C| \leq \beta f(n)$. The set of vertices C is called an *α-separator*. Separator theorems are known for many classes of graphs, i.e., for forests, grids [17], planar graphs [18], graphs of bounded genus [6], [11], graphs with an excluded minor [2], geometric graphs [20], and others. The $\sqrt{n}$-separator theorem for the class of planar graphs [18] is now a classic result in computational graph theory.

If an $o(n)$-separator theorem exists for S, then one can efficiently apply a divide-and-conquer method for solving graph problems on graphs from S. The sets of vertices A and B define subproblems which are independent and essentially smaller than the original problem. To find solutions to the subproblems defined by A and B one applies the same method until the sizes of the subproblems become so small that they can be easily solved directly. Separators have been successfully applied for solving different computational problems. Such applications include finding a suitable ordering of the equations of very large sparse systems of linear equations in order to minimize the fill-in during a Gaussian elimination [17], [12], finding approximate solutions to NP-complete problems [19], VLSI layout [3], [4], [16], designing efficient sequential and parallel algorithms [8], [15], [9], [13], and many others.

One inefficiency that results when the above method is straightforwardly applied is that the time required to find the decomposition of the original problem can be too large or even dominate the total time needed to solve the problem of interest. For instance, if an algorithm for finding a 2/3-separator of an n-vertex graph belonging to

* Received by the editors August 11, 1994; accepted for publication (in revised form) March 9, 1995.

† Bulgarian Academy of Science, Center for Informatics and Computer Technology, G. Bonchev 25-A, 1113 Sofia, Bulgaria.

‡ Bulgarian Academy of Science, Center for Informatics and Computer Technology, G. Bonchev 25-A, 1113 Sofia, Bulgaria and Department of Computer Science, Rice University, P.O. Box 1892, Houston, TX 77251 (hristo@cs.rice.edu). The research of this author was partially supported by National Science Foundation grant CCR-9409191.

a certain class of graphs requires $\Theta(n)$ time, then it will take $\Theta(n \log n)$ time to find a decomposition of the graph into subgraphs of $O(1)$ size by using that algorithm.

In our paper we give a solution to the problem of dividing an n-vertex graph into many pieces of small size in optimal linear time. More specifically, we prove that if G is an n-vertex graph with nonnegative vertex weights embedded on a surface of orientable genus g, then for any $\varepsilon \in (0,1)$ there exists a set C of no more than $4\sqrt{(g+1/\varepsilon)n}$ vertices of G whose removal leaves no component of total weight exceeding ε times the total weight of G. Such separators for $\varepsilon = o(n)$ are referred to in the literature as *ε-separators*. They have been used for the solutions of various problems, e.g., in constructing approximation algorithms for the maximum independent set and other NP-complete problems [19], pebbling [19], the embedding of data or communication structures [19], [4], shortest path problems [8], and the design of parallel algorithms [13]. We present an algorithm that finds such a separator in $O(n+g)$ time, given the embedding of G. A preliminary version of this result was published in [5]. Similar results for the class of planar graphs (i.e., for the case where g=0) were proved in [19], [8], where the complexity of the algorithms is $O(n \log n)$ and for the class of planar unweighted graphs in [13], where the size of the separator is not explicitly estimated. In [11] the existence of an ε-separator of size $O(\sqrt{ng/\varepsilon})$ for graphs of genus g is derived as a consequence of the iterative 2/3-separation. The complexity of the algorithm that follows from this approach is $O(n \log n)$. Thus, our algorithms improve the previous results with respect to the size of the separator and the time complexity of the algorithm. Moreover, we show that the size of the separators found by our algorithm is optimal within a constant factor.

Essential to our approach will be a data structure we call a *separation graph* and a technique we develop for manipulating with separation graphs. The separation graphs, which are sparse graphs of degree 3, contain all the relevant information we need in order to construct the separator. Separation graphs are more convenient to use for purposes of graph separation than the original graphs because of their simple structure and sparsity.

We also demonstrate the use of our new technique also by proving an optimal separator theorem for the class of graphs of bounded genus with a constant $\alpha = 1 - o(1)$. We prove that if G is an n-vertex graph with nonnegative vertex weights embedded on a surface of orientable genus g, then for any $\varepsilon \in (0, 1/15)$ there exists an $(1-\varepsilon)$-separator C_1 of G of $O(\sqrt{n\varepsilon(g\varepsilon+1)})$ vertices. The size of C_1 is optimal within a constant factor and C_1 can be found in $O(n+g)$ time given the embedding of G. A similar result was proved by Gilbert in [10] and by Djidjev and Gilbert in [7]; however, the constants in [10], [7] are much larger and the complexities of the algorithms are $O((n+g)\log n)$. Another application of separation graphs was described in [1].

We view the contribution of our work as follows: (i) we prove separator theorems for graphs embedded on orientable surfaces, where similar previous results concerned separators for planar graphs only; (ii) the complexity of our algorithms is linear; and (iii) our separators are smaller in size than the best previous separators for the planar case and are asymptotically optimal for arbitrary genus.

The paper is organized as follows. In §2 we give a definition of a separation graph and prove some of its properties. Next we give an algorithm for the construction of separation graphs and prove that any edge separator of the separation graph induces a set of cycles separating the original graph. In §3 we present the basic results of the paper. In §3.2 we construct a linear algorithm for the ε-separation of graphs embedded onto a surface of genus g in the case when ε is close to zero and in §3.3 we

construct a linear algorithm for the case when ε is close to one. In the final subsection we establish the optimality of our results.

2. Separation graphs. We begin with a brief description of some basic notions from topological graph theory. A more detailed and formal treatment can be found in [14].

A *graph* G is an ordered pair $(V(G), E(G))$ of sets, where $V(G)$ is a set of *vertices* and $E(G)$ is a set of *edges*. Each edge is an unordered pair of different vertices. We will consider graphs embedded on orientable surfaces. A *surface* is a connected, compact 2-manifold. An *embedding* $I(G)$ of the graph G onto a surface Z is a mapping of the vertices onto different points of Z and of the edges onto arcs along Z, so that the incidences are preserved, and no two arcs intersect at an inner point. We call the vertices and edges of G vertices and edges of $I(G)$ and we call *faces* of $I(G)$ the connected components of $Z \setminus I(G)$. We consider *simplicial* embeddings, i.e., each face is surrounded by at least three edges. An embedding is a *2-cell embedding* if each of its faces is homeomorphic to an open disk. If no ambiguity arises, we will refer to the 2-cell embeddings as *embeddings* and to $I(G)$ as G. The *orientable genus* (or the *genus*) of a graph G is the minimum genus of an orientable surface, called the *genus surface* of G, onto which G can be embedded. A *region* R of $I(G)$ we call any connected open subset of $I(G)$. The *boundary* of R is $\partial R = \bar{R} \setminus R$, where $\bar{R}$ is the closure of R.

The sets of vertices, edges, and faces of $I(G)$ (or G) will be denoted, respectively, by $V(G)$, $E(G)$, and $F(G)$ hereafter. An important relation among the numbers of vertices, edges, faces, and the genus of a 2-cell embedding $I(G)$ is given by Euler's formula

$$|V(G)| - |E(G)| + |F(G)| = 2c - 2g, \tag{1}$$

where c is the number of connected components of G and $|X|$ denotes the cardinality of a set X.

DEFINITION 2.1. *Let G be an embedded graph and T be a spanning tree of G. A* separation graph *of G with respect to the spanning tree T is a graph $S = S(G,T)$, where*

$$V(S) := F(G),$$
$$E(S) := \{(f_1, f_2) \ : \ f_1, f_2 \in F(G), \ f_1 \text{ and } f_2 \text{ share a nontree edge}\}.$$

Algorithm 2.1, described below, will assign proper weights to the edges of S which are essential for the use of a separation graph.

According to the above definition, the separation graph of a given embedded graph G is a subgraph of the dual graph of G and can be constructed in $O(|V(G)| + |E(G)|)$ time. For an example, see Fig. 1.

As the next lemma shows, if the genus g of G is small compared to $|V(G)|$, then S is much simpler (sparser) than G.

LEMMA 2.1. *Let G be a connected graph that has a 2-cell embedding on a surface of genus g with a spanning tree T. Then the separation graph $S(G,T)$ is connected and*

$$|E(S)| - |V(S)| = 2g - 1. \tag{2}$$

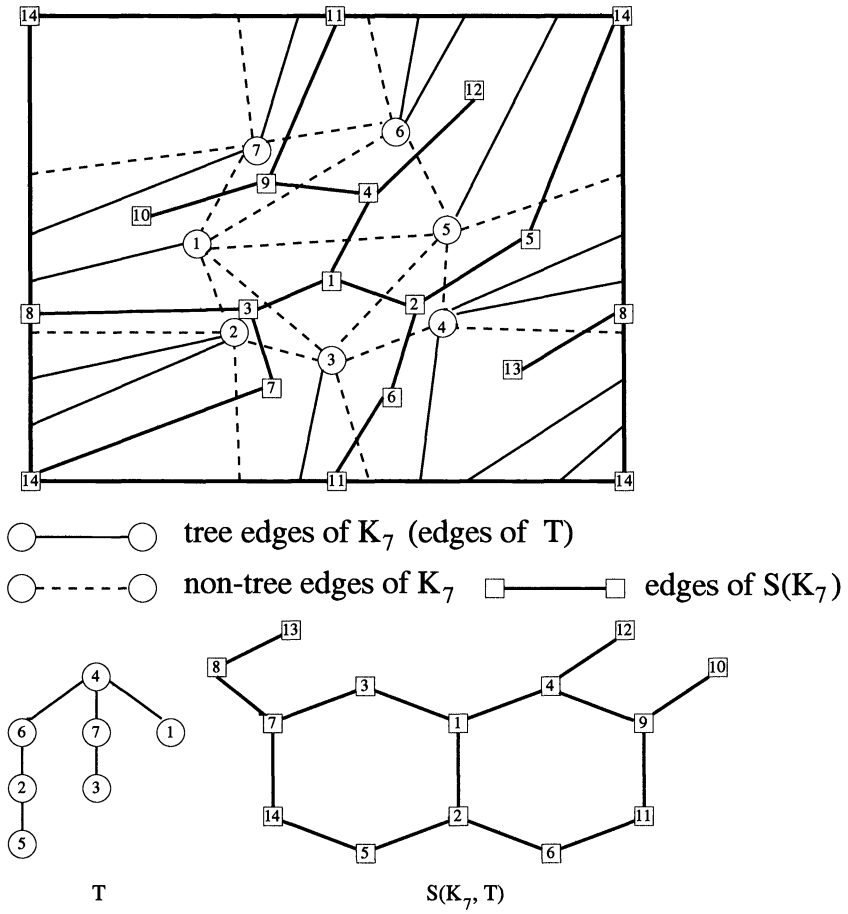


FIG. 1. *An embedding of K_7 on the torus and its separation graph.*

Proof. Assume that S is disconnected and let K be a component of S. Denote by R the region consisting of all faces that correspond to vertices of K together with their boundaries. The boundary ∂R of R is nonempty and it contains a simple closed curve that is an embedding of a cycle of G. On the other hand, by the definition of S, the boundary ∂R is a subset of the embedding $I(T)$ of T. This means that $I(T)$ contains a closed curve and therefore T contains a cycle, which is a contradiction. Thus S is connected.

By the definition of S, we have

$$|E(S)| = |E(G)| - |V(G)| + 1 \quad \text{and} \quad |V(S)| = |F(G)|.$$

Applying Euler's formula (1), we obtain

$$|E(S)| - |V(S)| = |E(G)| - |V(G)| + 1 - |F(G)| = 2g - 1,$$

which proves (2). □

For instance, if $g = 0$ (i.e., if G is planar), then by Lemma 2.1 S is a tree. In general, S can be represented as a tree plus $2g$ additional edges. An example for $g = 1$ is presented in Fig. 1.

We call a *T-cycle* of G any simple cycle that contains exactly one edge that is not in T. Any edge of $S(G,T)$ determines a unique T-cycle, the one containing the corresponding nontree edge of G. In what follows we will show that, if weights are assigned to the edges of S in a proper way, we can construct separators of G consisting of T-cycles by considering S instead of G.

Hereafter, we consider the case when the embedding of G is a triangulation, i.e., each of its faces is incident to three edges. For our purposes, this is not a restriction since any embedding could be extended to a triangulation by adding new edges. Subsequently, any vertex set that separates the resulting triangulation separates the original graph as well. Note that in the case when the embedding of G is a triangulation the vertices of S have maximal degree three, which can be used to simplify the algorithms on S.

We assume that the vertices of G have nonnegative weights $wt(\cdot)$. The algorithm below assigns weights on the edges of S that depend on the structure of G and on the weight function $wt(\cdot)$. We will use the following notations. For $v \in V(G)$ denote by $E(v,0)$ the set of all nontree edges of G incident to v and by $E(v,1)$ the set of all nontree edges of G whose both endpoints are adjacent to v. As the vertices at distance 1 from v define (at least one) simple cycle and there are no simple cycles in any spanning tree of G, then $E(v,1) \neq \emptyset$. The algorithm described below first defines for each nontree edge of G (or equivalently for each edge of S) a set $vert(e)$ of vertices of G such that the sets $vert(e)$ for $e \in E(G) \setminus E(T)$ form a partition of $V(G)$ (see Fig. 2). Then the algorithm assigns to each edge e of S a weight equal to the total weight of the set $vert(e)$.

ALGORITHM 2.1 (ASSIGNMENT OF WEIGHTS ON THE EDGES OF S).
Input: An embedded graph G with a spanning tree T and the corresponding separation graph $S(G,T)$; a weight function $wt(\cdot)$ defined on $V(G)$;
Output: A weight function $wt(\cdot)$ on the edges of S;

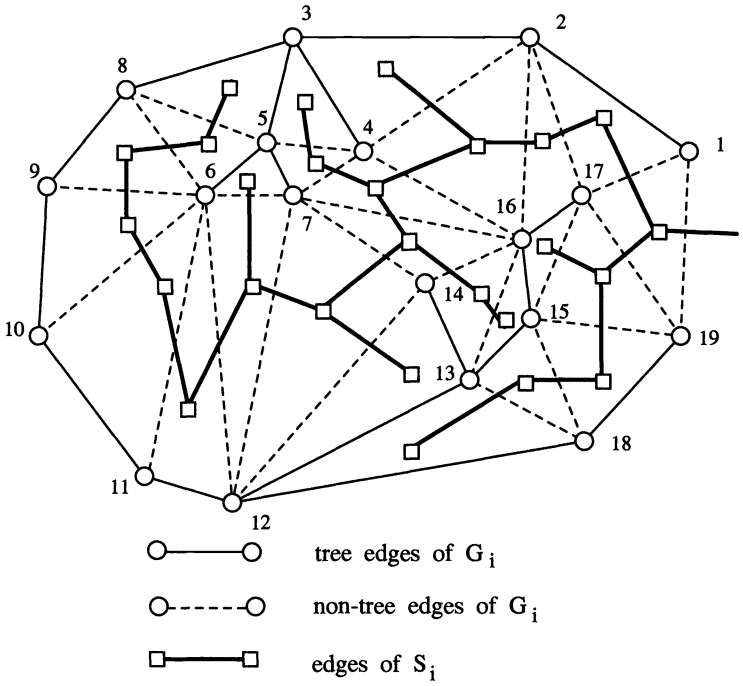
Set $vert(e) := \emptyset$ for $e \in E(G) \setminus E(T)$

For each $v \in V(G)$ **do**
  if $E(v,0) \neq \emptyset$
    then $e_1 :=$ any edge in $E(v,0)$
      $vert(e_1) := vert(e_1) \cup \{v\}$
    else {since always $E(v,1) \neq \emptyset$}
      $e_2 :=$ any edge in $E(v,1)$
      $vert(e_2) := vert(e_2) \cup \{v\}$
  endif
enddo
Set $wt(e') := \sum_{v \in vert(e)} wt(v)$, for $e' \in E(S)$,
where e is the corresponding to e' edge from $E(G) \setminus E(T)$.

Algorithm 2.1 can be implemented in linear time in a straightforward manner. Note that from the definition of $wt(e)$ it follows directly:

$$\sum_{e \in E(S)} wt(e) = \sum_{v \in V(G)} wt(v). \tag{3}$$

Remark. We will need weights on the edges of S as assigned by Algorithm 2.1 even in the case when the original graph is nonweighted. In this case we assume that all vertices of G have weight $1/|V(G)|$.



e	(1,17)	(2,4)	(2,17)	(4,5)	(5,8)	(6,8)	(6,9)	(6,10)	(6,11)	(7,12)	(13,18)	(14,16)	(15,18)	(17,19)	any other
vert(e)	{1}	{4}	{2,17}	{3}	{5}	{6,8}	{9}	{10}	{11}	{7,12}	{13}	{14,16}	{15,18}	{19}	{}

FIG. 2. *Illustrations to Algorithm* 2.1 *and Theorem* 2.1. *The subgraph* G_i *of* G *embedded inside a* T*-cycle of* G*, the corresponding region* Z_i*, and component* S_i.

The next theorem shows that to find a separator consisting of T-cycles one can use $S(G)$ instead of G (see Fig. 2). Observe that inequalities (4) and (5) below can be used to estimate the weights of the parts into which G is divided.

THEOREM 2.1. *Let* G *be a graph with weights on the vertices embedded on a surface* Z *such that each face is a triangle and let* T *be a spanning tree of* G*. Let* M *be a set of edges of* S *whose removal divides* S *into components* $S_1, \ldots, S_k$*. Then the set of curves* $\tilde{C}(M)$ *that are embeddings of the* T*-cycles corresponding to the edges of* M *divides* Z *into connected open regions* $Z_1, \ldots, Z_k$ *such that*

(i) *if* $wt(Z')$*, for* $Z' \subset Z$*, denotes the total weight of the vertices of* G *embedded in* Z' *then*

$$wt(Z_i) \leq wt(S_i), \quad i = 1, \ldots, k; \tag{4}$$

(ii) *if* T *is a breadth-first spanning tree of* G*, then*

$$wt(Z_i \cup \partial Z_i) \geq wt(S_i \cup \partial S_i) - wt(t), \quad i = 1, \ldots, k, \tag{5}$$

where ∂Z_i *is the boundary of* Z_i*,* ∂S_i *is the set of edges of* M *adjacent to* S_i*, and* t *is the root of* T.

Proof. Let the set of curves $\tilde{C}(M)$ divide Z into connected open regions $Z_1, \ldots, Z_{k'}$. For $i = 1, \ldots, k'$, denote by S_i' the subgraph of $(V(S), E(S) \setminus M)$ that is embed-

ded inside Z_i. Then the graphs S'_i, $i = 1, \ldots, k'$, define a partition of the graph $(V(S), E(S) \setminus M)$, i.e., we have

$$\bigcup_{i=1}^{k'} V(S'_i) = V(S) \quad \text{and} \quad V(S'_i) \cap V(S'_j) = \emptyset \quad \text{for} \quad i \neq j;$$

$$\bigcup_{i=1}^{k'} E(S'_i) = E(S) \setminus M \quad \text{and} \quad E(S'_i) \cap E(S'_j) = \emptyset \quad \text{for} \quad i \neq j.$$

We will show that each graph S'_i, $i = 1, \ldots, k'$, is connected. Assume that S'_i is disconnected for some $i \in \{1, \ldots, k'\}$ and let K be one of its components. Denote by R the region of Z consisting of all faces dual to the vertices of K. The boundary ∂R of R contains a simple closed curve that is an embedding of a cycle of G. According to the assumption it follows that $\partial R \subset I(M^* \cup T)$ and $\partial R \cap Z_i \neq \emptyset$, where M^* denotes the set of edges dual to the edges of M and $I(M^* \cup T)$ is the embedding of $M^* \cup T$. Consequently, there exists a simple closed curve $\gamma \subset \partial R$ such that $\gamma \cap Z_i \neq \emptyset$. In other words, we have found an embedding of a cycle $\gamma \subset I(M^* \cup T)$ with $\gamma \not\subset \tilde{C}(M)$. This is a contradiction, since the embedding of each cycle in $T \cup M^*$ belongs to $\tilde{C}(M)$.

We proved that S'_i, for $i = 1, \ldots, k'$, are connected components of the graph $(V(S), E(S) \setminus M)$. Therefore, $k = k'$ and $\{S_1, \ldots, S_k\} \equiv \{S'_1, \ldots, S'_k\}$. To simplify the notations we assume $S'_i = S_i$ for $i = 1, \ldots, k$.

Next, we prove that (i) and (ii) hold for Z_i for $i = 1, \ldots, k$. Let $v \in Z_i$ and e be the edge of S such that $v \in vert(e)$. Then, by Algorithm 2.1, the edge from $E(G) \setminus E(T)$ dual to e is either incident or has both endpoints adjacent to v. By the definition of Z_i, $e \in S_i \cup M$. On the other hand it is not possible that $e \in M$, because it implies $E(v, 0) \neq \emptyset$ and therefore $v \in \tilde{C}(M)$, which contradicts $v \in Z_i$. Thus $e \in S_i$ and (i) follows.

In order to prove (ii), we consider any edge $e \in S_i \cap \partial S_i$. We will show that $vert(e) \subset \{t\} \cup Z_i \cup \partial Z_i$. If $e \in S_i$, the relation follows immediately. Let $e \in \partial S_i$ and (v_1, v_2) be the edge of G dual to e. If $w \in vert(e)$ and $w \neq v_i, i = 1, 2$, then by Algorithm 2.1, $E(w, 0) = \emptyset$. (The assumption $E(w, 0) \neq \emptyset$ leads to $w \in vert(h)$ for some $h \in E(w, 0)$.) Therefore, any edge incident to w is from $E(T)$. The only vertex with this property in a breadth-first spanning tree of a triangulation is its root. Thus $vert(e) \subset \{v_1, v_2, t\}$ and inequality (ii) holds. □

We use the result of Theorem 2.1 as a basic tool for constructing separators in the next section. More precisely, if G, T, S, M, and $S_1, \ldots, S_k$ are as in Theorem 2.1, then the removal of the vertices of G that belong to the T-cycles corresponding to the edges in M divides G into components $G_1, \ldots, G_k$ such that $wt(G_i) \leq wt(S_i)$, $i = 1, \ldots, k$. Therefore, we can construct separators for G by constructing edge-separators for the separation graph S. This approach relies on the fact that, as we will show in the next section, we are able to construct edge-separators for sparse graphs in optimal linear time. Note that the length of a T-cycle can be $\Omega(|V(G)|)$, and thus a good bound on the size of the resulting separator of G does not directly follow from the existence of a small separator of S. So, in order to achieve both an optimal running time of our algorithm and an optimal size of the constructed separators, we will combine the use of separation graphs with an appropriate radius-reducing technique.

3. ε-separation of graphs of bounded genus. In this section we derive two results on the separation of graphs of bounded genus. The first one is related to the problem of dividing a graph into (possibly many) components of small weights, known as the *ε-separation* problem, and the second one concerns the problem of separating a single piece of small weight from a graph.

DEFINITION 3.1. *For any weighted graph G and $\varepsilon \in (0,1)$, the vertex (respectively, edge) set C is called an* ε-separator *(respectively,* ε-edge separator*) of G if the subgraph of G induced by $V(G) \setminus C$ (respectively, the subgraph induced by $E(G) \setminus C$) has no component of weight greater than $\varepsilon wt(G)$.*

First we will develop fast algorithms for the ε-separation of graphs of degree three.

3.1. Separating trees and sparse graphs. In the next lemma we present a linear algorithm that finds a small ε-edge separator for any graph of degree three.

LEMMA 3.1. *Let S be a graph of degree three with n vertices, m edges, and q connected components and nonnegative weights $wt(\cdot)$ on its edges. For any $\varepsilon \in (0,1)$ there exists a set M_ε of no more than $m - n + q + \lfloor 2/\varepsilon \rfloor$ edges, whose removal divides S into connected components $S_1, \ldots, S_k$ with the properties*

(i) *for any $i = 1, \ldots, k$*

$$wt(S_i) \leq \varepsilon wt(S); \tag{6}$$

(ii) *the inequality*

$$wt(S_i \cup \partial S_i) \geq \left(\frac{\varepsilon}{2}\right) wt(S) \tag{7}$$

holds for at least $k - q$ of the components S_i, $i = 1, \ldots, k$, where ∂S_i is the set of edges from M_ε adjacent to S_i. The set M_ε can be found in $O(m)$ time.

Proof. First we consider the case where S is a binary tree, i.e., $m = n - 1$ and $q = 1$. Let the set of edges of S be divided into levels $E_0, \ldots, E_r$ according to their distance to some fixed vertex of S. For $e \in E(S) \setminus E_0$ we denote by $pr(e)$ the unique edge at the lower level that shares a common vertex with e, and we create an additional edge to be a predecessor of the edges from E_0. Consider the following procedure.

Procedure: TREE SEPARATION

Input: A binary tree S, the set of levels $E_0, \ldots, E_r$ with respect to a root t, a parameter $\varepsilon \in (0,1)$, and the weights $wt(\cdot)$ on the edges of S.
Output: A set M_ε, which is an ε-edge separator of S.

For $i = r, r-1, \ldots, 0$, **do**
  For each $e \in E_i$ **do**
    If $wt(e) > (\varepsilon/2) wt(S)$ **then** insert e in M_ε
    else $wt(pr(e)) := wt(pr(e)) + wt(e)$.

This procedure runs in time linear on m, since every edge of S is considered only once. During the ith iteration of the loop, the current weight of any edge x in E_{r-i+1} is equal to its original weight plus the total weight of the component of S which would be cut if x and all edges already in M_ε are deleted. We include an edge e in M_ε if and only if its current weight $wt(e)$ is greater than $(\varepsilon/2) wt(S)$. Therefore, proposition (ii) follows for all components of $S \setminus M_\varepsilon$, except possibly for the one containing the root. Proposition (i) follows by the fact that the total weight of the component that will

be cut if e is deleted is equal to the sum of the current weights of its ancestors (no more than two), which were not included in M_ε during the previous iteration.

The estimation $|M_\varepsilon| \leq \lfloor 2/\varepsilon \rfloor$ follows by the observation that with each edge inserted in M_ε the total weight of the edges of S decreases by at least $(\varepsilon/2)wt(S)$.

In the general case when S is a graph of degree three we apply the following algorithm.

ALGORITHM 3.1 (EDGE SEPARATION OF GRAPHS OF DEGREE THREE).

Input: A graph of S degree three, an $\varepsilon \in (0,1)$, and weights $wt(\cdot)$ on edges of S.
Output: A set M_ε, which is an ε-edge separator of S.

Step 1: Find the set of the connected components of S.

Step 2: For each connected component Q with $wt(Q) > \varepsilon wt(S)$ do:

Step 2.1: Find a breadth-first spanning tree T of Q and divide the set of edges of T into levels with respect to the root of T.

Step 2.2: Insert the set of nontree edges of Q in M_ε.

Step 2.3: If $wt(T) > \varepsilon wt(S)$, then apply the tree separation procedure on T with parameter $\varepsilon_1 = \varepsilon wt(S)/wt(T)$ to find an ε_1-edge separator M_{ε_1} of T. Add M_{ε_1} to the set M_ε.

The proof that the set M_ε constructed by Algorithm 3.1 satisfies requirements of the lemma follows from the correctness of the tree separation procedure. □

3.2. Finding ε-separators for ε close to zero. Let G be an n-vertex graph with nonnegative vertex weights and $I(G)$ be a 2-cell embedding of G on a surface Z of orientable genus g. We are going to describe an algorithm that for a given $I(G)$ and $\varepsilon \in (0,1)$ finds an ε-separator C of G of no more than $4\sqrt{n(g+1/\varepsilon)}$ vertices in $O(n+g)$ time. Let us recall that, without loss of generality, we assume that $I(G)$ is a triangulation, since we can make each face a triangle by adding new edges. Any vertex set that separates the resulting graph will separate the original graph as well.

Let T be a breadth-first spanning tree rooted at a vertex t and let S be the corresponding separation graph as defined in Definition 2.1 and Algorithm 2.1. Denote the radius of T by R. For $r = 0, 1, \ldots, R$ we define level $L(r)$ to be the set of vertices lying at distance r from t.

We denote by $Z(r)$, $0 < r \leq R$, the closed region of Z consisting of all triangles that have at least one vertex lying on a level lower than r. The boundary of $Z(r)$ consists of simple closed curves that are embeddings of the vertices in $L(r)$ and certain nontree edges between them. They form a set of edge disjoint cycles in G, which we denote by $\tilde{L}(r) = \{c_1(r), \ldots, c_{\nu(r)}(r)\}$ (see Fig. 3). The set $\tilde{L}(r)$ can be constructed by traversing each connected component of the boundary of $Z(r)$ in one direction so that $Z(r)$ remains on the right and then choosing the first nontraversed edge in a counterclockwise order to be traversed next. This procedure requires time proportional to the size of the subgraph of G induced by the vertex set $L(r)$.

Let $Z^+(r)$ be the complement of $Z(r)$ to Z. We denote by $G(r)$, $T(r)$, and $S(r)$ the subgraphs of G, T, and S that are embedded on $Z(r)$. Correspondingly, the subgraphs embedded on $Z^+(r)$ are denoted by $G^+(r)$, $T^+(r)$, and $S^+(r)$. Clearly, the level $L(r)$ separates $G(r)$ and $G^+(r)$, and the set of edges of S that are dual to the edges of the cycles in $\tilde{L}(r)$ separates $S(r)$ and $S^+(r)$.

Our algorithm for finding an ε-separator constructs the required ε-separator iteratively. At each iteration step a fixed level $L(r)$ along which the graph will be

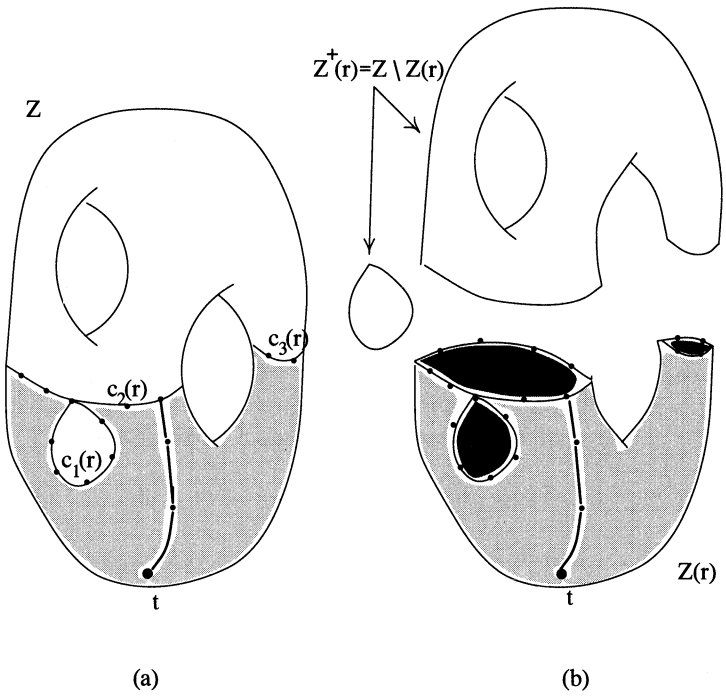


FIG. 3. *The region $Z(r)$ and its boundary.*

cut is considered. The algorithm constructs the set of embedded cycles $\tilde{L}(r)$ and the graph $S^+(r)$ defined above and by using the separation graph finds a set of vertices that divides the graph $G^+(r)$ into components of weights not exceeding $\varepsilon wt(G)$. The constructed vertex set is added to the current separator. Finally, the algorithm transforms the remaining (not still separated) subgraph $G(r)$ and the corresponding subgraph of the separation graph $S(r)$ in a form suitable for the next iteration step. Next we describe in more detail how this transformation is done and then we will give an outline of the whole algorithm.

The region $Z(r)$ can be considered as a surface with holes determined by the embeddings of the cycles in $\tilde{L}(r)$. Let $\tilde{Z}(r)$ be the surface obtained from $Z(r)$ by pasting open discs (faces) $f(c_1(r)), \ldots, f(c_\nu(r))$ on its holes. In our transformation we supplement the graphs $G(r)$ and $S(r)$, obtaining graphs $\tilde{G}(r)$ and $\tilde{S}(r)$ that possess the following properties: (a) the graph $\tilde{G}(r)$ triangulates the surface $\tilde{Z}(r)$; (b) the tree $T(r)$ is a breadth-first spanning tree of $\tilde{G}(r)$; and (c) the graph $\tilde{S}(r)$ is a weighted separation graph for $\tilde{G}(r)$ with respect to the tree $T(r)$. The following procedure describes how such graphs $\tilde{G}(r)$ and $\tilde{S}(r)$ can be constructed.

Procedure: SUBGRAPH SUPPLEMENTATION

Input: Subgraphs $G(r)$, $S(r)$ embedded on $Z(r)$; a set of embedded cycles $\tilde{L}(r)$ and the weights of the edges in S dual to the edges in $\tilde{L}(r)$.
Output: Graphs $\tilde{G}(r)$ and $\tilde{S}(r)$ embedded on $\tilde{Z}(r)$ satisfying properties (a), (b), and (c).

Step 1: For each cycle $c \in \tilde{L}(r)$ add a face $f(c)$ to the embedding of $G(r)$ with a boundary the embedding of c. Triangulate each face $f(c)$ by adding new

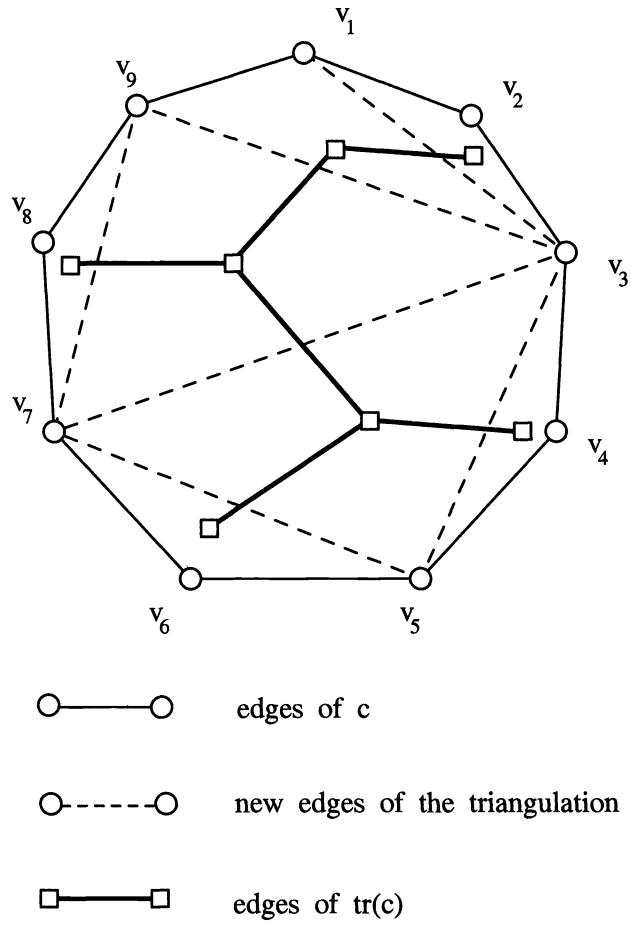


FIG. 4. *A triangulated face $f(c)$ and the corresponding $tr(c)$.*

edges to $G(r)$ embedded onto $f(c)$. Define $\tilde{G}(r)$ to be the resulting graph.

Step 2: For each face $f(c)$ $c \in \tilde{L}(r)$ construct the tree $tr(c)$ dual to the triangulation of $f(c)$ obtained in Step 1. Assign zero weights to the edges of $tr(c)$ (see Fig. 4). Add the trees $tr(c)$ for $c \in \tilde{L}(r)$ to the graph $S(r)$. Add to $S(r)$ the edges dual to the edges of the cycles in $\tilde{L}(r)$ preserving their original weights in S. Define $\tilde{S}(r)$ to be the resulting graph.

The proof that graphs $\tilde{G}(r)$, $T(r)$, and $\tilde{S}(r)$ have properties (a), (b), and (c) follows directly from their definitions. The running time of the procedure is proportional to the size of the set $\tilde{L}(r)$.

Next, we present our main algorithm that constructs an ε-separator for a given graph G embedded onto a surface of genus g. We assume that $g < n/16$; otherwise $4\sqrt{(g+1/\varepsilon)n} > n$ and the set $V(G)$ is a trivial ε-separator. Additionally, we assume without loss of generality that $I(G)$ is a triangulation, for otherwise we can define additional edges to make each face a triangle.

ALGORITHM 3.2 (CONSTRUCTION OF AN ε-SEPARATOR).

Input: A 2-cell embedding of a graph G on a surface of orientable genus g; a non-

negative weight function defined on vertices of G; a number $\varepsilon \in (0,1)$.

Output: An ε-separator C of G.

Step 1: Choose any vertex t of G and construct a breadth-first spanning tree T of G rooted at t. Let R be the radius of T. For $j = 0, 1, \ldots, R$, find the levels $L(j)$ consisting of all vertices at distance j from t.

Step 2: Construct the separation graph $S = S(G,T)$ of G with respect to T. Assign weights to the edges of S using Algorithm 2.1.

Step 3: Set $h = \lceil \frac{1}{2}\sqrt{n/(g+1/\varepsilon)} \rceil$. Find an integer k between 0 and $h-1$ that minimizes the number of vertices in the set $L = \bigcup_{j=0}^{l} L(k+jh)$, where $l = \lfloor \frac{R-k}{h} \rfloor$. (Thus $|L| \leq n/h$.) Set initially $C = L$.

Step 4: For $j = l+1, l, \ldots, 0$, do Steps 4.1 through 4.7 below.

Step 4.1: Set $r = k + (j-1)h$ if $j \geq 1$, or $r = 0$ if $j = 0$.

Step 4.2: Find the set of cycles $\tilde{L}(r)$.

Step 4.3: Determine the subgraph $S^+(r)$, i.e., the subgraph of S embedded onto $Z^+(r)$.

Step 4.4: If $wt(S^+(r)) > \varepsilon wt(S)$, then find an ε_j-edge separator M_j for $S^+(r)$ by applying Algorithm 3.1 on $S^+(r)$ with parameter $\varepsilon_j = \varepsilon wt(S)/wt(S^+(r))$. Otherwise set $M_j = \emptyset$.

Step 4.5: Add to the current separator C the set C_j of vertices of G on levels above $L(r)$ that belong to T-cycles corresponding to edges in M_j.

Step 4.6: Find subgraphs $G(r)$, $S(r)$, and $T(r)$.

Step 4.7: Apply the subgraph supplementation procedure on $G(r)$ and $S(r)$ and obtain graphs $\tilde{G}(r)$ and $\tilde{S}(r)$. Set $G = \tilde{G}(r)$, $S = \tilde{S}(r)$, and $T = T(r)$.

In the next theorem we prove the correctness of Algorithm 3.2 and estimate the size of the constructed ε-separator.

THEOREM 3.1. *Let G be an n-vertex graph with nonnegative vertex weights and $I(G)$ be a 2-cell embedding of G on an orientable surface of genus g. For any $\varepsilon \in (0,1)$ there exists an ε-separator C of G of no more than $4\sqrt{(g+1/\varepsilon)n}$ vertices. The separator C can be found in $O(n+g)$ time, given $I(G)$.*

Proof. Let C be the set constructed by Algorithm 3.2 for the graph G. The proof will consist of three parts.

A. *Proof that C is an ε-separator of G.* It will be appropriate to define the following set of curves that contain all vertices of C:

$$\tilde{C} = \left\{ \bigcup_{j=0}^{l} \tilde{L}(k+jh) \right\} \cup \left\{ \bigcup_{j=0}^{l+1} \tilde{C}_j \right\}, \tag{8}$$

where $\tilde{L}(k+jh)$ is the embedding of the cycles in $L(k+jh)$; $\tilde{C}_j$ denotes the set of curves which are embeddings of the parts of the T-cycles corresponding to the edges in M_j with endpoints on levels between $k+(j-1)h$ and $k+jh$.

To simplify our presentation we will introduce more convenient notations. For $j = 1, \ldots, l+1$ we denote $X(k+(j-1)h)$ by X_j, where X can be any of the symbols Z, Z^+, $\tilde{Z}$, S, S^+, and $\tilde{S}$. Additionally, let us denote $Z_0 = S_0 = \emptyset$, $Z_0^+ = \tilde{Z}_1$, $\tilde{Z}_{l+2} = Z$, and $\tilde{S}_{l+2} = S$.

LEMMA 3.2. *The set of curves $\tilde{C}$ divides Z into regions of weights not exceeding $\varepsilon wt(G)$, where the weight of a region is defined as the total weight of the vertices of G embedded onto it.*

Proof of Lemma 3.2. We will show that the set of curves $\tilde{C}_{l+1}$ divides $Z_{l+1}^+ = Z \setminus Z_{l+1}$ into regions of weights not exceeding $\varepsilon wt(G)$. Consider the set M of edges of S containing the edges dual to the edges of the cycles of $\tilde{L}(k+lh)$ plus the edges included in M_{l+1}. According to Theorem 2.1, there is a one-to-one correspondence between the components of $S \setminus M$ and the regions of $Z \setminus \{\tilde{L}(k+lh) \cup \tilde{C}_{l+1}\}$ associated with $I(G)$. In addition, the weight of each region does not exceed the weight of the corresponding component of $S \setminus M$ (see Theorem 2.1(i)). The set of curves $\tilde{L}(k+lh)$ separates Z_{l+1} from the regions on Z_{l+1}^+. The set of edges M_{l+1} separates S_{l+1}^+ into components of weight not exceeding $\varepsilon wt(G)$ (note that $wt(S) = wt(G)$), and therefore the set of curves $\tilde{C}_{l+1}$ divides Z_{l+1}^+ into regions of weights not exceeding $\varepsilon wt(G)$. Applying the same argument we prove that for $j = l, \ldots, 0$ the set of curves $\tilde{C}(M_j)$ divides Z_j^+ into regions of weights not exceeding $\varepsilon wt(G)$. According to our definitions, the set $\tilde{C}(M_j)$ is a superset of $\tilde{C}_j$, since Z_j^+ is equal to $Z_{j+1} \setminus Z_j$ plus the set of open discs (faces) pasted onto the holes determined by the curves in (embedded cycles) $\tilde{L}(k+jh)$. But $\tilde{C}_j$ is a restriction of $\tilde{C}(M_j)$ on $Z_{j+1} \setminus Z_j$, and thus $\tilde{C}_j$ divides $Z_{j+1} \setminus Z_j$ into regions of weights not exceeding $\varepsilon wt(G)$. The lemma follows by this fact and the observation that the set of curves $\cup_{j=0}^{l} \tilde{L}(k+jh)$ divides Z into the regions $Z_{j+1} \setminus Z_j$, $j = l+1, \ldots, 0$. □

The vertex set C constructed by Algorithm 3.2 induces a subgraph of G whose embedding contains the set of curves $\tilde{C}$. By Lemma 3.2 and the definitions it follows that C is an ε-separator for G.

B. *Time complexity.* Steps 1,2, and 3 run in $O(|I(G)|)$ time. Each of the substeps included in the loop of Step 4 runs in linear time on its input. During Step 4 the algorithm traverses the graph and therefore this step and the algorithm run in $O(|I(G)|) = O(n+g)$ time.

C. *Estimation of the size of* C. The set C was constructed as the union

$$C = L \cup \left\{ \bigcup_{j=0}^{l+1} C_j \right\}, \tag{9}$$

where $L = \bigcup_{j=0}^{l} L(k+jh)$ is the set of levels determined in Step 3 and sets C_j were constructed in Step 4.5. By the choice of k in Step 3, it follows that $|L| \leq n/h$ and thus

$$|C| \leq \frac{n}{h} + \sum_{j=0}^{l+1} |C_j'|,$$

where C_j' denotes the set C_j without the vertices in L. The sets C_j' consist of parts of T-cycles that lie on levels between $k+jh$ and $k+(j-1)h$ for $j > 0$, and on levels less than k for $j = 0$. Therefore, each of these parts could contain no more than $2(h-1)$ vertices and hence from the previous inequality we obtain

$$|C| \leq \frac{n}{h} + 2(h-1) \sum_{j=0}^{l+1} |M_j|. \tag{10}$$

Let us now estimate the sum $\sum_{j=0}^{l+1} |M_j|$. For $j = 0, \ldots, l+1$, we denote by $q(S_j^+)$ the number of the connected components of the graph S_j^+. According to the construction

of the sets M_j in Step 4.4, we have

$$|M_j| \leq |E(S_j^+)| - |V(S_j^+)| + q(S_j^+) + \left\lfloor \frac{2wt(S_j^+)}{\varepsilon wt(G)} \right\rfloor,$$

which follows by Lemma 3.1 if $wt(S_j^+) > \varepsilon wt(G)$, or by the fact that the right side is nonnegative otherwise (in this case $|M_j| = 0$). Summing these inequalities over $j = 0, \ldots, l+1$, we obtain

$$\begin{aligned} \sum_{j=0}^{l+1} |M_j| &\leq \sum_{j=0}^{l+1} \left(|E(S_j^+)| - |V(S_j^+)| + q(S_j^+) + \left\lfloor \frac{2wt(S_j^+)}{\varepsilon wt(G)} \right\rfloor \right) \\ &\leq \frac{2}{\varepsilon} + \sum_{j=0}^{l+1} q(S_j^+) + \sum_{j=0}^{l+1} (|E(S_j^+)| - |V(S_j^+)|), \end{aligned} \tag{11}$$

since $\sum_{j=0}^{l+1} wt(S_j^+) \leq wt(G)$. By the construction of graphs $\tilde{S}_j$ in the subgraph supplementation procedure it follows, for $j = 0, 1, \ldots, l+1$, that

$$E(\tilde{S}_{j+1}) = \left\{ E(S_j^+) \cup E(\tilde{S}_j) \right\} \setminus \left\{ \bigcup_{c \in \tilde{L}(k+(j-1)h)} E(tr(c)) \right\},$$

$$V(\tilde{S}_{j+1}) = \left\{ V(S_j^+) \cup V(\tilde{S}_j) \right\} \setminus \left\{ \bigcup_{c \in \tilde{L}(k+(j-1)h)} V(tr(c)) \right\},$$

and

$$\begin{aligned} &|E(\tilde{S}_{j+1})| - |V(\tilde{S}_{j+1})| \\ &= |E(S_j^+)| - |V(S_j^+)| + |E(\tilde{S}_j)| - |V(\tilde{S}_j)| \\ &\qquad + \sum_{c \in \tilde{L}(k+(j-1)h)} (|V(tr(c))| - |E(tr(c))|) \\ &= |E(S_j^+)| - |V(S_j^+)| + |E(\tilde{S}_j)| - |V(\tilde{S}_j)| + |\tilde{L}(k+(j-1)h)|, \end{aligned} \tag{12}$$

where $|\tilde{L}(k+(j-1)h)|$ denotes the number of the cycles in $\tilde{L}(k+(j-1)h)$. Summing (12) over j and since $\tilde{S}_0 = \emptyset$ and $\tilde{S}_{l+2} = S$ we obtain that

$$\sum_{j=0}^{l+1} \left(|E(S_j^+)| - |V(S_j^+)| \right) = |E(S)| - |V(S)| - \sum_{j=0}^{l+1} |\tilde{L}(k+(j-1)h)|. \tag{13}$$

We substitute (13) in (11) and apply (2):

$$\begin{aligned} \sum_{j=0}^{l+1} |M_j| &\leq \frac{2}{\varepsilon} + |E(S)| - |V(S)| + \sum_{j=0}^{l+1} (q(S_j^+) - |\tilde{L}(k+(j-1)h)|) \\ &= \frac{2}{\varepsilon} + 2g - 1 + \sum_{j=0}^{l+1} (q(S_j^+) - |\tilde{L}(k+(j-1)h)|) \end{aligned} \tag{14}$$

(see Lemma 2.1). For each $j = 0, \ldots, l+1$, the number $|\tilde{L}(k+(j-1)h)|$ is the number of cycles on the boundary of Z_j. Using Theorem 2.1 it is easy to see that the number of components of the graph S_j^+ is equal to the number of connected regions of $\tilde{Z}_{j+1} \setminus Z_j$. Each of the connected regions of $\tilde{Z}_{j+1} \setminus Z_j$ has at least one of the cycles from $\tilde{L}(k+(j-1)h)$ on its boundary, and no two of these regions have a common cycle on their boundaries. Hence, the number of cycles in $\tilde{L}(k+(j-1)h)$ is greater than or equal to the number of connected regions of $\tilde{Z}_{j+1} \setminus Z_j$, which gives the inequality

$$q(S_j^+) \leq |\tilde{L}(k+(j-1)h)| \quad \text{for} \quad j = 0, 1, \ldots, l+1. \tag{15}$$

By (14) and (15) it follows that

$$\sum_{j=0}^{l+1} |M_j| \leq 2g + \frac{2}{\varepsilon}. \tag{16}$$

We obtain the desired estimation substituting (16) and the value of h in (10), i.e.,

$$|C| \leq \frac{n}{h} + 2(h-1)\left(2g + \frac{2}{\varepsilon}\right) \leq 4\sqrt{\left(g + \frac{1}{\varepsilon}\right) n}. \qquad \square$$

As we will show in the last subsection, the size of C is optimal within a constant factor for any $\varepsilon \leq 2/3$. On the other hand, when ε is close to one, it is evident that the size of C cannot be optimal. We investigate such type of separators in the next subsection.

Intuitively, when ε is close to one, we need to cut off a small piece of the given graph in order to divide it into parts of sizes not greater than ε, and therefore the separator (the "boundary" of this small piece) should be small too. An algorithm that finds good separators in this case could be useful for designing incremental type algorithms on graphs.

3.3. Finding ε-separators for ε close to one. It will be more convenient to talk about $(1-\varepsilon)$-separators for small values of ε instead of talking about ε-separators for big values of ε. Below we show how Algorithm 3.2 can be extended to cope with $(1-\varepsilon)$-separators when ε is small. More precisely, we present an algorithm based on Algorithm 3.2 that, for $\varepsilon \in (0, 1/15)$, finds a $(1-\varepsilon)$-separator C_1 of no more than $2 + 30\sqrt{2}\sqrt{n\varepsilon(2g\varepsilon+1)}$ vertices.

The idea of the algorithm for constructing C_1 is the following. We first run Algorithm 3.2 on the input graph G with parameter 2ε, storing information for the obtained division. Let the constructed separator be $C_{2\varepsilon}$. As discussed above, the set $C_{2\varepsilon}$ is defined as the set of vertices belonging to the set of curves $\tilde{C}_{2\varepsilon}$; see (8). The set of curves $\tilde{C}_{2\varepsilon}$ divides the surface Z into connected open regions, which we denote by $Z_1, \ldots, Z_{p''}$. Let $G_1, \ldots, G_{p''}$ be the subgraphs of G embedded inside these regions and $\partial G_1, \ldots, \partial G_{p''}$ be the subgraphs that form region boundaries. Clearly, the vertex set $V(\partial G_i)$ separates the graph G_i from G. From the previous subsection we know that the weight of any Z_i does not exceed $2\varepsilon wt(G)$. For our purposes, however, we will need some *lower* bound on the weight of the parts of G. For this end we "group" the sets Z_i as follows. Using the division $Z_1, \ldots, Z_{p''}$ we derive a collection $V_1, \ldots, V_p (p \leq p'')$ of nonintersecting sets of vertices of G. Each of these sets contains the vertices of an appropriately chosen set of graphs G_i so that its weight plus the weight of its boundary exceeds $\varepsilon wt(G)$. The boundary of a given vertex set is defined

as the set of vertices that are not in the set but are adjacent to a vertex in the set. So, the boundary ∂V_i of a set V_i, for each $i = 1, \ldots, p$, is the union of the vertices of the boundaries of those of the graphs $G_1, \ldots, G_{p''}$, whose vertex sets are in V_i. We find the required $(1-\varepsilon)$-separator C_1 as the smallest among the sets $\partial V_1, \ldots, \partial V_p$.

Next we give more details about how to find the sets V_i for $i = 1, \ldots, p$. Initially, we define a collection of p' sets $U_i(r)$ $(p \leq p' \leq p'')$, where $i = 1, \ldots, p'(r)$ and $r \in R_{k,h} = \{k+lh, k+(l-1)h, \ldots, k, 0\}$, during the implementation of Algorithm 3.2 as follows. Consider any specific iteration of Step 4, i.e., let $r \in R_{k,h}$ be fixed. In Step 4.4 we found an edge separator M for the graph $S^+(r)$. Let the connected components of $S^+(r) \setminus M$ be $S_1^+(r), \ldots, S_{p'(r)}^+(r)$. According to Theorem 2.1, these components correspond to regions $Z_1^+(r), \ldots, Z_{p'(r)}^+(r)$ that satisfy inequalities (4) and (5). The relation between the regions $Z_1^+(r), \ldots, Z_{p'(r)}^+(r)$ and the regions $Z_1, \ldots, Z_{p''}$ is that each connected region of the interior of the intersection $Z(r) \cap Z_i^+(r)$ is equal to one of the regions $Z_1, \ldots, Z_{p''}$. (Note that the region $Z_i^+(r)$ could contain parts of faces added by the subgraph supplementation procedure that pastes open disks onto the holes of $Z(r)$.) Then, for $i = 1, \ldots, p'(r)$, we define $U_i(r)$ as the set of vertices of G embedded inside $Z \cap Z_i^+(r)$. The boundary $\partial U_i(r)$ consists of the vertices of G lying on the boundary of the region $Z \cap Z_i^+(r)$. Since the separation graphs are associated with the embedding of G, i.e., the vertices of a component $S_i^+(r)$ correspond to the triangles inside the region $Z_i^+(r)$ for $i = 1, \ldots, p'(r)$, then we easily construct sets $U_i(r)$ and their boundaries, for $i = 1, \ldots, p'(r)$, from the components of $S^+(r) \setminus M$ during the corresponding iteration of Step 4 in Algorithm 3.2. According to Lemma 3.1 and Theorem 2.1,

$$wt(U_i(r)) \leq 2\varepsilon wt(G), \quad i = 1, \ldots, p'(r). \tag{17}$$

Additionally, since the number of the connected components of $S^+(r)$ is $q(r)$, then for at least $p'(r) - q(r)$ (w.l.o.g. for the first $p'(r) - q(r)$) of the sets $U_i(r)$ we have

$$\varepsilon wt(G) \leq wt(U_i(r) \cup \partial U_i(r)), \quad i = 1, \ldots, p'(r) - q(r). \tag{18}$$

By reordering the sequence of the sets $U_i(r)$ so that the sets satisfying inequality (18) precede the others, we obtain a sequence $U_1, \ldots, U_{p'}$ with $p' = \sum_{r \in R_{k,h}} p'(r)$. Let the sets satisfying (18) be $U_1, \ldots, U_{p_1}$ for some $p_1 \leq p'$. For $i = 1, \ldots, p_1$ we define $V_i = U_i$.

In the second stage we define sets $V_{p_1+1}, \ldots, V_p$ as unions of groups of the "light" sets $U_{p_1+1}, \ldots, U_{p'}$, i.e., we find numbers $i_0 = 0 < i_1 < \cdots < i_{p-p_1} \leq p' - p_1$, so that

$$\varepsilon wt(G) \leq \sum_{i=p_1+i_{j-1}+1}^{p_1+i_j} wt(U_i \cup \partial U_i) \leq 2\varepsilon wt(G), \quad j = 1, \ldots, p - p_1$$

and

$$\sum_{i=p_1+i_{p-p_1}+1}^{p'} wt(U_i \cup \partial U_i) \leq 2\varepsilon wt(G).$$

Such numbers i_j, for $j = 1, \ldots, p - p_1$, exist since the sets U_i, for $i = p_1, \ldots, p'$, do not satisfy (18). So the sets $V_{p_1+1}, \ldots, V_p, V_{p+1}$ are defined by

$$V_{p_1+j} = \bigcup_{i=p_1+i_{j-1}+1}^{p_1+i_j} U_i \quad \text{for} \quad j = 1, \ldots, p - p_1$$

and

$$V_{p+1} = \bigcup_{i=p_1+i_{p-p_1}+1}^{p'} U_i.$$

We define the boundary ∂V_i of a set V_i to be the union of the boundaries of the sets that are included in V_i.

As described above, we obtain vertex sets $V_1, \ldots, V_{p+1}$ with boundaries $\partial V_1, \ldots, \partial V_{p+1}$ satisfying

$$wt(V_i) \leq 2\varepsilon wt(G) \quad \text{for } i = 1, \ldots, p+1 \tag{19}$$

and

$$wt(V_i \cup \partial V_i) \geq \varepsilon wt(G) \quad \text{for } i = 1, \ldots, p. \tag{20}$$

Each of the boundaries $\partial V_1, \ldots, \partial V_p$ is a $(1-\varepsilon)$-separator and we just chose the minimal among these separators. Formally, our algorithm is presented below.

ALGORITHM 3.3 (CONSTRUCTION OF A $(1-\varepsilon)$-SEPARATOR).

Input: A 2-cell embedding of a graph G on a surface of orientable genus g; weight function defined on vertices of G; a number $\varepsilon \in (0, 1/15)$.

Output: A set C_1 of vertices that is a $(1-\varepsilon)$-separator of G.

Step 1: Run Algorithm 3.2 on G with parameter 2ε. Find a (2ε)-separator $C_{2\varepsilon}$, vertex sets $U_1, \ldots, U_{p'}$, and their boundaries $\partial U_1, \ldots, \partial U_{p'}$, as defined above.

Step 2: If $wt(C_{2\varepsilon}) \geq (1/15)wt(G)$, then define C_1 to contain the heaviest $\lceil 15\varepsilon|C_{2\varepsilon}| \rceil$ vertices of $C_{2\varepsilon}$ and output C_1.

Step 3: As described before the algorithm construct sets $V_1, \ldots, V_{p+1}$ and boundaries $\partial V_1, \ldots, \partial V_{p+1}$, so that inequalities (19) and (20) hold.

Step 4: Among the sets of vertices $\partial V_1, \ldots, \partial V_p$ find the set containing the minimum number of vertices. Output that set to be the required separator C_1.

The correctness and complexity of this algorithm will be formally proved in the next theorem.

THEOREM 3.2. *Let G be an n-vertex graph with nonnegative vertex weights and $I(G)$ be a 2-cell embedding of G on an orientable surface Z of genus g. For any $\varepsilon \in (0, 1/15)$ there exists a $(1-\varepsilon)$-separator C_1 of G of no more than $2 + 30\sqrt{2}\sqrt{n\varepsilon(2g\varepsilon+1)}$ vertices. The separator C_1 can be found in $O(n+g)$ time given $I(G)$.*

Proof: Let C_1 be the separator constructed by Algorithm 3.3. We divide our proof into three parts.

A. *Proof that C_1 is a $(1-\varepsilon)$-separator.* Consider the case where the 2ε-separator $C_{2\varepsilon}$ found in Step 1 has weight greater than $(1/15)wt(G)$. In this case, according to the construction of C_1 in Step 2, we have

$$wt(C_1) \geq \frac{\lceil 15\varepsilon|C_{2\varepsilon}| \rceil}{|C_{2\varepsilon}|} wt(C_{2\varepsilon}) \geq \varepsilon wt(G).$$

This means that C_1 is a $(1-\varepsilon)$-separator of G and the theorem follows, since by Theorem 3.1

$$\lceil 15\varepsilon|C_{2\varepsilon}| \rceil \leq \left\lceil 30\sqrt{2}\sqrt{n\varepsilon(2\varepsilon g+1)} \right\rceil.$$

Hereafter, we assume that

$$wt(C_{2\varepsilon}) < \left(\frac{1}{15}\right) wt(G). \tag{21}$$

In this case the set C_1 is found in Step 4 as the smallest of the boundaries $\partial V_1, \ldots, \partial V_p$. Therefore C_1 is a $(1-\varepsilon)$-separator for G, since inequality (19) holds for each of the sets $V_1, \ldots, V_p$.

B. *Time complexity.* The first step of Algorithm 3.3 consists of running Algorithm 3.2 and constructing the sets $U_1, \ldots, U_{p'}$ and their boundaries. Algorithm 3.2 runs in $O(n+g)$ as shown in Theorem 3.1. The additional time for the construction of the sets $U_1, \ldots, U_{p'}$ and $\partial U_1, \ldots, \partial U_{p'}$ during the implementation of Algorithm 3.2 is clearly linear too. Step 2 can be implemented in $O(n)$ time by using a linear algorithm for ordered statistics. Steps 3 and 4 obviously run in $O(n)$ time. Therefore, the complexity of Algorithm 3.3 constructs a $(1-\varepsilon)$-separator for G in $O(n+g)$ time.

C. *Estimation of the size of* C_1. For the size of C_1, we have by Step 4

$$|C_1| \leq \frac{1}{p} \sum_{i=1}^{p} |\partial V_i|. \tag{22}$$

First we estimate the sum $\sum_{i=1}^{p} |\partial V_i|$ and then we give an estimation of p.

Recall that the union $\bigcup_{i=1}^{p+1} \partial V_i$ coincides with the separator $C_{2\varepsilon}$. On the other hand, Algorithm 3.2 constructs $C_{2\varepsilon}$ as the set of vertices lying onto a set of curves $\tilde{C}_{2\varepsilon}$, see (8). The set of curves $\tilde{C}_{2\varepsilon}$ could be considered as an embedding of a graph with a vertex set $V(\tilde{C}_{2\varepsilon}) = \bigcup_{i=1}^{p''} V(\partial Z_i)$, an edge set $E(\tilde{C}_{2\varepsilon})$ consisting of the edges embedded on $\tilde{C}_{2\varepsilon}$, and faces $Z_1, \ldots, Z_{p''}$. Note that this embedding may not be a 2-cell embedding. In this case Euler's formula (1) gives the inequality

$$|E(\tilde{C}_{2\varepsilon})| - |V(\tilde{C}_{2\varepsilon})| \leq 2g + p'' - 2. \tag{23}$$

Each edge of $E(\tilde{C}_{2\varepsilon})$ belongs to the boundary of two of the regions $Z_1, \ldots, Z_{p''}$. The boundaries $\partial Z_1, \ldots, \partial Z_{p''}$ consist of simple closed curves (cycles). Then

$$\sum_{i=1}^{p''} |V(\partial Z_i)| = 2E(\tilde{C}_{2\varepsilon}).$$

By the definition of the sets V_i and (23) we obtain

$$\sum_{i=1}^{p+1} |\partial V_i| \leq \sum_{i=1}^{p''} |V(\partial Z_i)| = 2E(\tilde{C}_{2\varepsilon}) \leq 2\left(|C_{2\varepsilon}| + 2g + p'' - 2\right). \tag{24}$$

Next, we find a lower bound for p. By (19),

$$(p+1)2\varepsilon wt(G) \geq \sum_{i=1}^{p+1} wt(V_i) = wt(G \setminus C_{2\varepsilon}).$$

By (21) and since $\varepsilon < 1/15$

$$p \geq \frac{wt(G) - wt(C_{2\varepsilon})}{2\varepsilon wt(G)} - 1 > \frac{2}{5\varepsilon}. \tag{25}$$

Combining (22), (24), and (25) we obtain

$$|C_1| \le 2 + 5\varepsilon\,(|C_{2\varepsilon}| + 2g + p'' - p - 2)\,. \tag{26}$$

To complete the estimation of $|C_1|$ we use the inequality

$$p'' - p \le 6\sqrt{n\left(g + \frac{1}{2\varepsilon}\right)} + 6(g-1), \tag{27}$$

which will be proved below. We substitute (27) in (26), and by using $n > 16g$ and Theorem 3.1 we obtain

$$\begin{aligned} |C_1| &\le 2 + 5\varepsilon\left(|C_{2\varepsilon}| + 6\sqrt{n\left(g + \frac{1}{2\varepsilon}\right)} + 8(g-1)\right) \\ &\le 2 + 60\varepsilon\sqrt{n\left(g + \frac{1}{2\varepsilon}\right)} = 2 + 30\sqrt{2}\sqrt{n\varepsilon(2g\varepsilon + 1)}. \end{aligned}$$

Let us now prove (27). Recall that p'' is the number of the connected open regions $Z_1, \ldots, Z_{p''}$ of $Z \setminus \tilde{C}_{2\varepsilon}$ and p is the number of the vertex sets $V_1, \ldots, V_p$. We will use the parameter p', which is the number of the vertex sets $U_1, \ldots, U_{p'}$. Since by definition $U_1, \ldots, U_{p'}$ form a subpartition of $V_1, \ldots, V_p$ and for $i = 1, \ldots, p'$ the set U_i comprises the vertices embedded into one or more of the regions Z_j for $j = 1, \ldots, p''$, then $p \le p' \le p''$. According to our constructions, vertices embedded in two regions Z_{i_1} and Z_{i_2} are in one $U_i(r)$ for some $i = 1, \ldots, p'(r)$ and $r \in R_{k,h}$, if and only if the regions Z_{i_1} and Z_{i_2} are components of $Z \cap Z^+(r)$. Therefore, both Z_{i_1} and Z_{i_2} neighbor a hole of $Z(r)$; see Fig. 3. Therefore, they have at least one edge of the graph $\tilde{L}(r)$ on its boundary, where $\tilde{L}(r)$ was defined as the graph of cycles $\{c_1(r), \ldots, c_{\nu(r)}(r)\}$ forming the boundary of $Z(r)$. Moreover, no two regions corresponding to a fixed r can be adjacent to the same edge of $\tilde{L}(r)$. Hence,

$$p'' - p' \le \sum_{r \in R_{k,h}} (|E(\tilde{L}(r))| - \nu(r)). \tag{28}$$

Further, the difference $p' - p$ does not exceed the number $p' - p_1$ of sets U_i, for $i = 1, \ldots, p'$, for which (18) is not valid. As pointed out, for any $r \in R_{k,h}$, at most $q(r)$ of the sets $U_i(r)$ do not satisfy (18), where $q(r)$ denotes the number of connected components of $S^+(r)$. Thus,

$$p' - p_1 \le \sum_{r \in R_{k,h}} q(r)$$

and by (15), where the numbers $q(r)$ are compared to the numbers $\nu(r)$ denoting the number of cycles in $\tilde{L}(r)$, we obtain

$$p' - p \le p' - p_1 \le \sum_{r \in R_{k,h}} q(r) \le \sum_{r \in R_{k,h}} \nu(r). \tag{29}$$

Adding (28) to (29) gives

$$p'' - p \le \sum_{r \in R_{k,h}} |E(\tilde{L}(r))|. \tag{30}$$

So, to estimate $p'' - p$ we need an estimation for the number of edges of the graph $\tilde{L} = \bigcup_{r \in R_{k,h}} \tilde{L}(r)$. The graph $\tilde{L}$ is simplicial (has no parallel edges or loops) and has an embedding (not necessarily a 2-cell) onto a surface of genus g. Therefore, using Euler's formula (1) as an inequality and the inequality $2|E(\tilde{L})| \geq 3|F(\tilde{L})|$ between the number of edges and the number of faces in any embedding of a simplicial graph, we obtain

$$|E(\tilde{L})| \leq 3|V(\tilde{L})| + 6(g-1). \tag{31}$$

The set $V(\tilde{L})$ is the union of the vertices lying on the levels $L(r)$ for $r \in R_{k,h}$. According to the choice of h and k (see Step 3 of Algorithm 3.2), we have

$$|V(\tilde{L})| = \sum_{r \in R_{k,h}} |L(r)| \leq 2\sqrt{\left(g + \frac{1}{2\varepsilon}\right) n},$$

that, together with (31) and (30), implies (27). □

3.4. Tightness of results. In this subsection we show that the results of Theorems 3.1 and 3.2 are optimal up to a constant factor. More precisely, we are going to prove the following theorem.

THEOREM 3.3. *There exists a constant $d > 0$ such that for any $\varepsilon \in (0, 2/3)$, g, and $n > g + 1/\varepsilon$*

(i) *there exists an n-vertex graph G of genus g that has no ε-separator of size less than $d\sqrt{n(g+1/\varepsilon)}$; and*

(ii) *there exists an n-vertex graph G_1 of genus g that has no $(1-\varepsilon)$-separator of size less than $d\sqrt{n\varepsilon(g\varepsilon+1)}$.*

Proof. We will base our proof on two special cases of our theorem that are proved in previous works. The case $g = 0$ follows immediately from the result of Theorem 6 in [18]. The case $\varepsilon = 2/3$ is proved in [6], [11].

We consider graphs with equal weights on the vertices. For a graph G and $\varepsilon \in (0,1)$ we denote by $c(G,\varepsilon)$ the minimum number of vertices in an ε-separator of G. Let $K(n,g)$ be the class of graphs with no more than n vertices and genus not exceeding g. We define the following function:

$$\mu(n,g,\varepsilon) = \max_{G \in K(n,g)} c(G,\varepsilon).$$

From the definition it is clear that the function μ is increasing on n and g and decreasing on ε. The validity of the theorem in the cases $g = 0$ and $\varepsilon = 2/3$ means that

$$\mu(n,0,\varepsilon) = \Omega\left(\sqrt{\frac{n}{\varepsilon}}\right), \tag{32}$$

$$\mu(n,0,1-\varepsilon) = \Omega\left(\sqrt{n\varepsilon}\right), \tag{33}$$

$$\mu(n,g,2/3) = \Omega\left(\sqrt{ng}\right). \tag{34}$$

By using the monotonicity of μ, (32), and (34) we obtain

$$\begin{aligned} 2\mu(n,g,\varepsilon) &\geq \mu(n,0,\varepsilon) + \mu(n,g,2/3) \\ &= \Omega\left(\sqrt{\frac{n}{\varepsilon}}\right) + \Omega\left(\sqrt{ng}\right) = \Omega\left(\sqrt{n\left(g + \frac{1}{\varepsilon}\right)}\right), \end{aligned}$$

which proves (i).

We shall prove (ii) by considering two cases.

1) $\varepsilon g \leq 1$. In this case by the monotonicity of μ and (33) we have

$$\mu(n, g, 1-\varepsilon) \geq \mu(n, 0, 1-\varepsilon) = \Omega\left(\sqrt{n\varepsilon}\right) = \Omega\left(\sqrt{n\varepsilon(g\varepsilon+1)}\right),$$

which gives (ii).

2) $\varepsilon g > 1$. To show that (ii) holds in this case, we shall prove first that

$$\varepsilon\mu\left(n, g, \frac{2}{3}\right) \leq \mu(n, g, 1-\varepsilon). \tag{35}$$

Let G be any graph from $K(n, g)$. We will construct a 2/3-separator of G as a union of $(1-\varepsilon)$-separators. We set $G_0 = G$ and define a sequence of graphs $G_1, \ldots, G_k$ in the following way. Let C_{i-1}, for $i = 1, \ldots, k$, be the minimal $(1-\varepsilon)$-separator of the graph G_{i-1} and G_i be the greatest component of $G_{i-1} \setminus C_{i-1}$. The weights $wt(G_i)$ decrease with i and satisfy the inequalities $wt(G_i) \leq (1-\varepsilon)^i wt(G)$, for $i = 1, \ldots, k$. The integer k is chosen to be the smallest such that $wt(G_k) \leq (2/3)wt(G)$ and thereby $k \leq 1/\varepsilon$. Since C_{i-1}, for $i = 1, \ldots, k$, is an $(1-\varepsilon)$-separator of G_{i-1}, then the only component of $G_{i-1} \setminus C_{i-1}$, for $i = 1, \ldots, k$ that is heavier than $(2/3)wt(G)$ is G_i. Thus the union $C = \cup_{i=1}^{k} C_{i-1}$ is a 2/3-separator of G. Therefore, we have

$$c\left(G, \frac{2}{3}\right) \leq |C| = \sum_{i=1}^{k} |C_{i-1}| = \sum_{i=1}^{k} c(G_i, 1-\varepsilon) \leq k\mu(n, g, 1-\varepsilon),$$

which together with the inequality $k \leq 1/\varepsilon$ gives (35). Finally, we prove (ii) by combining (35), (34), and $\varepsilon g > 1$:

$$\mu(n, g, 1-\varepsilon) \geq \varepsilon\ \mu\left(n, g, \frac{2}{3}\right) = \varepsilon\Omega\left(\sqrt{ng}\right) = \Omega\left(\sqrt{n\varepsilon(g\varepsilon+1)}\right). \qquad \square$$

REFERENCES

[1] L. ALEKSANDROV AND H. N. DJIDJEV, *Improved upper and lower bounds on separation of toroidal graphs*, in Proc. Optimal Algorithms '89, Lecture Notes in Computer Science 401, Springer–Verlag, Berlin, New York, 1989, pp. 126–138.

[2] N. ALON, P. SEYMOUR, AND R. THOMAS, *A separator theorem for graphs with an excluded minor and its applications*, in Proceedings of the 22nd Ann. ACM Simp. on Theory of Computing, Baltimore, MD, 1990, pp. 293–299.

[3] S. N. BHATT AND F. T. LEIGHTON, *A framework for solving VLSI graph layout problems*, J. Comput. System Sci., 28 (1984), pp. 300–343.

[4] H. N. DJIDJEV, K. DIKS, O. SYKORA, AND I. VRTO, *Edge separators for planar graphs and their applications*, J. Algorithms, 14 (1993), pp. 258–279.

[5] H. N. DJIDJEV, *Linear algorithms for graph separation problems*, in Proceedings of SWAT '88, Lecture Notes in Computer Science 318, Springer–Verlag, Berlin, New York, 1988, pp. 216–221.

[6] ———, *A separator theorem*, Compt. Rend. Acad. Bulg. Sci., 34 (1981), pp. 643–645.

[7] H. N. DJIDJEV AND J. R. GILBERT, *Separators in graphs with negative or multiple vertex weights*, SIAM Conf. on Discrete Mathematics, Atlanta, GA, 1990.

[8] G. N. FREDERICKSON, *Fast algorithms for shortest paths in planar graphs with applications*, SIAM J. Comput., 16 (1987), pp. 1004–1021.

[9] G. N. FREDERICKSON AND R. JANARDAN, *Efficient message rooting in planar networks*, SIAM J. Comput., 18 (1989), pp. 843–857.

[10] J. R. GILBERT, *Graph Separator Theorem and Sparse Gaussian Elimination*, Ph.D. thesis, Stanford University, Stanford, CA, 1980.

[11] J. R. GILBERT, J. P. HUTCHINSON, AND R. E. TARJAN, *A separator theorem for graphs of bounded genus*, J. Algorithms, 5 (1984), pp. 391–407.
[12] J. R. GILBERT AND R. E. TARJAN, *The analysis of a nested disection algorithm*, Numer. Math., 50 (1987), pp. 377–404.
[13] M. T. GOODRICH, *Planar separators and parallel polygon triangulation*, in Proceedings of the 24th Symp. on Theory of Computing, Victoria, BC, 1992, pp. 507–516.
[14] J. L. GROSS AND T. W. TUCKER, *Topologycal Graph Theory*, John Wiley and Sons, New York, 1987.
[15] X. HE AND Y. YESHA, *A nearly optimal parallel algorithm for constructing depth-first spanning trees in planar graphs*, SIAM J. Comput., 17 (1988), pp. 486–491.
[16] C. E. LEISERSON, *Area efficient VLSI computation*, in Foundations of Computing, MIT Press, Cambridge, MA, 1983.
[17] R. J. LIPTON, D. J. ROSE, AND R. E. TARJAN, *Generalized nested dissection*, SIAM J. Numer. Anal., 16 (1979), pp. 346–358.
[18] R. J. LIPTON AND R. E. TARJAN, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.
[19] ———, *Applications of a planar separator theorem*, SIAM J. Comput., 9 (1980), pp. 615–627.
[20] G. L. MILLER, S.-H. TENG, AND S. A. VAVASIS, *A unified geometric approach to graph separators*, in Proceedings of the 23rd Ann. ACM Simp. on Theory of Computing, New Orleans, LA, 1991, pp. 538–547.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 151–154, February 1996

013

NOWHERE-ZERO 4-FLOWS AND CAYLEY GRAPHS ON SOLVABLE GROUPS*

BRIAN ALSPACH†, YI-PING LIU‡, AND CUN-QUAN ZHANG§

Abstract. We prove that every Cayley graph on a finite solvable group admits a nowhere-zero 4-flow. In particular, every cubic Cayley graph on a solvable group is 3-edge-colorable.

Key words. integer flow, edge-coloring, Cayley graph

AMS subject classifications. 05C25, 05C15

1. Introduction.

Throughout this paper graphs have neither loops nor multiple edges. We use the term *multigraph* when multiple edges are allowed. If X is a graph, $V(X)$ and $E(X)$ denote the vertex set and edge set, respectively, of X.

DEFINITION 1.1. Let X be a graph and $D(X)$ be an orientation of X. A *k-flow* on X is an integer-valued function $f : E(X) \to (-k, k)$ such that for every vertex $u \in V(X)$ the sum of the flow values on the outgoing arcs from u in $D(X)$ equals the sum of the flow values on the incoming arcs at u in $D(X)$. If $f(e) \neq 0$ for every edge $e \in E(X)$, the flow is called a *nowhere-zero k-flow*.

There are several well-known unsolved problems related to flow problems. Probably the best known unsolved problem dealing with flows is the following problem of Tutte [13].

CONJECTURE 1.2. Every 2-connected graph containing no subdivision of the Petersen graph admits a nowhere-zero 4-flow. F. Jaeger [6] proved the first of the following two results. The second of the two is a consequence of the four-color theorem.

THEOREM 1.3. *Every 4-edge-connected graph admits a nowhere-zero 4-flow.*

THEOREM 1.4. *Every 2-edge-connected planar graph admits a nowhere-zero 4-flow.*

DEFINITION 1.5. Let G be a finite group and $S \subset G$ satisfy $1 \notin S$ and $s \in S$ if and only if $s^{-1} \in S$. The *Cayley graph* $X(G; S)$ is the graph with vertex set G and $ab \in E(G)$ if and only if $b = as$ for some $s \in S$.

The first of the following two conjectures was originally posed by L. Lovász [9] as a research problem and has come to be known as Lovász's conjecture. The consideration of Lovász's conjecture quickly led a number of people to consider the second of the two. It has been attributed to various people in the literature, but it is not at all clear who initially posed it.

CONJECTURE 1.6. Every connected vertex-transitive graph has a Hamilton path.

CONJECTURE 1.7. Every connected Cayley graph with three or more vertices contains a Hamilton cycle.

* Received by the editors November 2, 1993; accepted for publication (in revised form) March 9, 1995.

† Department of Mathematics and Statistics, Simon Fraser University, Burnaby, BC, V5A 1S6 Canada. The research of this author was partially supported by Natural Sciences and Engineering Research Council of Canada grant A-4792.

‡ Department of Mathematics, Nanjing Normal University, Nanjing, People's Republic of China.

§ Department of Mathematics, West Virginia University, Morgantown, WV 26506-6310 (cqzhang@wrnvm.wvnet.edu). The research of this author was partially supported by National Science Foundation grant DMS-9104824.

These two conjectures have attracted considerable attention over the last 24 years, and there have been many partial results. Some of the partial results are the results of very nice work; nevertheless, in many ways very little is really known about resolving the two conjectures. Since a graph with a Hamilton cycle admits a nowhere-zero 4-flow, in order for Conjecture 1.7 to be true it must be the case that appropriate Cayley graphs admit nowhere-zero 4-flows. This led Alspach and Zhang to make the following weaker conjecture at the Louisville workshop on Hamilton cycles in 1992.

CONJECTURE 1.8. Every Cayley graph with degree at least two admits a nowhere-zero 4-flow.

Another motivation for the preceding conjecture is that every graph admitting a nowhere-zero 4-flow admits a cycle double cover (see [7], [14], and [5]). It has also been shown that every connected Cayley graph has a cycle double cover [3].

2. Main results. The following lemma is crucial for the proofs of the main results. It is not hard to prove, and a proof can be found in [6], [11], [13] (see [8]).

LEMMA 2.1. *Let X be a cubic graph. The following two statements are equivalent.*

1. *The graph X admits a nowhere-zero 4-flow.*
2. *The graph X is 3-edge-colorable.*

The remainder of the paper addresses the following two results.

THEOREM 2.2. *Every cubic Cayley graph on a solvable group is 3-edge-colorable.*

R. Stong [12] proved that every Cayley graph $X(G;S)$ on a nilpotent group of even order has a 1-factorization as long as S is a minimal generating set for G. In particular, Stong's result implies that every cubic Cayley graph on a nilpotent group is 3-edge-colorable. The preceding theorem is an extension to solvable groups of this special case of Stong's theorem.

COROLLARY 2.3. *Every Cayley graph of degree at least two on a solvable group admits a nowhere-zero 4-flow.*

Proof. Let X be a Cayley graph of degree at least 2 on a solvable group. If X is of degree 2, then its components are cycles and it admits a nowhere-zero 2-flow. It is known that the edge connectivity of a connected Cayley graph is equal to its degree [10]. Thus, if X is of degree 4 or more, each component of X is 4-edge-connected and by Theorem 1.3 all of X admits a nowhere-zero 4-flow. This leaves only the case that X is cubic. In this case X is 3-edge-colorable by the preceding theorem. Lemma 2.1 then implies that X admits a nowhere-zero 4-flow and we are done. □

Proof of Theorem 2.2. Let $X = X(G;S)$ be a cubic Cayley graph on the solvable group G. The theorem is proved by induction on the order $|G|$ of G. We may assume that X is connected, for if it is not, we may apply the induction assumption to each component. R. Stong [12] has proved that every connected Cayley graph on a finite abelian group of even order has a 1-factorization (a partition of the edge set into 1-factors, that is, the chromatic index equals the degree). Thus, the result follows if G is abelian, and consequently, we assume that G is not abelian.

Let us examine S. We know that an element of order 2 in G generates a 1-factor of X. Since $|S| = 3$, we know that S contains either one element or three elements of order 2. In the latter case, X is 3-edge-colorable. Hence, we assume that $S = \{a, a^{-1}, b\}$, where $|b| = 2$ and $|a| = r > 2$.

If G has a nontrivial normal subgroup N such that $S \cap N = \emptyset$, then consider the quotient graph $\overline{X}$ obtained by first contracting every coset of N to a single vertex. If some vertex of a coset Ng is adjacent to d vertices of another coset Nh, then every vertex of Ng is adjacent to d vertices of Nh and vice versa because N is a normal

subgroup. Then we put an edge of multiplicity d between the vertices corresponding to the cosets Ng and Nh in $\overline{X}$.

There are three possibilities for $\overline{X}$. First, $\overline{X}$ may be $3K_2$ (that is, two vertices joined by an edge of multiplicity 3). In this case, X is a bipartite graph. It is 3-edge-colorable because regular bipartite graphs have a 1-factorization.

Second, every vertex of $\overline{X}$ may be incident with an edge of multiplicity 1 and another edge of multiplicity 2. This means the quotient graph looks like an even length cycle in which every other edge around the cycle has multiplicity 2. Since each edge of $\overline{X}$ corresponds to a bipartite subgraph of X that is either regular of degree 1 or degree 2, it is again easy to see that X is 3-edge-colorable.

Third, $\overline{X}$ may be a cubic graph. Since G/N is also solvable, we know that $\overline{X}$ is 3-edge-colorable by induction. Each color class lifts to a 1-factor of X so that X is 3-edge-colorable too.

Thus we may assume that every nontrivial normal subgroup of G has nonempty intersection with S.

If the group $\langle a\rangle$ generated by a contains a nontrivial normal subgroup N, then $b \notin N$, as this would imply that $b \in \langle a\rangle$, that is, that G is abelian (cyclic). By the above assumption, $a \in N$, so $\langle a\rangle = N$. This implies that X itself is a generalized Petersen graph. F. Castagna and G. Prins [1] proved that all generalized Petersen graphs, other than the Petersen graph, are 3-edge-colorable. The Petersen graph is not a Cayley graph [4, p. 322]. Thus, we may assume that $\langle a\rangle$ contains no nontrivial normal subgroups of G.

We now use the previous assumption to reach two useful conclusions. If $ba^i = a^j b$ for some $i, j \in \{1, 2, \ldots, r-1\}$, then $\langle a^i, a^j\rangle$ is a normal subgroup of G contained in $\langle a\rangle$. By assumption we know this is not the case. Thus we conclude that

1. b does not commute with any a^i for $i = 1, 2, \ldots, r-1$ and
2. $ba^i b \notin \langle a\rangle$ for $i = 1, 2, \ldots, r-1$.

Since G is solvable, G contains a nontrivial abelian normal subgroup N (see [2, Prob. 11, p. 107]). We know that $S \cap N \neq \emptyset$ and $S \not\subseteq N$ (since X is connected). We consider the case that $a \in N$ and $b \notin N$. Then $[G : N] = 2$. Since N is abelian, $|a| = |bab| = r$ and by 2 above, $N = \langle a\rangle \times \langle bab\rangle$. Thus, $|N| = r^2$. Note that

$$N = \langle a\rangle \cup bab\langle a\rangle \cup ba^2b\langle a\rangle \cup \cdots \cup ba^{r-1}b\langle a\rangle$$

and

$$bN = b\langle a\rangle \cup ab\langle a\rangle \cup a^2b\langle a\rangle \cup \cdots \cup a^{r-1}b\langle a\rangle.$$

Denote the cycle of $ba^ib\langle a\rangle$ by

$$C_i = v_{i,0}v_{i,1}\ldots v_{i,r-1}v_{i,0},$$

where $v_{i,j} = ba^iba^j$ for $i, j \in \{0, 1, \ldots, r-1\}$, and denote the cycle of $a^ib\langle a\rangle$ by

$$D_i = u_{i,0}u_{i,1}\ldots u_{i,r-1}u_{i,0},$$

where $i, j \in \{0, 1, \ldots, r-1\}$. The b-edge incident with $v_{i,j} = ba^iba^j$ is also incident with $u_{j,i} = a^jba^i = ba^iba^jb$ because N is abelian and both ba^ib and a^j are in N. Let $P_i = C_i - v_{i,i}v_{i,i+1}$ and $Q_i = D_i - u_{i,i-1}u_{i,i}$ for $i = 0, 1, \ldots, r-1$ and with subscripts reduced modulo r. Then the union of all paths P_i and Q_i and the b-edges $v_{i,i}u_{i,i}, v_{i,i+1}u_{i+1,i}$, $i = 0, 1, \ldots, r-1$, is a Hamilton cycle of X. Thus, X is 3-edge-colorable.

We now consider the case that $b \in N$ and $a \notin N$. Observe that $\langle b \rangle \neq N$, for otherwise, $aba^{-1} \in N$ implies that $ab = ba$, which in turn implies that G is abelian. Now N is abelian and $b \in N$, so by 2 above, $a^k \notin N$ for any $k = 1, 2, \ldots, r-1$ and $a^i b a^{-i} \neq a^j b a^{-j}$ for $i \neq j$ (otherwise, a^{i-j} commutes with b).

Consider an auxiliary Cayley graph $X' = X(N; S')$ on N with $S' = \{b, aba^{-1}\}$. Both elements have order 2 and define edges in a Cayley graph on an abelian group. Thus, X' consists of vertex-disjoint 4-cycles. A typical 4-cycle has the form y, yb, $ybaba^{-1}$, $ybaba^{-1}b, y$. Back in the original graph X, each vertex z of a 4-cycle corresponds to the r-cycle $z, za, za^2, \ldots, za^{r-1}$. Notice that there is an edge joining yba and $ybaba^{-1}a = ybab$ and an edge joining ya and $ybaba^{-1}ba = yab$. Hence, the typical 4-cycle mentioned above lifts to a $4r$-cycle in X by removing the edges (y, ya), (yb, yba), $(ybaba^{-1}, ybaba^{-1}a)$, and $(ybaba^{-1}b, ybaba^{-1}ba)$ from the four r-cycles corresponding to the vertices of the 4-cycle and by replacing them with the four edges (y, yb), $(ybaba^{-1}, ybaba^{-1}b)$, $(ya, ybaba^{-1}ba)$, and $(yba, ybaba^{-1}a)$. Hence, X has a 2-factor made up of cycles of length $4r$, so X is 3-edge-colorable. This completes the proof of the theorem. □

REFERENCES

[1] F. CASTAGNA AND G. PRINS, *Every generalized Petersen graph has a Tait coloring*, Pacific J. Math., 40 (1972), pp. 53–58.

[2] D. S. DUMMIT AND R. S. FOOTE, *Abstract Algebra*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[3] F. HOFFMAN, S. C. LOCKE, AND A. D. MEYEROWITZ, *A note on cycle double covers in Cayley graphs*, Math. Pannon. 2, Issue 1, Univ. Miskolc, Miskolc, 1991, pp. 63–66.

[4] D. A. HOLTON AND J. SHEEHAN, *The Petersen graph*, Austral. Math. Soc. Lect. Ser. 7, Cambridge University Press, Cambridge, 1993.

[5] B. JACKSON, *On Circuit Covers, Circuit Decompositions and Euler Tours of Graphs*, British Combinatorics Conference, 1993, preprint.

[6] F. JAEGER, *Flows and generalized coloring theorems in graphs*, J. Combin. Theory Ser. B, 26 (1979), pp. 205–216.

[7] ———, *A survey of the cycle double cover conjecture*, in Cycles in Graphs, B. Alspach and C. Godsil, eds., Ann. Discrete Math. 27, North-Holland, Amsterdam, 1985, pp. 1–12.

[8] ———, *Nowhere-zero flow problems*, in Selected Topics in Graph Theory 3, L. Beineke and R. Wilson, eds., Wiley, New York, 1988, pp. 71–95.

[9] L. LOVÁSZ, Problem 11 in *Combinatorial Structures and Their Applications*, in Proc. Calgary International Conference on Combinatorial Structures and Their Applications, R. Guy, H. Hanani, N. Sauer, and J. Schonheim, eds., Gordon and Breach, New York, 1970, p. 497.

[10] L. LOVÁSZ, *Combinatorial Problems and Exercises*, North-Holland, Amsterdam, 1979, p. 75.

[11] G. J. MINTY, *A theorem on three-coloring the edges of a trivalent graph*, J. Combin. Theory, 2 (1967), pp. 164–167.

[12] R. STONG, *On 1-factorizability of Cayley graphs*, J. Combin. Theory Ser. B, 39 (1985), pp. 298–307.

[13] W. TUTTE, *A contribution on the theory of chromatic polynomial*, Canad. J. Math., 6 (1955), pp. 80–91.

[14] C.-Q. ZHANG, *Minimum cycle coverings and integer flows*, J. Graph Theory, 14 (1990), pp. 537–546.

SIAM J. DISCRETE MATH.
Vol. 9, No. 1, pp. 155–166, February 1996

014

TIGHT BOUNDS FOR DYNAMIC STORAGE ALLOCATION *

MICHAEL G. LUBY†, JOSEPH (SEFFI) NAOR‡, AND ARIEL ORDA§

Abstract. This paper is concerned with on-line storage allocation to processes in a dynamic environment. This problem has been extensively studied in the past. We provide a new, tighter bound for the competitive ratio of the well-known First Fit algorithm. This bound is obtained by considering a new parameter, namely the maximum number of concurrent active processes. We observe that this bound is also a lower bound on the competitive ratio of any deterministic on-line algorithm. Our second contribution is an on-line allocation algorithm that uses coloring techniques. We show that the competitive ratio of this algorithm is the same as that of First Fit. Furthermore, we indicate that this algorithm may be advantageous in certain applications. Our third contribution is to analyze the performance of randomized algorithms for this problem. We obtain lower bounds on the competitive ratio that are close to the best deterministic upper bounds.

Key words. on-line algorithms, memory management, dynamic storage allocation, bandwidth allocation, First Fit, interval graph

AMS subject classifications. 60C05, 60E15, 68Q25, 68R10

1. Introduction.

This paper is concerned with a classic problem in computer science: the allocation of area in a one-dimensional storage device to processes in a dynamic environment. In a typical setting, at the time of arrival of a process, it is allocated storage area. This area is required to form a contiguous location in the storage device. Once allocated, a process cannot be moved to a different location. At some later point in time (*unknown* at the time of allocation), the process leaves, thereby liberating the storage area it occupied and making it available to other processes. As a result, wasted space or "holes" are generated over time in the storage device. The objective is to find an allocation algorithm that minimizes the wasted space. This is a typical on-line setting in which decisions must be based upon the current state without knowledge of future events.

The processes form an interval graph denoted by $G = (V, E)$; a process corresponds to the interval defined (on the time axis) between its arrival and departure times. Each interval is associated with a weight that is equal to the storage area required by the corresponding process. The allocation of storage area can be thought of as a *weighted coloring* of G: the color of interval i is a range $[a_i, b_i]$ (where a_i and b_i are integers), such that $b_i - a_i + 1$ is equal to the weight of interval i and there is no intersection between the ranges of two intervals that overlap. The objective is to minimize b_i, where b_i is taken over all intervals.

We denote the maximum weight, taken over all intervals, by $W_{\max}$. Let $\omega^*(G)$

* Received by the editors October 1, 1993; accepted for publication (in revised form) March 9, 1995. A preliminary version of this paper appeared in *Proceedings of the 5th Annual ACM–SIAM Symposium on Discrete Algorithms*, Arlington, VA, 1994, pp. 724–732.

† International Computer Science Institute, University of California at Berkeley, CA 94704 (luby@icsi.berkeley.edu). The research of this author was supported in part by National Science Foundation grant CCR-9016468 and United States–Israel Binational Science Foundation (BSF) grant 89-00312.

‡ Department of Computer Science, Technion, Haifa 32000, Israel (naor@cs.technion.ac.il). The research of this author was supported in part by United States–Israel Binational Science Foundation (BSF) 92-00225 grant and the Technion Argentinian Research Fund. Part of this research was performed while this author was visiting the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Rutgers University.

§ Department of Electrical Engineering, Technion, Haifa 32000, Israel (ariel@ee.technion.ac.il).

denote the weight of the heaviest clique in G, where the weight of a clique is the sum of the weights of the intervals in the clique. Clearly, $\omega^*(G)$ is a lower bound on the area used by *any* allocation algorithm.

We note that the problem of determining an optimal weighted coloring for a given set of intervals (i.e., the off-line version) is NP-complete in the strong sense [9, p. 226]. Recently, polynomial-time approximation algorithms for the off-line version were obtained. Kierstead [10] showed that if the intervals are sorted by their weights, then the competitive ratio of First Fit is 80. Later, Kierstead [11] gave a different approximation algorithm that achieves a competitive factor of 6. Both algorithms use $\omega^*(G)$ as a benchmark for measuring the approximation factor.

The question of evaluating the performance of on-line algorithms was addressed by Sleator and Tarjan [19], who argued that the traditional approach of measuring the worst-case behavior does not seem appropriate for many on-line algorithms. Therefore, they suggested a different measure, the *competitive ratio.* The performance of an on-line algorithm is compared with the performance of an *optimal* off-line algorithm that knows the sequence of events in advance. The maximum ratio between their respective performances, taken over all sequences, is called the competitive ratio. Extensive work has been done in recent years for finding the competitive ratio for different problems such as paging [19], [8], servers in a metric space [14], and managing a linked list [19]. We also adopt the competitive ratio as our performance measure. Notice that for dynamic storage allocation, it follows from Kierstead's work [11] that the competitive ratio (up to a constant factor) is the ratio between the area used by the on-line algorithm and $\omega^*(G)$.

1.1. Previous work. Dynamic storage allocation has been an important area of research since the 1950s [12], [18]. Coffman [3] surveyed the results and developments in this area until the early 1980s. This area of research is also strongly linked with dynamic bin packing (see [4], [5]).

There are two natural heuristics that were developed early on for dynamic storage allocation: *First Fit* and *Best Fit.* First Fit finds the first free area that can fit the requirement of the current process; Best Fit finds the free area that best fits the requirement of the current process, i.e., the one that causes minimal fragmentation. A different category of methods for dynamic storage allocation constitute *segregated storage methods* [18]. These methods have many variations. The idea here is to partition the memory into blocks, such that in each block, only processes that have the same (or similar) requirement are allocated.

Denote the maximum weight, taken over all intervals, by $W_{\max}$. In the early 1970s, Robson [15], [16] gave an algorithm that had a competitive ratio of $O(\log(W_{\max}))$. He also showed that this is the best possible bound up to constant factors. Independently, Woodall [20] showed that the competitive ratio of First Fit is $O(\log(\omega*))$. Subsequently, Robson [17] showed that the competitive ratio of First Fit is also $O(\log(W_{\max}))$. It is interesting to note that Robson [17] also showed that the competitive ratio of Best Fit can be as bad as $W_{\max}$. As previously mentioned, algorithms for the off-line case with constant approximation factors have been recently obtained [10], [11].

1.2. Our contribution.

1.2.1. First Fit. Our first contribution is to provide tighter bounds for First Fit by considering a *new* parameter for analyzing the competitive ratio, i.e., the maximum number of concurrent active processes in memory.

Let the chromatic number of G be denoted by $\chi(G)$. Since interval graphs are perfect, $\chi(G) = \omega(G)$, where $\omega(G)$ denotes the clique number of G. Note that $\chi(G)$ is also the maximum number of concurrent active processes.

We show that First Fit achieves a competitive ratio of $\min\{O(\log(W_{\max})), O(\log(\chi(G)))\}$. This is an improvement over Robson [17], since, in general, $W_{\max}$ and $\chi(G)$ are incomparable. Tightening the analysis of First Fit is important, since it is a natural heuristic that is widely used.

It follows from the lower bounds proved by [15], [20] that our results are tight.

1.2.2. The Coloring algorithm. We next analyze a second deterministic online algorithm, which we call the *Coloring algorithm.* It belongs to the category of segregated storage methods [18] and uses an altogether different strategy than First Fit.

Consider a portion of storage area that is used by First Fit. Over time, we will generally see processes of diverse weights being allocated to that same portion. In fact, it is a fundamental property of First Fit to use "holes" created by processes that left the system in order to allocate smaller weight processes. As discussed in what follows, this property (i.e., allocating over time processes of various weights within the same area) is often a deficiency. Moreover, in an empirical study comparing different dynamic storage allocation methods, Detlefs et al. [6] show that hybrid algorithms that allocate differently small and large requirements are very efficient in practice.

We now describe the Coloring algorithm. This algorithm never places processes of different weights (up to a constant factor of 2) within the same area. It assigns slots of storage area to each process "type" (according to weights); once a slot is assigned, the space within it is occupied only by processes of its type.

We show that the Coloring algorithm achieves the same (tight) worst-case competitive ratio as First Fit. To the best of our knowledge, the competitive factor of algorithms belonging to the category of segregated storage has not been previously analyzed. In the following we briefly discuss several applications in which this type of algorithms can be expected to be efficient.

Consider the classical problem of allocating dynamic memory. It is well known that segregated storage algorithms are much more efficient than First Fit in finding free space for a new process. (See, e.g., [18], [21]; see also [2] for an efficient implementation of First Fit). Indeed, by using standard techniques, the Coloring algorithm can allocate a new process within O(1) operations.

Dynamic storage allocation algorithms also arise in typical problems concerning storage of commodities. In such environments, one often needs to "prepare" a storage area that fits a certain type of commodities. Since the weight of a commodity is usually related to its type, the Coloring algorithm saves the need to "reshape" storage areas for different types of commodities, whereas First Fit would frequently demand such actions.

Yet another field to which dynamic storage allocation algorithms are applicable is that of communication networks. Consider for example a radio network, in which transmission bandwidth is divided into time, or frequency, slots. Transmitting stations dynamically request variable numbers of consecutive slots from a network controller. After completing transmission, a station informs the controller on the release of its allocated bandwidth. Obviously, the controller should attempt to allocate bandwidth upon demand so as to minimize the total amount of used bandwidth. The application of dynamic storage allocation algorithms in such an environment is straightforward. Typically, there is a strong relation between the size of the requested bandwidth and

the application that the corresponding transmission serves. This means that a receiving station, which attempts to follow the transmissions of a particular application, would need to be tuned to all portions of bandwidth in which such an application may be transmitted. First Fit would demand such a station to readjust its tuning frequently; the Coloring algorithm, on the other hand, guarantees that the receiving station would need just a few initial tunings.

1.2.3. Randomization. There has been extensive work devoted to exploiting and understanding the power of randomization in an on-line environment [1]. The idea is that randomized on-line algorithms might exhibit a better competitive ratio than deterministic ones, since they manage to "outsmart" the adversary sometimes.

A randomized on-line algorithm is defined to be a probability distribution over a space of deterministic on-line algorithms. There are several types of adversaries in this context, which differ by the information that the adversary is allowed to have. The most commonly considered adversary (and the one considered in this paper) is the *oblivious* one, who must construct the request sequence in advance based on the description of the algorithm but has no access to the random choices made by the algorithm.

A randomized on-line algorithm A is said to have competitive ratio α if, for each request sequence, the ratio between the expected value of the performance of A and the off-line performance is at most α.

We show that, for the dynamic storage allocation problem, allowing randomized algorithms is not very helpful. In particular, we prove a lower bound of

$$\min\{O(\log W_{\max}/\log\log W_{\max}), O(\log \chi(G)/\log\log \chi(G))\}$$

on the competitive ratio of any randomized algorithm. Note that these lower bounds are only $\min\{O(\log\log W_{\max}), O(\log\log \chi(G))\}$ away from the deterministic lower and upper bounds. This is in contrast to the exponential gap between the competitive ratio of deterministic on-line algorithms to the competitive ratio for randomized on-line algorithms for the somewhat related paging problem: If k denotes the number of pages in the cache, then k is the competitive ratio of the best deterministic algorithm; on the other hand, it is known [8] that randomized algorithms can achieve a competitive ratio of $\log k$.

We conjecture that our randomized lower bounds can be further improved to match those of the deterministic case.

1.2.4. Multiple storage devices. Our results can be extended to the case where there are several storage devices, and a process can be allocated to any of the devices. Simple analysis shows that having several devices has no (negative or positive) effect on the performance of dynamic storage allocation. The details can be found in [13].

2. Upper bounds. In this section we consider the First Fit and Coloring algorithms.

To clarify the exposition, we adopt the following notation. A process is referred to as an "item" where the item size is equal to the weight of the corresponding process. The storage device consists of cells, or locations, that are addressed consecutively from zero. The space that is used by an algorithm is equal to the value of the highest address of a location that was used by the algorithm. We denote the chromatic number of G, $\chi(G)$, by k.

2.1. First Fit. First Fit is defined as follows. When a process appears, First Fit searches the memory for the first available area that satisfies the requirement of the process. In other words, the item is allocated in the area that has the lowest possible address.

We now bound the competitive ratio of First Fit. The idea is that locations with a "high" address will not contain "small" items. These notions are defined more rigorously in the following. Let $W = \lfloor W_{\max}/k \rfloor$.

LEMMA 2.1. *Items that have size less than or equal to $4W$ are always completely contained in the first $5\omega^*$ locations.*

Proof. Suppose First Fit needs to allocate area to an item I that has size $U \leq 4W$, and assume further that this cannot be done within the first $5\omega^*$ locations. In this case, the size of the largest hole in the first $5\omega^*$ locations is at most $U-1$. Since the number of holes is at most k (including the area above the highest item), this means that the total area occupied by holes within the first $5\omega^*$ locations cannot exceed $k(U-1) < 4W_{\max} \leq 4\omega^*$. The area that is occupied by currently active items (of any size) cannot be more than $\omega^* - U$. Hence, there must be contiguous free space of size U within the first $5\omega^*$ locations, contradicting the assumption. □

For ease of presentation, we first bound the competitive ratio for the case where all weights are powers of two. Let ℓ be the smallest integer such that $2^\ell \geq 4W$. We show that in this case the following invariant holds:

> An item of size 2^j, where $j \geq \ell$, can always be allocated within the first $(6 + j - \ell)\omega^*$ locations.

It follows from the invariant that the competitive ratio of First Fit is at most $5+\log_2 k$ by substituting $j = \log_2 W_{\max}$. (Note that j is an integer.)

The proof of the invariant is by induction on j. We first prove the base case where $j = \ell$. It follows from Lemma 2.1 that items that have size smaller than or equal to $4W$ are completely contained in the first $5\omega^*$ locations. Since we assumed that all item sizes are powers of 2, it follows that in locations with address higher than $5\omega^*$ the minimum item size is 2^ℓ. This implies that the minimum size of a hole above address $5\omega^*$, except perhaps for the first hole, is also 2^ℓ. Suppose that an item of size 2^ℓ cannot be allocated in the first $6\omega^*$ locations. This means that the only hole in locations with address between $5\omega^*+1$ and $6\omega^*$ is the first hole, which must have size less than 2^ℓ. This implies that the total size of the items that are currently active is strictly greater than ω^*, a contradiction. The inductive step is argued similarly, thus proving the invariant.

The analysis of First Fit in the general case is more subtle. Notice that it is not the case that one can round up the sizes to the closest power of two and then argue that at most a factor of two in the competitive ratio is lost.

We first prove a technical lemma regarding harmonic numbers. Let $H_n = \sum_{i=1}^{n} \frac{1}{i}$.

LEMMA 2.2. *Assume that $p \geq 1.5q$ and $q \geq 6$. Then $H_p - H_q \geq 0.3$.*

Proof. We use the following known bound on H_n [12, p. 74]:

$$H_n = \ln n + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \frac{1}{120n^4} - \varepsilon$$

where γ is Euler's constant and $0 < \varepsilon < \frac{1}{252n^6}$. This implies that

$$H_p - H_q \geq \ln\left(\frac{p}{q}\right) + \frac{1}{2p} - \frac{1}{2q} + \frac{1}{12q^2} - \frac{1}{12p^2} + \frac{1}{120p^4} - \frac{1}{120q^4} - \frac{1}{252p^6}.$$

Since $p \geq 1.5q$, we have that

$$\frac{1}{12q^2} - \frac{1}{12p^2} \geq \frac{1}{120q^4} - \frac{1}{120p^4} + \frac{1}{252p^6}.$$

Also, since $q \geq 6$, we have that

$$\ln\left(\frac{p}{q}\right) + \frac{1}{2p} - \frac{1}{2q} > 0.3,$$

which concludes the proof. □

LEMMA 2.3. *The competitive ratio of First Fit is at most* $9 + 4\ln k$.

Proof. We claim that items of size at most j, where $j \geq W + 1$, are always allocated by First Fit in locations with address lower than

$$5\omega^* + \frac{\omega^*}{0.3} \cdot \sum_{i=W+1}^{j} \frac{1}{i}.$$

The claim is proved by induction. From Lemma 2.1, it follows that the claim holds for the case where $j \leq 4W$. We assume the claim holds for all values smaller than or equal to j (and at least $W + 1$) and show that it holds for $j + 1$. Let

$$L_i = 5\omega^* + \frac{\omega^*}{0.3} \cdot \sum_{\ell=W+1}^{i} \frac{1}{\ell}.$$

Assume the claim does not hold for $j+1$. This implies that in the first L_{j+1} locations, there are no holes of size bigger than j. The following holds for locations between $(L_{i-1}+1)$ to L_i for $W+1 \leq i \leq j+1$, assuming that each item is completely contained in the area between $(L_{i-1} + 1)$ and L_i, for some i:

> Any contiguous occupied area is of size at least i, and the holes are of size at most j. Hence, the total occupied area in locations between $(L_{i-1} + 1)$ to L_i is at least
>
> $$\frac{i}{i+j} \cdot (L_i - L_{i-1}).$$

Notice that, in the above, each hole is paired with the item below it. Thus, if an item is not completely contained in the area between $(L_{i-1} + 1)$ and L_i, then the area it occupies within $(L_{i-2} + 1)$ and L_{i-1} would not be not paired with any hole. This means that such an item can be paired with the first hole within $(L_{i-1} + 1)$ and L_i, yielding that it is no longer necessary to assume that each item is completely contained in the area between $(L_{i-1} + 1)$ and L_i.

Summing up over all possible values of i, we get that the area that is occupied within the first L_{j+1} locations is at least

$$\sum_{i=W+1}^{j+1} (L_i - L_{i-1}) \cdot \frac{i}{i+j} = \frac{\omega^*}{0.3} \cdot \sum_{i=W+1}^{j+1} \frac{1}{i} \cdot \frac{i}{i+j} = \frac{\omega^*}{0.3} \cdot \sum_{i=W+j+1}^{2j+1} \frac{1}{i}.$$

Since $j \geq 4W$, it follows that $\frac{2j+1}{W+j+1} \geq 1.5$, and $W + j + 1 \geq 6$. Hence, by Lemma 2.2,

$$\sum_{i=W+j+1}^{2j+1} \frac{1}{i} \geq 0.3,$$

implying that the area occupied within the first L_{j+1} locations is at least ω^*. This means that no new item can appear, contradicting the assumption that the above claim does not hold for $j+1$.

Hence, the total area occupied by First Fit is at most

$$5\omega^* + \frac{\omega^*}{0.3} \cdot \sum_{i=W+1}^{W_{\max}} \frac{1}{i}.$$

Since $\ln n < H_n < \ln n + 1$, we get that the above is at most

$$5\omega^* + \frac{\omega^*}{0.3} \cdot \left(1 + \ln\left(\frac{W_{\max}}{W+1}\right)\right) \leq \omega^* \cdot \left(\frac{2.5 + \ln k}{0.3}\right) \leq \omega^* \cdot (9 + 4\ln k),$$

thus proving the lemma. □

We thus have the following theorem.

THEOREM 2.4. *The competitive ratio of First Fit is bounded by* $\min\{O(\log W_{\max}), O(\log k)\}$.

Proof. This follows from Lemma 2.3 together with Robson's proof [17]. □

Robson [15] and Woodall [20] showed a lower bound of $\Omega(\log W_{\max})$ on the competitive ratio of any deterministic on-line algorithm. We observe that, in the dynamic allocation problem instance, they construct for the proof of the lower bound $k = W_{\max}$. Hence, their lower bound extends immediately to a lower bound of $\min\{\Omega(\log W_{\max}), \Omega \log k\}$ on any deterministic algorithm. In fact, this observation can be generalized to hold in a more general setting, i.e., for $k < W_{\max}$, by slightly modifying the dynamic allocation problem instance in the proof of Woodall [20]. In this instance, items are inserted in iterations, where in the ith iteration, items of size 2^{i-1} are inserted. (Initially, $i = 1$.) The value of ω^* is fixed in advance by the adversary, and $W_{\max} = \omega^*$. A careful examination of Woodall's proof reveals that if the insertion process starts by inserting items of size 2^j, for any $j > 1$, then a lower bound of $\Omega(\log(\omega^*/2^j))$ on the competitive ratio can be proved. We note that, in this case, $k = O(\omega^*/2^j)$. Thus, a lower bound of $\Omega(\log k)$ on the competitive ratio of any deterministic on-line algorithm is proved for arbitrary $k < W_{\max}$. We conclude with the following corollary.

COROLLARY 2.5. *The competitive ratio of First Fit is* $\min\{\Theta(\log W_{\max}), \Theta(\log k)\}$.

2.2. The coloring algorithm. In this section we analyze an algorithm for dynamic allocation, called the Coloring algorithm. The algorithm splits items into sets according to their sizes. An item of size l is assigned to set r, where $2^{r-1} < l \leq 2^r$. At any given time, each set r is assigned zero or more slots in memory, each of size 2^r. The assignment of slots is performed as follows. Suppose a new item belonging to the rth set appears. The Coloring algorithm tries to allocate it within a free slot assigned to that set. If no such slot is available, the algorithm produces a new slot by increasing the used area by 2^r. A slot assigned to set r (that is, the space allocated to it) is reserved forever for items belonging to that set. Thus, a location is never allocated with items belonging to different sets.

Let G_r denote the subgraph of G induced by intervals that have weight w, where $2^{r-1} < w \leq 2^r$. Denote by $\chi(G_r)$ the maximal number of concurrent items belonging to the rth set.

We observe that with respect to each set r, the Coloring algorithm uses at most twice the optimal amount of space by the following argument. Suppose that the size of each item in the rth set is 2^r. Since the intervals (corresponding to items) are

sorted by their left endpoint, the greedy algorithm produces an optimal coloring. The factor of 2 is due to the fact that the size of each item in G_r is "rounded" up to the nearest power of 2. The following lemma seems to be folklore.

LEMMA 2.6. *The Coloring algorithm achieves a competitive ratio of* $O(\log(W_{\max}))$.

Proof. The amount of space assigned to a set r is precisely $\chi(G_r) \cdot 2^r$. For all values of r we have that $\chi(G_r) \cdot 2^r \leq 2 \cdot \omega^*$, meaning that each set occupies space of at most $2 \cdot \omega^*$ locations. The number of distinct sets is at most $\log(2W_{\max})$. Thus, the overall space used by the Coloring algorithm is at most $2 \cdot (\log(W_{\max}) + 1) \cdot \omega^*$. The lemma follows. □

LEMMA 2.7. *The total number of locations allocated to all sets* r *for which* $2^r \leq \frac{W_{\max}}{k}$ *is at most* $2 \cdot W_{\max}$.

Proof. As was observed in the previous proof, the total area assigned to a set r is precisely $\chi(G_r) \cdot 2^r$. Clearly, $\chi(G_r) \leq k$. This means that the total amount of space allocated to all sets for which $2^r \leq \frac{W_{\max}}{k}$ is at most

$$\sum_{r=0}^{\lfloor \log(\frac{W_{\max}}{k}) \rfloor} 2^r \cdot k \leq 2k \cdot \frac{W_{\max}}{k} = 2 \cdot W_{\max}. \qquad \square$$

LEMMA 2.8. *The total amount of space allocated to all sets* r *for which* $2^r \geq \lfloor \frac{W_{\max}}{k} \rfloor$ *is at most* $2\log(2k) \cdot \omega^*$.

Proof. The number of distinct sets containing items of size larger than $\lfloor \frac{W_{\max}}{k} \rfloor$ is at most $\log(2k)$. Since each set occupies at most $2\omega^*$ locations, we get that the total area used is $2\log(2k) \cdot \omega^*$. □

THEOREM 2.9. *The competitive ratio of the Coloring algorithm is* $\min\{\Theta(\log W_{\max}), \Theta(\log k)\}$.

Proof. The proof follows directly from Lemmas 2.6, 2.7, 2.8, and the discussion preceding Corollary 2.5. □

2.3. Comparison. When comparing First Fit with the Coloring algorithm, it should be emphasized that neither algorithm dominates the other, i.e., for each algorithm there exist examples in which it outperforms the other. In particular, there are instances in which First Fit is near-optimal, whereas the Coloring algorithm reaches the worst-case bound.

The Coloring algorithm tends to perform well compared to First Fit in heterogeneous systems, i.e., when concurrency of items of various sizes is common. Weinstock and Wulf [21] describe an implementation of a segregated storage scheme called *Quick Fit*. It is interesting to note that they indicate that Quick Fit utilizes storage effectively in practice. The reason they provide for this phenomenon is that if a particular storage area has been allocated and then deallocated, there is a high probability that it will be allocated again.

3. A randomized lower bound. In this section we prove a lower bound of $O(\omega^* \log \omega^* / \log\log \omega^*)$ on the space required by any randomized on-line dynamic storage allocation algorithm. The lower bound will be proved for the case of a randomized oblivious adversary. (See §1.)

Our proof will use a corollary due to Yao [22] of von Neumann's minimax principle. Let $\mathcal{P}$ be a probability distribution defined on the input. Yao's result implies in our case that the expected value (with respect to $\mathcal{P}$) of the competitive ratio of any deterministic on-line algorithm is always a lower bound on the competitiveness of the best randomized on-line algorithm. Hence, our goal is to generate a "hard"

distribution on the input, i.e., a distribution such that the expected value of the space required by any deterministic on-line algorithm is at least $O(\omega^* \log \omega^* / \log \log \omega^*)$.

Consider the following oblivious adversary *Adv*. *Adv* chooses a dynamic storage allocation instance probabilistically, and thus, *Adv* (implicitly) defines the probability distribution $\mathcal{P}$ on the input. *Adv* first decides on the value of ω^* before the run of the algorithm. The strategy of *Adv* is to insert into the system items of growing sizes and then to delete half of them at random. *Adv* uses item sizes that are powers of g, where the value of g will be determined later. Without loss of generality it is assumed that ω^* is a power of 2, say $\omega^* = 2^r$. More specifically, *Adv* works in iterations, where in each it performs the following.

Insertion of items at iteration i. Denote by the random variable ω_i the total area occupied by items currently residing in the system (initially 0). *Adv* inserts $\lfloor \frac{\omega^* - \omega_i}{g^{i-1}} \rfloor$ items of size g^{i-1} into the system.

Deletion of items at iteration i. Adv deletes each item with probability $\frac{1}{2}$, independently of the other items.

Starting with the the first iteration, *Adv* sequentially performs the insertion followed by the deletion steps for all iterations up to $i = \lfloor (r-1) \cdot \log_g 2 \rfloor + 1$.

Throughout the analysis, we mark at each iteration, following the insertion step, a subset of the items that currently reside in the system. Each such item will be termed as *marked.* A marked item may become unmarked at some later iteration. We denote by L_i the list of marked items, after the insertion step of iteration i, ordered according to their position in the storage device from bottom to top. The marking process is performed as follows.

- **Rule A:** At each iteration, all the new items inserted are marked.
- **Rule B:** At each iteration $i > 1$, we select approximately $1/(g+2)$ fraction of the marked items that resided in the system in the previous iteration (and were not deleted in the last deletion step). All other previously marked items become unmarked. The precise procedure is as follows. In the list L_{i-1}, we skip the first g items and then search for the next item in the list that was not deleted in the deletion step of iteration $i-1$. This item is marked and inserted into the list L_i. The process is repeated until the list L_{i-1} is exhausted.

We stress that the notion of a marked item is solely for the purpose of analysis.

LEMMA 3.1. *Following the insertion step of the ith iteration, the expected number of marked items is at least*

$$\frac{i \cdot 2^{r-1}}{(g+2)^{i-1}} - 2.$$

Proof. The claim clearly holds for $i = 1$. Assuming inductively that it holds for iteration $i-1$, we show that it also holds for iteration i. By the above marking rules, marked items are either newly inserted items or a subset of the items marked in the previous iteration.

The total area occupied by items, at any time, cannot be more than ω^*. Since each item is deleted with probability $\frac{1}{2}$, the expected value of the occupied area, after the deletion step of any iteration, cannot be more than $\frac{\omega^*}{2} = 2^{r-1}$. This means that, for all i, the expected value of ω_i is at most 2^{r-1}. Hence, the expected number of newly inserted items at iteration i is at least

$$\frac{2^{r-1}}{g^{i-1}} - 1,$$

and all these items are marked by Rule A.

We now evaluate the expected number of items marked by Rule B. For the purpose of the analysis, we assume that L_{i-1} is of infinite length; however, $|L_{i-1}|$ still refers to the original length of the list.

Let d_j be a random variable that counts the number of items in L_{i-1} between the $(j-1)$st marked item and the jth marked item. By Rule B and by the deletion strategy of Adv, $(d_j - g)$ is geometrically distributed with parameter $\frac{1}{2}$. This means that $\mathrm{Exp}[d_j] = g + 2$. Let T be a random variable that denotes the smallest t such that

$$\sum_{j=1}^{t} d_j > |L_{i-1}|.$$

It is clear that the expected number of items marked by Rule B is $\mathrm{Exp}[T] - 1$. Notice that a stopping rule can be associated with Rule B in a natural way: Stop whenever the list L_{i-1} is exhausted. Hence, Wald's identity [7] can be applied in this case; it states that

$$\mathrm{Exp}[T] \cdot \mathrm{Exp}[d_i] = \mathrm{Exp}\left[\sum_{j=1}^{T} d_j\right].$$

Clearly,

$$|L_{i-1}| < \mathrm{Exp}\left[\sum_{i=1}^{T} d_i\right] \leq |L_{i-1}| + g + 2.$$

Therefore, we get that the expected number of items marked by Rule B is at least

$$\frac{|L_{i-1}|}{g+2} - 1.$$

Let ℓ_i be a random variable that denotes the cardinality of L_i. From the above it follows that

$$\mathrm{Exp}[\ell_i | \ell_{i-1}] \geq \frac{\ell_{i-1}}{g+2} - 1 + \frac{2^{r-1}}{g^{i-1}} - 1.$$

Since $\mathrm{Exp}[\ell_i|\ell_{i-1}]$ is a linear function of ℓ_{i-1}, it follows that

$$\mathrm{Exp}[\ell_i] \geq \frac{\mathrm{Exp}[\ell_{i-1}]}{g+2} - 1 + \frac{2^{r-1}}{g^{i-1}} - 1.$$

By the inductive assumption,

$$\mathrm{Exp}[\ell_{i-1}] \geq \frac{(i-1)\cdot 2^{r-1}}{(g+2)^{i-2}} - 2.$$

Thus, the expected number of items marked by Rules A and B is at least

$$\mathrm{Exp}[\ell_i] \geq \left(\frac{(i-1)\cdot 2^{r-1}}{(g+2)^{i-2}} - 2\right) \cdot \frac{1}{g+2} - 2 + \frac{2^{r-1}}{g^{i-1}}$$

$$\geq \frac{i \cdot 2^{r-1}}{(g+2)^{i-1}} - 2 + \frac{2^{r-1}((g+2)^{i-1} - g^{i-1})}{g^{i-1}(g+2)^{i-1}} - \frac{2}{g+2}$$

$$\geq \frac{i \cdot 2^{r-1}}{(g+2)^{i-1}} - 2 + \frac{(g+2)^{i-1} - g^{i-1}}{(g+2)^{i-1}} - \frac{2}{g+2}$$

$$= \frac{i \cdot 2^{r-1}}{(g+2)^{i-1}} - 2 + \frac{1}{g+2}\left(g + 2 - \left(\frac{g}{g+2}\right)^{i-2} \cdot g - 2\right)$$

$$\geq \frac{i \cdot 2^{r-1}}{(g+2)^{i-1}} - 2. \quad (1) \qquad \square$$

We define the distance between two items residing in the storage device as the distance between their centers.

LEMMA 3.2. *Following the insertion step of the ith iteration, the distance between two consecutive marked items is at least* $g^{i-1}/2$.

Proof. We prove the lemma by induction. The claim trivially holds for the first iteration. Assuming that it holds for iteration $i-1$, we prove it for iteration i. Let X and Y be two consecutive items in L_i. If one of them was marked by Rule A, then its size is g^{i-1}; thus the distance between the two centers is at least $g^{i-1}/2$. Suppose, then, that both items were marked by Rule B. This means that both X and Y also belong to L_{i-1}. Recall that Rule B skips at least g items in L_{i-1} between X and Y. Coupling this with the inductive hypothesis, we conclude that the distance between X and Y in iteration i is at least

$$g \cdot \frac{g^{i-2}}{2} = \frac{g^{i-1}}{2}. \qquad \square$$

THEOREM 3.3. *The competitive ratio of any randomized dynamic storage allocation algorithm is at least* $\Omega(\log \omega^* / \log\log \omega^*)$.

Proof. Consider the storage device after the insertion step of iteration i. By Lemmas 3.1 and 3.2, the occupied storage area is at least

$$\left(\frac{i \cdot 2^{r-1}}{(g+2)^{i-1}} - 2\right) \cdot \frac{g^{i-1}}{2} = \frac{i \cdot 2^{r-2}}{2 \cdot \left(1 + \frac{2}{g}\right)^{i-1}} - g^{i-1} \geq \frac{i \cdot 2^{r-2}}{2 \cdot \left(1 + \frac{2}{g}\right)^{i-1}} - 2^{r-1},$$

where the last transition follows from $i \leq \lfloor (r-1) \cdot \log_g 2 \rfloor + 1$. Hence, after the insertion step of the last iteration, the occupied storage area is at least

$$\frac{((r-1) \cdot \log_g 2) \cdot 2^{r-2}}{2 \cdot \left(1 + \frac{2}{g}\right)^{(r-1)\log_g 2}} - 2^{r-1}.$$

We choose g such that $g = \lfloor r / \log_2 r \rfloor$. This means that the occupied storage area is at least

$$\frac{(r-1) \cdot 2^{r-2} \cdot \frac{1}{\log_2 r - \log_2 \log_2 r}}{2 \cdot \left(1 + \frac{2}{\frac{r}{2\log_2 r}}\right)^{\frac{2r}{\log_2 r}}} - 2^{r-1} \geq \frac{\frac{r-1}{\log_2 r} \cdot 2^{r-2}}{2 \cdot e^8} - 2^{r-1} = \Omega\left(\frac{\omega^* \cdot \log \omega^*}{\log\log \omega^*}\right). \qquad \square$$

Since $\omega^* \geq W_{\max}$ and $\omega^* \geq k$, it turns out that $\Omega(\log W_{\max} / \log\log W_{\max})$ and $\Omega(\log k / \log\log k)$ are also lower bounds on the competitive ratio. Note that in the strategy of *Adv* the three values ω^*, $W_{\max}$, and k are equal up to a constant factor.

Acknowledgments. We thank the two anonymous referees for insightful comments. Many thanks to Oran Sharon for bringing this problem to our attention. We would like to thank Yishay Mansour and Ron Shamir for helpful discussions.

REFERENCES

[1] S. BEN-DAVID, A. BORODIN, R. M. KARP, G. TARDOS, AND A. WIGDERSON, *On the power of randomization in on-line algorithms*, Algorithmica, 11 (1994), pp. 2–14.

[2] R. P. BRENT, *Efficient implementation of the first-fit strategy for dynamic storage allocation*, ACM Trans. Programming Languages and Systems, 11 (1989), pp. 388–403.

[3] E. G. COFFMAN, *An introduction to combinatorial models of dynamic storage allocation*, SIAM Rev., 25 (1983), pp. 311–325.

[4] E. G. COFFMAN, M. R. GAREY, AND D. S. JOHNSON, *Dynamic bin packing*, SIAM J. Comput., 12 (1983), pp. 227-258.

[5] ———, *Approximation algorithms for bin-packing—an updated survey*, in Algorithm Design for Computer System Design, G. Ausiello, M. Lucertini, and P. Serafini, eds., Springer-Verlag, Wien and New York, 1984, pp. 49–106.

[6] D. DETLEFS, A. DOSSER, AND B. ZORN, *Memory Allocation Costs in Large C and C++ Programs*, Technical Report CU-CS-665-93, University of Colorado, Boulder, 1993.

[7] W. FELLER, *An Introduction to the Theory of Probability and Its Applications*, John Wiley and Sons, New York, 1967.

[8] A. FIAT, R. M. KARP, M. G. LUBY, L. A. MCGEOCH, D. D. SLEATOR, AND N. E. YOUNG, *Competitive paging algorithms*, J. Algorithms, 12 (1991), pp. 685–699.

[9] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability—A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[10] H. A. KIERSTEAD, *The linearity of first-fit coloring of interval graphs*, SIAM J. Discrete Math, 1 (1988), pp. 526–530.

[11] ———, *A polynomial time approximation algorithm for dynamic storage allocation*, Discrete Math., 88 (1991), pp. 231–237.

[12] D. E. KNUTH, *Fundamental Algorithms, Vol.* 1, 2nd ed., Section 2.5, Addison-Wesley, Reading, MA, 1973.

[13] M. G. LUBY, J. NAOR, AND A. ORDA, *Tight bounds for dynamic storage allocation*, in Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, 1994, pp. 724–732.

[14] M. S. MANASSE, L. A. MCGEOCH, AND D. D. SLEATOR, *Competitive algorithms for on-line problems*, in Proc. 20th Annual ACM Symposium on Theory of Computing, Chicago, IL, 1988, pp. 322–333.

[15] J. M. ROBSON, *An estimate of the store size necessary for dynamic storage allocation*, J. Assoc. Comput. Mach., 18 (1971), pp. 416–423.

[16] ———, *Bounds for some functions concerning dynamic storage allocation*, J. Assoc. Comput. Mach., 12 (1974), pp. 491–499.

[17] ———, *Worst case fragmentation of first fit and best fit storage allocation strategies*, Computer J., 20 (1977), pp. 242–244.

[18] T. A. STANDISH, *Data Structures Techniques*, Addison-Wesley, Reading, MA, 1980.

[19] D. D. SLEATOR AND R. E. TARJAN, *Amortized efficiency of list update and paging rules*, J. Assoc. Comput. Mach., 28 (1985), pp. 202–208.

[20] D. R. WOODALL, *The bay restaurant—a linear storage problem*, Amer. Math. Monthly, 81 (1974), pp. 240–246.

[21] C. B. WEINSTOCK AND W. A. WULF, *Quickfit: an efficient algorithm for heap storage allocation*, ACM SIGPLAN Notices, 23 (1988), pp. 141–144.

[22] A. C.-C. YAO, *Probabilistic computations: towards a unified measure of complexity*, in Proc. 18th Annual IEEE Symposium on Foundations of Computer Science, Providence, RI, 1977, pp. 222–227.

SIAM J. DISC. MATH.
Vol. 9, No. 2, pp. 167–172, May 1996

FINDING A DOMATIC PARTITION OF AN INTERVAL GRAPH IN TIME $O(n)$ *

GLENN K. MANACHER† AND TERRANCE A. MANKUS†

Abstract. We present a simple $O(n)$ time and space algorithm for producing a *domatic partition* and the *domatic number* for members of the class of interval graphs, where n is the number of intervals in a graph.

Key words. domination, domatic number, domatic partition, interval graph

AMS subject classifications. 68Q25, 68R10

1. Introduction. A graph $G = (V, E)$ is an *interval graph* if its vertices can be put in one-to-one correspondence with a family I of intervals on the real line such that two vertices are adjacent in G if and only if their corresponding intervals have nonempty intersection. I is called the *intersection model* for G. If the endpoints of I are presorted, I is called a *sorted model.*

A set S *dominates* V if and only if every vertex in $V - S$ is adjacent to at least one member of S. Let $DP = \{P | P$ is a partition of V and every set in P dominates $V\}$. The *domatic number* is defined as $DN(G) = \max\{\#P | P \in DP\}$, where # denotes cardinality. A *maximum domatic partition* $MDP(G)$ is a partition P' in DP for which $\#P' = DN(G)$.

Finding $DN(G)$ is NP-complete for arbitrary graphs [GJ79] and even for circular-arc graphs [Bo85]. For interval graphs, Bertossi [Be88] showed that, given the model, $DN(G)$ and $MDP(G)$ can be found in time $O(n^{2.5})$; for a *proper* interval graph, no two of whose intervals contain one another, this can be reduced to time $O(n)$ [LHC90].

After submission of our original draft, we discovered that our results were equaled in a contemporaneous proceedings paper by Peng and Chang [PC91]. This work is referenced accessibly and extended to strongly chordal graphs in [PC92]. Peng and Chang first found the $MDP(G)$ in time $O(n \log\log n)$. Their proof of correctness of this algorithm relied on two elegant, high-level invariants, which also apply to our algorithm. They then used an insightful resequencing of their set operations, using queues to represent sets, to reduce the time complexity to $O(n)$. Our approach avoids these complexities, using a less elegant but simpler linear-time method amenable to direct proof.

2. Preliminaries. In [CH77], lower and upper bounds on $DN(G)$ were studied for various classes of graphs G. Let d_i denote the degree of vertex V_i, and let $\delta(G) = \min\{d_i | V_i$ is a vertex in $G\}$. In [CH77] Cockayne and Hedetniemi established the easy result $DN(G) \leq \delta(G) + 1$ for any graph G. G is called *domatically full* if $DN(G) = \delta(G) + 1$. In [CH77] it was determined that every graph in certain classes of graphs, e.g., trees and maximal outerplanar graphs, are domatically full.

In an important paper apparently overlooked by most of these references, Farber [F84] showed nonconstructively that *strongly chordal graphs* (chordal or triangulated

* Received by the editors February 11, 1991; accepted for publication (in revised form) June 30, 1995.

† Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, IL 60607 (manacher@uis.edu).

graphs in which every even cycle of length ≥ 6 has an "odd" chord) are also domatically full. Since interval graphs are a subset of strongly chordal graphs [J85] it follows that interval graphs are domatically full. This result also is a simple consequence of our algorithm (Corollary 1). It is known that chordal graphs are not domatically full; an example was first discovered by Arikati [A90] with $n = 9$ and $e = 27$.

We consider the class of interval graphs G with an endpoint-sorted model as input. A simple scan can compute $\delta(G)$, and thereby $DN(G)$, in time $O(n)$, where n equals the number of intervals. In this paper, we show how $MDP(G)$ can also be computed in time $O(n)$.

3. Domatic number. Without loss of generality, suppose all endpoints in I are distinct. Let $I_1, \ldots, I_n$ represent the intervals labelled by the appearance of their left endpoints in a left-to-right scan. In the sequel we denote I_i simply as i. Let $LE(i), RE(i)$ denote, respectively, the left endpoint and right endpoint of I_i. For any pair of endpoints, α and β, we write $\alpha < \beta$ to indicate that α lies to the left of β.

It is clear that the vertex degree of interval i is given by the simple formula

$$d_i = a_i - b_i - 1,$$

where $a_i \equiv \#\{j | LE(j) < RE(i)\}$ and $b_i \equiv \#\{k | RE(k) < LE(i)\}$. It is a trivial matter to scan I, accumulating left and right endpoints in separate counts, and thereby to compute d_i for all i. Hence we have immediately Lemma 1.

LEMMA 1. *Given a sorted interval model I, d_i can be computed for all i in time $O(n)$.*

COROLLARY 1. *Given a sorted model of an interval graph, $\delta(G)$, and therefore $DN(G) = \delta(G) + 1$, can be computed in time $O(n)$.*

In the sequel, we will denote $\delta(G)$ simply by δ.

4. MDP algorithm. Our partitioning process is dependent upon the relative positions of the right endpoints of the intervals in the given model. To represent these, we set up an array $R[1 \ldots n]$ where $R[j] = i$ if the right endpoint of interval i is the jth right endpoint encountered during a left-to-right scan of I.

Our algorithm partitions the intervals $1 \ldots n$ into $\delta + 1$ partition sets, each of which is a dominating set. Each interval will be assigned a *partition designation number* in the range $1 \ldots \delta + 1$, with the number stored in an array PART$[1 \ldots n]$. If interval i is a member of the kth partition, we set PART$[i] = k$. If it is determined that interval i can be placed into any of the $\delta + 1$ partition sets, this fact will be reflected by assigning PART$[i] = \infty$.

The partitioning process has the following simple rules. We assume that $R[1 \ldots n]$ has already been determined and is available for use, along with the interval endpoints listed in a suitable form.

Step 1. We begin by initializing the $\delta + 1$ partition sets. For the intervals labelled 1 to $\delta + 1$, we set PART$[i] = i$. The interval last placed into each of the $\delta + 1$ partition sets will be referred to as a *last*. Initially, intervals 1 to $\delta + 1$ are *lasts*. Since there are exactly $\delta + 1$ partition sets, there are exactly $\delta + 1$ *lasts*. An array $L[1 \ldots n]$ will be used to keep track of the $\delta + 1$ *lasts*. If interval i is a *last*, then $L[i] = 1$ and interval i is in the partition set numbered PART$[i]$. If interval i is not a *last* then $L[i] = 0$.

The remaining intervals, $\delta + 2$ to n, will be considered for placement in increasing label order. **c** will be used to denote the label of the interval currently under consideration for placement into one of the partition sets.

Step 2. Consider the interval right endpoint array $R[1 \ldots n]$ and the *last* array $L[1 \ldots n]$. Let j in $1 \ldots n$ be the smallest number such that $L[R[j]] = 1$. Then interval

$R[j]$ is a *last* and it also has the leftmost right endpoint of all the *lasts*. We will refer to this *last* interval as *LREP__last*; in the sequel, let z represent *LREP__last*.

Since each of the $\delta + 1$ *lasts* already are elements in distinct partition sets, we see that for each interval j for which $L[j] = 1$, $j < \mathbf{c}$ since $\mathbf{c}$ is as yet unplaced. (Note that $j < \mathbf{c}$ means $LE(j) < LE(\mathbf{c})$.) Since z is *LREP__last*, it follows that if $j \neq z$, then $RE(z) < RE(j)$ for each such *last* interval j. This simple fact is the guiding idea behind the partition placement of $\mathbf{c}$.

Step 3. If $RE(z) < RE(\mathbf{c})$, we place interval $\mathbf{c}$ into the same partition as interval z, executing PART[$\mathbf{c}$] := PART[z]. Interval $\mathbf{c}$ becomes the new *last* interval for that partition; thus we execute $L[z] := 0$ and $L[\mathbf{c}] := 1$. Interval z is no longer *LREP__last*, so the new *LREP__last* must be found. We need only examine the right endpoints to the right of $RE(z)$ to find the new *LREP__last*. This is achieved by scanning array $R[1 \ldots n]$ beginning with $RE(z)$'s position. This fact makes the cumulative time required $O(n)$ for determining all *LREP__lasts*.

If $RE(\mathbf{c}) < RE(z)$, then (1) for all j such that $L[j] = 1, j < \mathbf{c}$, and (2) since z is *LREP__last*, $RE(z) < RE(j)$. Thus we have, for all *last* intervals j,

$$LE(j) < LE(\mathbf{c}) < RE(\mathbf{c}) < RE(j).$$

This shows that $\mathbf{c}$ is properly contained in all *lasts*. Thus $\mathbf{c}$ can be placed in any partition set since any intervals it dominates are already dominated by the *lasts*. This will be reflected by the assignment PART[$\mathbf{c}$] := ∞. In this case, interval z remains *LREP__last* since $\mathbf{c}$ was not placed into the partition PART[z].

Since both parts of Step 3 make a partitioning determination for $\mathbf{c}$, we need to find the next interval for partition placement. This is a simple matter.

Step 4. Repeat Step 3 until all n intervals are placed into partition sets.

We first present the algorithm in a high-level format and then follow with a more detailed presentation which provides the framework for both analysis and proof of correctness.

```
    ALGORITHM D__P
  INPUT: Model of the Set of Intervals 1...n and integer δ = δ(G).
  OUTPUT: Dominating Partition Sets PART(1)... PART(δ + 1).
/* Let RRi represent the rightmost right endpoint of
/* any interval in PART(i).
    for i := 1 to δ + 1 do
      Place Interval i into PART(i) od
    for c := δ + 2 to n do
      Find j such that RRj is leftmost of all RRk's for k in 1...δ + 1:
      If right endpoint of Interval c extends to the right of RRj
        Then place Interval c in PART(j)
      Else
        Interval c is already dominated by all partition sets and
        may be placed in any of the partition sets.
    od
  end D__P
```

The following detailed algorithm can easily be seen to be a refinement of the above and also to perform the instructions in Steps 1–4.

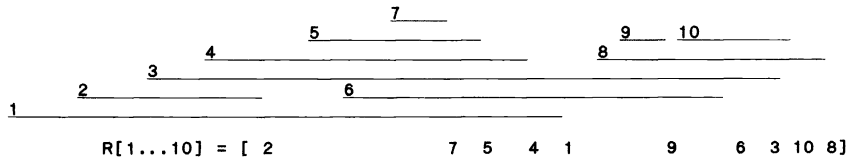


FIG. 1.

ALGORITHM *DOMATIC_PARTITION*
INPUT: Array $R[1 \ldots n]$ and integer $\delta = \delta(G)$.
OUTPUT: Array PART$[1 \ldots n]$.

```
  for i := 1 to n do                /* initialize array L
    L[i] := 0 od
  for i := 1 to δ + 1 do
    PART[i] := i;                   /* interval i into partition set #i
    L[i] := 1 od                    /* interval i is now a last
  j := 1;                           /* initialize scan variable
  for c := δ + 2 to n do            /* if n = δ + 1, then stop
                                    /* else place remaining intervals
    while (L[R[j]] ≠ 1) do          /* find LREP_last
    j := j + 1 od
    z := R[j];                      /* z is an interval label holder
    if (RE(z) < RE(c)) then         /* if c extends past z
      PART[c] := PART[z];           /* then assign to c the
                                    /* partition number of z.
      L[z] := 0;                    /* z is no longer LREP_last
      L[c] := 1                     /* c becomes a last
    else
      PART[c] := ∞                  /* c may be in any partition
    fi
  od
end DOMATIC_PARTITION
```

Example. For the interval model of Fig. 1, the interested reader will find that $n = 10$ and $\delta = 3$. The right endpoints appear in the order $R[1 \ldots 10] = [2 \quad 7 \quad 5 \quad 4 \quad 1 \quad 9 \quad 6 \quad 3 \quad 10 \quad 8]$. The operation of *DOMATIC_PARTITION* on this model results in the partition array PART$[1 \ldots 10] = [1 \quad 2 \quad 3 \quad 4 \quad 2 \quad 2 \quad \infty \quad 4 \quad 1 \quad 1]$.

THEOREM. *Algorithm DOMATIC_PARTITION produces a domatic partition for the interval model in $O(n)$ time.*

Proof. We use an alternative definition of a domatic partition in this proof. According to this definition, we need only show that each interval in $1 \ldots n$ satisfies the requirement that it intersect at least one member of each of the δ partition sets of which it is not a member. The equivalence of the two definitions is elementary and the proof omitted.

The intervals $1 \ldots \delta + 1$ clearly satisfy this requirement. Suppose to the contrary that for some pair of intervals i and j, where $i < j \leq \delta + 1$, interval i did not intersect interval j; then $RE(i) < LE(j)$, and interval i could only intersect the intervals 1 to $j - 1$. Thus $\deg(i) \leq j - 2 \leq \delta + 1 - 2 = \delta - 1$. This is a contradiction since each interval has degree δ or greater. Hence each interval $i \in 1 \ldots \delta + 1$ intersects a member of each of the $\delta + 1$ partitions.

For all $i \in 1 \ldots n$, let $\mathbf{N}(i) \equiv \{i\} \cup \{j | j \neq i$ and interval j intersects interval $i\}$. $\mathbf{N}(i)$ is the reflexive neighbor set of interval i.

Upon completion of the partitioning algorithm, for each interval $i \in \delta + 2 \ldots n$, there are two possibilities described as follows.

Case 1. *For some* $j \in \mathbf{N}(i)$, PART$[j] = \infty$. By the explanation of Step 3, during the stage of execution of the loop when $\mathbf{c}$ was j, $\mathbf{c}$ was properly contained in each *last* and therefore was (and remains) properly contained in a member of each partition set. Thus interval i intersects the same partition members in which interval j was contained and therefore satisfies the domatic partition requirement.

Case 2. *For every* $j \in \mathbf{N}(i)$, PART$[j] \neq \infty$. Consider two subcases.

Case 2a. *For every distinct pair of intervals* j, k *in* $\mathbf{N}(i)$, PART$[j] \neq$ PART$[k]$. Since $\#(\mathbf{N}(i)) \geq \delta + 1$, it follows that $\#(\mathbf{N}(i)) = \delta + 1$. Hence interval i trivially intersects a member of each partition set.

Case 2b. *For some distinct pair of intervals* $j, k \in \mathbf{N}(i)$, *with* $j < k$, PART$[j] =$ PART$[k]$. Since PART$[j] \neq \infty$, PART$[j] =$ PART$[k]$, and $j < k$, it follows that during some stage of the execution of the main algorithm loop, interval j was *LREP__last*, and during this stage, say $\mathbf{c}^*$, either $\mathbf{c}^*$ was k or was less than k. During stage $\mathbf{c}^*$, for all intervals s (including j) such that $L[s] = 1$, we have the following two facts:

(f1) $RE(j) < RE(s)$, since interval j is *LREP__last*.

(f2) $s < k$, since $s < \mathbf{c}^*$ and $\mathbf{c}^* \leq k$, implying $LE(s) < LE(k)$.

In addition, we know that

(f3) $LE(k) < RE(i)$ and $LE(i) < RE(j)$, since k and j both intersect i.

Combining (f1), (f2), and (f3), we see that for all *lasts* s during stage $\mathbf{c}^*$,

$$LE(s) < RE(i) \qquad \text{and} \qquad LE(i) < RE(s).$$

This implies that interval i intersects each of the $\delta + 1$ *lasts* existing during this stage. Hence interval i intersects a member of each of the $\delta + 1$ partition sets.

The algorithm clearly takes $O(n)$ time to complete. During each execution of the **for c** loop, the **while** loop either advances the pointer j or is terminated. Each of the other steps is clearly $O(1)$ in time complexity. Thus the cumulative time complexity for our algorithm, given the model, is $O(n)$ time and space. □

5. Open problem. An algorithm for MDP simpler than Algorithm Domatic__Partition or that of [PC91] is presented below. No proof for it of comparable simplicity is known to us.

```
PART [0...n] := 0;
for i := 1 to δ + 1 do
  PART[i] := i od
j := 1;
for c := δ + 2 to n do
  while (j <= n) and (PART[R[j]] = 0) do
    PART[R[j]] := ∞;
    j := j + 1 od
  if (PART[c] = 0) then
    PART[c] := PART[R[j]];
    j := j + 1 fi
od
```

We leave as an open problem whether such a proof exists.

Acknowledgments. We wish to thank M. S. Chang for keeping us abreast of his work from 1993 onward and to acknowledge the anonymous referee who sketched the high-level algorithm and suggested its inclusion.

REFERENCES

[A90] S. R. ARIKATI, private communication, December 1990.

[Be88] A. A. BERTOSSI, *On the domatic number of interval graphs*, Inform. Process. Lett., 28 (1988), pp. 275–280.

[Bo85] M. A. BONUCCELLI, *Dominating sets and domatic number of circular arc graphs*, Discrete Appl. Math., 12 (1985), pp. 203–213.

[CH77] E. J. COCKAYNE AND S. T. HEDETNIEMI, *Toward a theory of domination in graphs*, Networks, 7 (1977), pp. 247–261.

[F84] M. FARBER, *Domination, independent domination, and duality in strongly chordal graphs*, Discrete Appl. Math., 7 (1984), pp. 115–130.

[GJ79] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability, a Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[J85] D. S. JOHNSON, *The NP-completeness column: An ongoing guide*, J. Algorithms, 6 (1985), pp. 434–451.

[LHC90] T.-L. LU, P.-H. HO, AND G. CHANG, *The domatic number problem in interval graphs*, SIAM J. Discrete Math., 3 (1990), pp. 531–536.

[PC91] S. L. PENG AND M. S. CHANG, *A new approach for domatic number problem on interval graphs*, in Proceedings of the National Computer Symposium, 1991, **R.O.C.**, pp. 236–241.

[PC92] ———, *A simple linear time algorithm for the domatic partition problem on strongly chordal graphs*, Inform. Process. Lett., 43 (1992), pp. 297–300.

[RR89] S. R. ARIKATI AND C. PANDU RANGAN, *Linear algorithms for domatic number problems on interval graphs*, Inform. Process. Lett., 33 (1989/90), pp. 29–33.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 173–177, May 1996

002

ON A QUESTION OF ERDÖS ON SUBSEQUENCE SUMS*

DON COPPERSMITH† AND STEVEN PHILLIPS‡

Abstract. Paul Erdös asked how dense a sequence of integers, none of which is the sum of a consecutive subsequence, can be. In other words, let $\langle x_1, \ldots, x_m \rangle$ be an increasing sequence of integers in $[1, n]$, such that there do not exist i, j, and k, with $0 < i < j < k \leq m$ and $x_i + x_{i+1} + \cdots + x_j = x_k$. Erdös asked if $m > n/2 + 1$ is possible. A simple argument shows that $m > 2n/3 + O(\log n)$ is impossible. Freud recently constructed a sequence with $m = 19n/36$. This note constructs a sequence with $m = 13n/24 - O(1)$ and extends the simple upper bound to show that $m > (2/3 - \epsilon)n + (\log n)$ is impossible for $\epsilon = 1/512$.

Key words. sequence, integers, density, external problem

AMS subject classifications. 05D05, 11B05

1. Preliminaries.

When describing subintervals of $[1, n]$, we will use "1" to designate an element x_i of the sequence, "0" to designate an integer that is not in the sequence, and "-" for an integer which may or may not be in the sequence. So for example, "$[y, y+4] = 01100$" means that for some i we have $x_{i-1} < y$, $x_i = y + 1$, $x_{i+1} = y + 2$, $x_{i+2} > y + 4$. Property S_k says that the sum of k adjacent elements is not an element. For a nonnegative integer i, layer i is all integers in the interval $(n/2^{i+1}, n/2^i]$ (so the size of layer i is $\lfloor n/2^i \rfloor - \lfloor n/2^{i+1} \rfloor$). A 0 is *forced* if it is the sum of an adjacent pair of elements, and *unforced* otherwise. A string is forced if all its 0's are forced.

A simple lower bound of $n/2$ is obtained by the sequence consisting of layer 0. Erdös [2] asked if $m > n/2+1$ is possible. Both the upper and lower bounds described below have the following simple upper bound as their starting point.

LEMMA 1.1. *A sequence of integers in* $[1, n]$ *satisfying* S_2 *contains at most* $2n/3 + 3/2(\log_4 n + 1)$ *elements.*

Proof. For a nonnegative integer i, each adjacent pair of elements in layer $2i + 1$ eliminates (by property S_2) a distinct element in layer $2i$. So layers $2i$ and $2i + 1$ together contain at most $1 + \lfloor n/2^{2i} \rfloor - \lfloor n/2^{2i+1} \rfloor \leq 3/2 + n/2^{2i+1}$ elements. Thus the number of elements in $[1, n]$ is at most $\sum_{i=0}^{\lfloor \log_4 n \rfloor} (3/2 + n/2^{2i+1}) \leq 2n/3 + 3/2(\log_4 n + 1)$. □

Note that the constant 2/3 in Lemma 1.1 is tight if the sequence must only satisfy S_i for even i, as evidenced by the sequence which excludes integers $\equiv 0 \pmod 3$.

The rest of this note is organized as follows: §2 presents the construction of $m = 13n/24 - O(1)$, while §3 applies property S_3 to improve the upper bound of Lemma 1.1 to $(2/3 - \epsilon)n + O(\log n)$.

2. Lower bound.

In this section we will construct a lower bound of $13n/24 - O(1)$. The lower bound has $n/2$ elements in the first two layers. It adds "extra" elements between 1/5 and 1/4 in such a way that consecutive sequences of 2, 3, or 4 of the extra elements add up to elements excluded from the first two layers. The

* Received by the editors February 8, 1993; accepted for publication (in revised form) April 5, 1995.

† IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

‡ Computer Science Department, Stanford University, Stanford, CA 94305. A portion of this research was performed while the author was visiting the IBM T. J. Watson Research Center. The research of this author was partially supported by National Science Foundation grant CCR-9010517 and an Office of Technology Licensing grant.

TABLE 1
The sequence proving the lower bound.

From	To	Keep	Mod	Value		
1/5	2/9	0,1,2,5,6,8,10,11,14,15	16	$10/16 \times 1/45$	=	1/72
2/9	1/4	1	1	$1 \times 1/36$	=	1/36
1/4	8/27	None		0	=	0
8/27	1/3	1,2,3	4	$3/4 \times 1/27$	=	1/36
1/3	3/8	1,2	3	$2/3 \times 1/24$	=	1/36
3/8	4/9	0,2,6,8,9,12,13, 16,19,20,23,24,26,30	32	$14/32 \times 5/72$	=	35/1152
4/9	1/2	0	2	$1/2 \times 1/18$	=	1/36
1/2	16/27	1	1	$1 \times 5/54$	=	5/54
16/27	2/3	1,2,4,6,7	8	$5/8 \times 2/27$	=	5/108
2/3	3/4	1,2	3	$2/3 \times 1/12$	=	1/18
3/4	8/9	All Except 2,8,14,17,21, 25,29,35,39,43,47,50,56,62	64	$50/64 \times 5/36$	=	125/1152
8/9	1	0,1,3	4	$3/4 \times 1/9$	=	1/12

density of elements achieved is $0.541666\ldots$. The idea of the lower bound arose from patterns observed in sequences found experimentally by J. H. Davenport (personal correspondence).

THEOREM 2.1. *For any n there is a sequence of $13n/24 - O(1)$ integers in $[1, n]$, none of which is the sum of a consecutive subsequence.*

Proof. We begin with Freud's [1] construction of $m \geq 19n/36$, which will motivate our construction of $m \geq 13n/24 - O(1)$. To achieve $m = 19n/36$, start by putting in all integers between $2n/9$ and $n/4$. This means we must exclude odd integers between $4n/9$ and $n/2$, integers $\equiv 0 \pmod 3$ between $2n/3$ and $3n/4$, and integers $\equiv 2 \pmod 4$ between $8n/9$ and n. Now fortunately the nonexcluded integers between $4n/9$ and $n/2$ and between $8n/9$ and n mesh together perfectly: two adjacent even numbers add to something $\equiv 2 \pmod 4$. So, in keeping with the restriction that there be $n/2$ elements in the first two layers, we must keep exactly the even integers between $4n/9$ and $n/2$ and those $\equiv 0, 1, 3 \pmod 4$ between $8n/9$ and n.

To make the integers $\equiv 0 \pmod 3$ that have been knocked out between $2n/3$ and $3n/4$ also be excluded by sums from layer 1, we introduce integers between $n/3$ and $3n/8$, and it turns out that allowing those $\equiv 1, 2 \pmod 3$ does the trick. Then the ranges $[3n/4, 8n/9]$ and $[3n/8, 4n/9]$ can be filled in by any complementary manner.

To extend this construction to $m = 13n/24$, we add some integers between $n/5$ and $2n/9$. This complicates the construction since we have to add integers below $n/3$ and triples of these integers now have to be considered. Specifically, we add some integers between $8n/27$ and $n/3$, chosen so that triples knock out already excluded integers between $8n/9$ and n. The integers between $n/5$ and $2n/9$ are chosen so that triples knock out only integers in $[3n/5, 2n/3]$ that have already been knocked out by pairs of integers between $[3n/10, n/3]$.

The details are as follows: start with the "keep" integers from Table 1 (for instance, the fourth line says keep all integers between $8n/27$ and $n/3$ that are congruent to $1, 2,$ or $3 \pmod 4$, for a total of $n/36$ elements). The number of kept elements is $13n/24$. Sequences within an interval sum to nonelements, but we might have a problem at the interval boundaries. A sequence spanning interval boundaries might sum to an element. If this occurs, then eliminate that sum (the larger element) from the sequence. This in turn affects sequences spanning the space where that eliminated element was. Only a constant number of elements get eliminated because you're always pushing forward, never back. □

3. Upper bound. The tighter upper bound applies property S_3 to elements in the interval $[3n/16, n/4]$ to show that there must be at least ϵn unforced 0's (0's that aren't accounted for in Lemma 1.1.) The effect of these unforced 0's on the sequence size is shown by the following lemma.

LEMMA 3.1. *A sequence of integers in* $[1, n]$ *satisfying* S_2 *and having* k *unforced* 0*'s in even layers contains at most* $2n/3 - k + 3/2(\log_4 n + 1)$ *elements.*

Proof. For a nonnegative integer i, each adjacent pair of elements in layer $2i + 1$ eliminates (by property S_2) a distinct element in layer $2i$. Therefore layers $2i$ and $2i + 1$ together contain at most $1 + \lfloor n/2^{2i} \rfloor - \lfloor n/2^{2i+1} \rfloor - k_i \leq 3/2 + n/2^{2i+1} - k_i$ elements, where k_i is the number of unforced 0's in layer $2i$. Thus, the number of elements in $[1, n]$ is at most $\sum_{i=0}^{\lfloor \log_4 n \rfloor}(3/2 + n/2^{2i+1} - k_i) \leq 2n/3 - k + 3/2(\log_4 n + 1)$. □

The rest of this section is organized as follows. Lemmas 3.2 and 3.3 prove the existence of a collection of unforced 0's and forced 010's in the first 4 layers. Lemmas 3.4, 3.5, and 3.6 show how the forced 010's can be traced down through the rest of the layers, spawning a collection of unforced 0's in even layers. Finally, Theorem 3.7 assembles all the lemmas to produce the upper bound.

LEMMA 3.2. 1. 00 *cannot be forced.*

2. $[2y, 2y + 2] = 010$ *cannot be forced.*

3. $[2y + 1, 2y + 3] = 010$ *is forced only if* $[y, y + 2] = 111$.

Proof. For part 1, two consecutive forced 0's are $x_i + x_{i+1}$ and $x_{i+1} + x_{i+2}$, which differ by $x_{i+2} - x_i$, which is at least 2. For part 2, if both zeroes are forced we must have $2y = x_i + x_{i+1}$, $2y + 2 = x_{i+1} + x_{i+2}$, $x_{i+2} - x_i = 2$, so $x_{i+1} = x_i + 1$ and $2y = 2x_i + 1$, which is a contradiction. Part 3 is similar. □

LEMMA 3.3. *Let* $y \in [3n/64 + 2, n/16 - 3]$. *Then we have either a* 010 *centered in* $[y - 1, y + 2]$ *or* $[3y, 3y + 3]$, *or an unforced* 0 *in* $[y - 2, y + 3]$, $[3y - 2, 3y + 5]$, $[4y - 1, 4y + 5]$, *or* $[12y + 2, 12y + 10]$.

Proof. Assume there is no 010 centered in $[y-1, y+2]$ or unforced 0 in $[y-2, y+3]$. Then the string $[y - 2, y + 3]$ matches one of the following, shown in Table 2 (there is no 00 by Lemma 3.2).

TABLE 2

-2	-1	0	1	2	3	case
-	1	1	1	1	-	A
1	0	1	1	1	-	B
1	1	0	1	1	-	C
-	1	1	1	0	1	D
1	0	1	1	0	1	E
-	1	1	0	1	1	F

Assume also that there is neither a 010 centered in $[3y, 3y + 3]$ nor an unforced 0 in $[3y - 2, 3y + 5]$ or $[4y - 1, 4y + 5]$.

Case A. $[y-1, y+2] = 1111$. We have 0 at $3y$, $3y+3$, $4y+2$. Therefore $[3y-1, 3y+4]$ $= 101101$ and $[4y + 1, 4y + 4] = 1011$ (since 00 is outlawed by Lemma 3.2 part 1 and $[4y + 1, 4y + 4] = 1010$ is outlawed by Lemma 3.2 part 2). These cause 0's at $12y + 6$, $12y + 8$, respectively, one of which must be an unforced 0.

Case B. $[y - 2, y + 2] = 10111$. We have 0 at $3y - 1$, $3y + 3$, $4y + 1$. Therefore $[3y - 1, 3y + 4] = 011101$ and $[4y, 4y + 3] = 1011$ (since by Lemma 3.2 part 3, forced 1010 would imply $[2y, 2y + 2] = 111$, which is ruled out by S_2 since $[y, y + 1] = 11$). These cause 0's at $12y + 7$, $12y + 5$, respectively, one of which must be an unforced 0.

Case C. $[y-2, y+2] = 11011$. We have 0 at $3y-2$, $3y+2$, $4y$. Therefore $[3y-2, 3y+3] = 011101$ and $[4y-1, 4y+2] = 1011$. These cause 0's at $12y+3$, $12y+2$, respectively, one of which must be an unforced 0.

Case D. $[y-1, y+3] = 11101$. We have 0 at $3y$, $3y+4$, $4y+3$. Therefore $[3y-1, 3y+4] = 101110$ and $[4y+1, 4y+4] = 1101$. These cause 0's at $12y+5$, $12y+7$, respectively, one of which must be an unforced 0.

Case E. $[y-2, y+3] = 101101$. We have 0 at $3y-1$, $3y+4$, $4y+2$. Therefore $[3y-1, 3y+4] = 011110$ and $[4y+1, 4y+4] = 1011$. These cause 0's at $12y+6$, $12y+8$, respectively, one of which must be an unforced 0.

Case F. $[y-1, y+3] = 11011$. We have 0 at $3y+1$, $3y+5$, $4y+4$. Therefore $[3y, 3y+5] = 101110$ and $[4y+2, 4y+5] = 1101$. These cause 0's at $12y+9$, $12y+10$, respectively, one of which must be an unforced 0. □

LEMMA 3.4. *A forced* 010 *centered at* y *implies a forced* 010 *centered at* $y/4$ (*implying* y *is a multiple of* 4) *or an unforced* 0 *in the interval* $[\lfloor y/4 \rfloor - 1, \lceil y/4 \rceil + 1]$.

Proof. This is easily checked for $y < 8$. If $y \geq 8$, let $x = \lfloor y/4 \rfloor$.

Case 1. $y = 4x$. A forced 010 centered at y implies $[2x-1, 2x+1] = 111$. By property S_2, $[x-1, x+1]$ is one of 010, -00, or 00-. The string 00 cannot be forced.

Case 2. $y = 4x+2$. A forced 010 centered at y implies $[2x, 2x+2] = 111$. By property S_2, $[x-1, x+2]$ is one of -00-, 001-, or -100.

Case 3. y is odd. This is impossible since a forced 010 must be centered at an even integer. □

LEMMA 3.5. *If there are* k *forced* 010*'s in layer* $2i$, *then there is an* m *such that there are* m *forced* 010*'s and at least* $\max\{(k-m-6)/4, 0\}$ *unforced* 0*'s in layer* $2i+2$.

Proof. Let the k forced 010's in layer $2i$ be centered at $y_1, \ldots, y_k$. Call an 010 centered at y_j *descending* if there is a forced 010 centered at $y_j/4$. Let m be the number of descending 010's. For each nondescending 010 centered at y_j, choose an unforced 0 closest to $y_j/4$: by Lemma 3.4 there is an unforced 0 in the range $[\lfloor y_j/4 \rfloor - 1, \lceil y_j/4 \rceil + 1]$. How many forced 010's can choose a given unforced 0 at x? The unforced 0 at x can be chosen by at most 7 forced 010's, centered at $4x-6$, $4x-4$, $4x-2$, $4x$, $4x+2$, $4x+4$, $4x+6$ (since a forced 010 must be centered at an even number).

However, the 7 can be reduced to a 4 as follows. Consider the subinterval I between the nearest 1's on either side of x (restricted to the layer that x is in). Let $x-\alpha$ be the leftmost unforced 0 in this subinterval and $x+\beta$ the rightmost unforced 0. There are at least $\lceil \beta/2 \rceil + \lceil \alpha/2 \rceil + 1$ unforced 0's in $[x-\alpha, x+\beta]$ and either a layer boundary or a 1 in $[x-\alpha-2, x-\alpha-1]$ and likewise in $[x+\beta+1, x+\beta+2]$.

Now, assuming no layer boundary, the forced 010's associated with these unforced 0's (in I) are centered between $4(x-\alpha)-6$ and $4(x+\beta)+6$ inclusive. There are at most $2\beta+2\alpha+7$ such forced 010's centered at even integers. But the 1's near $x-\alpha$ and $x+\beta$ cause a 0 in $[2x-\alpha+\beta-1, 2x-\alpha+\beta+1]$, which eliminates three forced 010's near $4x-2\alpha+2\beta$. So the number of forced 010's choosing unforced 0's in the subinterval I is really at most $2\beta+2\alpha+4$, which is at most 4 times the number of unforced 0's in I.

Hence, in result, the number of nondescending 010's is at most 4 times the number of unforced 0's. However, the above argument breaks down at the ends of the layer, where at most 3 more nondescending 010's at each end can be associated with unforced 0's. □

LEMMA 3.6. *If there are* k *forced* 010*'s in layer* $2i$ $(k > 0)$, *then there are at least* $k/4 - (3/2)\log_4 n + 3i/2$ *unforced* 0*'s in even layers in* $[1, n/2^{2i+2}]$.

Proof. There are no 010's in layer $2\lfloor \log_4 n \rfloor$. Hence the k forced 010's have been converted to unforced 0's according to Lemma 3.5. The number of times Lemma 3.5 is invoked is at most $\lfloor \log_4 n/2^{2i} \rfloor \leq \log_4 n - i$. □

THEOREM 3.7. *A sequence of integers in* $[1, n]$ *satisfying* S_2, S_3, *and* S_4 *contains at most* $2n/3 - \lfloor n/512 \rfloor + 3\log_4 n - 1/2$ *elements.*

Proof. Let y be an integer, with $3n/64 + 2 \leq y \leq n/16 - 3$. By Lemma 3.3 and an application of Lemma 3.4, there is either a forced 010 centered in $[y-1, y+2]$ or at $\lceil 3y/4 \rceil$ (hence in layer 4), or an unforced 0 in $[y-2, y+3]$, $[3y-2, 3y+5]$, $[4y-1, 4y+5]$, or $[12y+2, 12y+10]$. At most 6 values of y can share any unforced 0, and at most 2 values of y can share any forced 010 (since a forced 010 must be centered at an even integer). The number of values for y is $\lfloor n/64 \rfloor - 5$. Thus for some $m \geq 0$, there are m forced 010's in layer 4, and at least $(\lfloor n/64 \rfloor - 2m - 5)/6$ unforced 0's in levels 0, 2, and 4. Applying Lemma 3.6 gives a collection of $m/4 - (3/2)\log_4 n + 3$ unforced 0's in levels $6, 8, \ldots$, for a total of at least $\lfloor n/512 \rfloor - (3/2)\log_4 n + 2$ unforced 0's in even levels (minimized when $m = \lfloor n/128 \rfloor - 2$). Finally, applying Lemma 3.1 gives the desired result. □

4. Conclusion. Let $m(n)$ denote the length of the longest sequence of integers in $[1, n]$, none of which is the sum of a consecutive subsequence. This note has shown that both the trivial lower bound of $m(n)/n \geq 1/2$ and the trivial upper bound of $m(n)/n \leq 2/3 + o(1)$ fail to be tight. It is also likely that neither of the bounds obtained here is tight. We close with the natural open question.

OPEN QUESTION 1. *What is the value of* $\liminf m(n)/n$? *Alternatively, what is the largest constant* α *such that for any* n, *there is a sequence of* $\alpha n - O(n)$ *integers in* $[1, n]$, *none of which is the sum of a consecutive subsequence*?

REFERENCES

[1] R. FREUD, *Adding numbers*, James Cook Mathematical Notes, 6 (1993), pp. 6199–6202.
[2] P. ERDÖS, *Problem* 91:02, in Western Number Theory Problems, R. Guy, ED., presented December 19, 22, 1992.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 178–200, May 1996

SPANNING TREES—SHORT OR SMALL*

R. RAVI†, R. SUNDARAM‡, M. V. MARATHE§, D. J. ROSENKRANTZ¶, AND S. S. RAVI‖

Abstract. We study the problem of finding small trees. Classical network design problems are considered with the additional constraint that only a specified number k of nodes are required to be connected in the solution. A prototypical example is the kMST problem in which we require a tree of minimum weight spanning at least k nodes in an edge-weighted graph. We show that the kMST problem is NP-hard even for points in the Euclidean plane. We provide approximation algorithms with performance ratio $2\sqrt{k}$ for the general edge-weighted case and $O(k^{1/4})$ for the case of points in the plane. Polynomial-time exact solutions are also presented for the class of treewidth-bounded graphs, which includes trees, series-parallel graphs, and bounded bandwidth graphs, and for points on the boundary of a convex region in the Euclidean plane.

We also investigate the problem of finding short trees and, more generally, that of finding networks with minimum diameter. A simple technique is used to provide a polynomial-time solution for finding k-trees of minimum diameter. We identify easy and hard problems arising in finding short networks using a framework due to T. C. Hu.

Key words. approximation algorithm, network design, spanning tree

AMS subject classifications. 05C05, 68Q25, 68R10

1. Introduction.

1.1. Motivation: Small trees. The oil reconnaissance boats are back from their final trip off the coast of Norway and present you with a detailed map of the seas surrounding the coastline. Marked in this map are locations that are believed to have a good chance of containing oil under the sea bed. Your company has a limited number of oil rigs that it is willing to invest in the effort. Your problem is to position these oil rigs at marked places so that the cost of laying down pipelines between these rigs is minimized. The problem at hand can be modeled as follows: given a graph with nonnegative edge weights and a specified number k, find a tree of minimum weight spanning at least k nodes. Note that a solution to the problem will be a tree spanning exactly k nodes. We call this problem the *k-minimum spanning tree* (or the kMST) problem. Moreover, the kMST problem is at the heart of several other optimization

* Received by the editors April 5, 1994; accepted for publication (in revised form) April 5, 1995. A preliminary version of this paper [31] appeared in *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994, pp. 546–555.

† Center for Discrete Mathematics and Theoretical Computer Science, Department of Computer Science, Princeton University, 35 Olden Street, Princeton, NJ 08854 (ravi@cs.princeton.edu). The research of this author was performed at Brown University and supported by an IBM Graduate Fellowship, National Science Foundation Presidential Young Investigator award CCR-9157620, and Defense Advanced Research Projects Agency contract N00014-91-J-4052/order 8225.

‡ Department of Computer Science, Massachussetts Institue of Technology, Cambridge MA 02139 (koods@theory.lcs.mit.edu). The research of this author was supported by Defense Advanced Research Projects Agency contract N0014-92-J-1799 and National Science Foundation grant 92-12184 CCR.

§ Los Alamos National Laboratory, P.O. Box 1663, MS M986, Los Alamos, NM 87545 (madhav@c3.lanl.gov). The research of this author was performed at the Department of Computer Science, State University of New York at Albany and supported by National Science Foundation grant CCR 89-03319.

¶ Department of Computer Science, State University of New York at Albany, Albany, NY 12222 (djr@cs.albany.edu). The research of this author was supported by National Science Foundation grant CCR-90-06396.

‖ Department of Computer Science, State University of New York at Albany, Albany, NY 12222 (ravi@cs.albany.edu). The research of this author was supported by National Science Foundation grant CCR-89-05296.

problems, such as the latency problem [9] and the prize-collecting traveling salesperson problem [1], and hence is of independent interest. In this paper, we study such classical network-design problems as the MST problem with the additional constraint that only a specified number of nodes need to be incorporated into the network. Unlike the MST problem, which admits a polynomial-time solution [25], [28], the kMST problem is considerably harder to solve. In Theorem 2.1 we prove that the kMST problem is NP-complete. This result was independently obtained by Lozovanu and Zelikovsky [26]. The kMST problem remains NP-complete even when all the edge weights are drawn from the set $\{1, 2, 3\}$ (i.e., the graph is complete and every edge takes one of three different weights). It is not hard to show a polynomial-time solution for the case of two distinct weights. The problem remains NP-hard even for the class of planar graphs as well as for points in the plane.

1.2. Approximation algorithms. A ρ-approximation algorithm for a minimization problem is one that delivers a solution of value at most ρ times the minimum. Consider a generalization of the kMST problem, the k-Steiner tree problem: given an edge-weighted graph, an integer k, and a subset of at least k vertices specified as terminals, find a minimum-weight tree spanning at least k terminals. We can apply approximation results for the kMST problem to this problem by considering the auxiliary complete graph on the terminals with edges weighted by shortest-path distances. A ρ-approximation for the kMST problem on the auxiliary graph yields a 2ρ-approximation for the k-Steiner tree problem. Therefore we focus on approximations for the kMST problem. We provide the first approximation algorithm for this problem. In Theorem 3.1 we present a polynomial-time algorithm $2\sqrt{k}$-approximation algorithm for the kMST problem. The algorithm is based on a combination of a greedy technique that constructs trees using edges of small cost and a shortest-path heuristic that merges trees when the number of trees to be merged is small. The analysis of the performance ratio is based on a solution–decomposition technique [4], [14], [24], [29], [30] that uses the structure of an optimal solution to derive a bound on the cost of the solution constructed by the approximation algorithm.

Theorem 3.1 provides a $4\sqrt{k}$-approximation algorithm for the k-Steiner tree problem as well. Moreover, we construct an example that demonstrates the performance guarantee of the approximation algorithm is tight to within a constant factor.

We derive a better approximation algorithm for the case of points in the Euclidean plane. In Theorem 4.1 we show that there is a polynomial-time algorithm that, given n points in the Euclidean plane and a positive integer $k \leq n$, constructs a tree spanning at least k of these points such that the total length of the tree is at most $O(k^{1/4})$ times that of a minimum-length tree spanning any k of the points.

As before, we construct an example showing that the performance ratio of the algorithm in Theorem 4.1 is tight. Our proof of Theorem 4.1 also yields as a corollary an approximation algorithm for the rectilinear kMST problem.

1.3. Polynomially solvable special cases. Since the kMST problem is NP-complete even for the class of planar graphs, we focus on special classes of graphs and provide exact solutions that run in polynomial time. Robertson and Seymour in their seminal series of papers [32] introduced and developed the notion of treewidth. Many hard problems have exact solutions when attention is restricted to the class of treewidth-bounded graphs and much work has been done in this area, especially by Bodlaender [11]. Independently, Bern, Lawler, and Wong [8] introduced the notion of decomposable graphs. Later, it was shown [5] that the class of decomposable graphs and the class of treewidth-bounded graphs are one and the same. A class

of decomposable graphs is defined using a finite number of primitive graphs and a finite collection of binary composition rules. Examples of decomposable graphs include trees, series-parallel graphs, and bounded-bandwidth graphs. We use a dynamic programming technique to show that for any class of decomposable graphs (or treewidth-bounded graphs), there is an $O(nk^2)$-time algorithm for solving the kMST problem. A polynomial-time algorithm for trees was also independently obtained by Lozovanu and Zelikovsky [26].

Though the kMST problem is hard for arbitrary configurations of points in the plane, we show in §5.2 that there is a polynomial-time algorithm for solving the kMST problem for the case of points in the Euclidean plane that lie on the boundary of a convex region. We also provide a faster algorithm to find the optimal kMST when all the points lie on a circle. The proof of the above facts uses a monotonicity property of an optimal tree along with a degree constraint on an optimal solution. This allows us to apply dynamic programming to find the exact solution. Several researchers in computational geometry have presented exact algorithms for choosing k points that minimize other objectives such as diameter, perimeter, area, and volume [3], [16]–[18].

1.4. Short trees. Keeping the longest path in a network small is often an important consideration in network design. We investigate the problem of finding networks with small diameter. Recall that the diameter of a tree is the maximum distance (path length) between any pair of nodes in the tree. The problem of finding a minimum-diameter spanning tree of an edge-weighted graph was shown to be polynomially solvable by Camerini, Galbiati, and Maffioli [13] when the edge weights are nonnegative. They also show that the problem becomes NP-hard when negative weights are allowed. Camerini and Galbiati [12] proposed polynomial-time algorithms for a bounded-path tree problem on graphs with nonnegative edge weights. Their result can be used to show that the minimum-diameter spanning tree problem as well as its natural generalization to Steiner trees can be solved in polynomial time. We use a similar technique to show that the following *minimum-diameter k-tree* problem is polynomially solvable: given a graph with nonnegative edge weights, find a tree of minimum diameter spanning at least k nodes.

We investigate easy and hard results in finding short networks. For this, we use a framework due to T. C. Hu [22]. In this framework, we are given a graph with nonnegative distance values d_{ij} and nonnegative requirement values r_{ij} between every pair of nodes i and j in the graph. The communication cost of a spanning tree is defined to be the sum over all pairs of nodes i, j of the product of the distance between i and j in the tree under d and the requirement r_{ij}. The objective is to find a spanning tree with minimum communication cost. Hu considered the case when all the d values are one and showed that a Gomory–Hu cut tree [21] using the r values as capacities is an optimal solution. Hu also considered the case when all the r values are one and derived sufficient conditions under which the optimal tree is a star. The general version of the latter problem is NP-hard [2], [13], [23].

We define the diameter cost of a spanning tree to be the maximum cost over all pairs of nodes i, j of the distance between i and j in the tree multiplied by r_{ij}. In Table 1, we present current results in this framework. All r_{ij} and d_{ij} values are assumed to be nonnegative. The first two rows of the table examine the cases when either of the two parameters is uniform-valued. The last two rows illustrate that the two problems become NP-complete when both parameters are two-valued.

1.5. Short small trees. We consider the k-tree versions of the minimum-communication-cost and minimum-diameter-cost spanning tree problems and show in

TABLE 1
Results on minimum-communication-cost spanning trees and minimum-diameter-cost spanning trees.

r_{ij}	d_{ij}	Communication cost	Diameter cost
Arbitrary	$\{a\}$	Cut-tree [22]	Open
$\{a\}$	Arbitrary	NP-complete [23]	Poly-time [13]
$\{a, b\}$	$\{0, c\}$	Cut-tree variant (this paper, [22])	Poly-time (this paper)
$\{a, 4a\}$	$\{c, 5c\}$	NP-complete [23]	NP-complete (this paper)

Theorem 6.6 that the minimum-communication k-tree problem and the minimum-diameter k-tree problem are both hard to approximate within any factor even when all the d_{ij} values are one and the r_{ij} values are nonnegative.

In the next section, we present the NP-completeness results. Section 3 contains the $2\sqrt{k}$ approximation for the kMST problem. In §4 we present the stronger result for the case of points in the plane. In §5 we address polynomially solvable cases of the problem. In §6 we prove our results on short trees. We close with a discussion of directions for future research.

2. NP-completeness results.

THEOREM 2.1. *The (decision version of the) kMST problem is NP-complete.*

Proof. It is easy to see that the kMST problem is in NP. In this section we show that the kMST problem is NP-hard by reducing the Steiner tree problem to it. The Steiner tree problem is known to be NP-hard [19]. As an instance of the Steiner tree problem we are given an undirected graph G, a set of terminals R (which is a subset of the vertex set of G), and a positive integer M, and the question is whether there exists a tree spanning R and containing at most M edges. We transform this input to an instance G', k, M of the kMST problem as follows: We let $X = |V(G)| - |R|$ and connect each terminal of G to a distinct path of X new vertices, the path consisting of zero-weighted edges. We assign weight one to the already existing edges of G and set the weight between all other pairs of vertices to ∞ (a very large number). This is the graph G' (see Fig. 1). We set k to be $|R| \cdot (X + 1)$. And now we ask if G' has a tree spanning k vertices of weight at most M. If there exists a Steiner tree in G spanning the set R and containing at most M edges, then it is easy to construct a kMST of weight at most M in G'. Conversely, by our choice of k and X, any kMST in G' must contain at least one node from the path corresponding to each terminal in R. Hence any kMST can be used to derive a Steiner tree for R in G. This completes the reduction. Extensions of hardness to the case of planar graphs and points in the plane follow in a similar way from the hardness of the Steiner tree problem in these restricted cases. Given a planar embedding of G we can create an embedded version of G' since only paths are added.

The NP-hardness holds even when all the edge costs are from the set $\{1, 2, 3\}$. The reduction for this case is similar to the above. Without loss of generality we assume that in the given instance of the Steiner tree problem, G is connected and $M \leq |V| - 1$. We let $X = |V(G)| - |R|$ as before and connect each terminal of G to a distinct set of X vertices by edges of weight one. We set the original edges of G to have weight two and all other edges to have weight three. We choose $k = |R| \cdot X + M + 1$ and the bound on the cost of the kMST to be $|R| \cdot X + 2M$. If there exists a Steiner tree in G spanning the set R and containing at most M edges, then it is easy to construct a kMST of weight at most $|R| \cdot X + 2M$ in G'. This is done by connecting all the newly added vertices to the Steiner tree using the weight-one edges and then

picking up more vertices (note that the graph is connected and $M \leq |V| - 1$) using the weight-two edges until there are $|R| \cdot X + M + 1$ vertices. If there exists a kMST of weight at most $|R| \cdot X + 2M$ in G', then observe that the kMST cannot contain an edge of weight three because it has exactly $k - 1 = |R| \cdot X + M$ edges; and if it contained an edge of weight three, then it would have to contain at least $|R| \cdot X + 1$ edges of weight one but there are only $|R| \cdot X$ edges of weight one in G'. Further, the kMST must span R, and since it has at most M edges of weight two, there must exist a Steiner tree in G spanning R and containing at most M edges. □

When there are only two distinct edge costs, i.e., the graph is complete and every edge has one of two possible weights, the kMST problem can be solved in polynomial time. The basic idea is the following: Let w_1 and w_2 denote the two edge weights, where $w_1 < w_2$. Construct an edge subgraph G_1 of G containing all the edges of weight w_1. Choose a minimum number, say r, of the connected components of G_1 to obtain a total of k nodes (dropping some nodes if necessary). Construct a spanning tree for each chosen component, and connect the trees into a single tree by adding exactly $r - 1$ edges of weight w_2. It is straightforward to verify that the resulting solution is optimal.

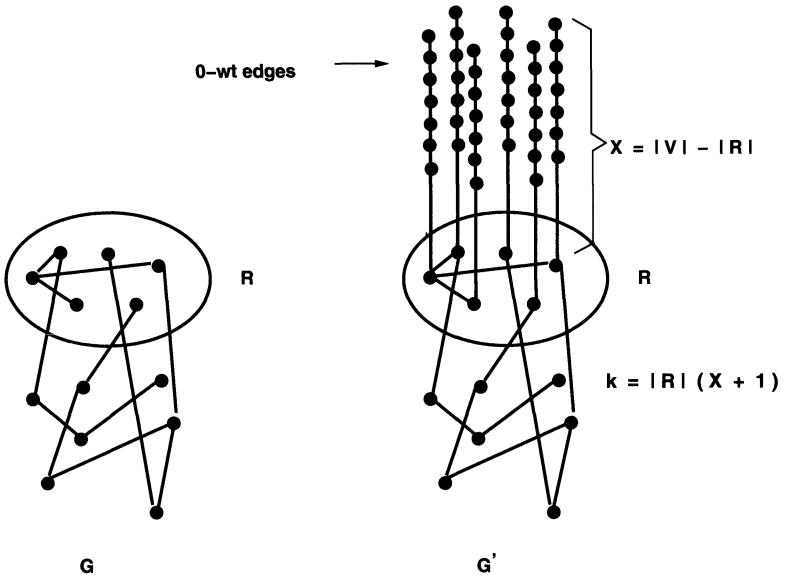


FIG. 1. *The basic NP-hardness reduction from Steiner tree to kMST.*

3. The approximation algorithm for the general case.

THEOREM 3.1. *There is a polynomial-time algorithm that, given an undirected graph G on n nodes with nonnegative weights on its edges and a positive integer $k \leq n$, constructs a tree spanning at least k nodes of weight at most $2\sqrt{k}$ times that of a minimum-weight tree spanning any k nodes.*

In this section, we present the proof of the above theorem. As input, we are given an undirected graph G with nonnegative edge weights and an integer k.

3.1. The algorithm and its running time. It is useful to think of the algorithm as running in two distinct phases: a merge phase and a collect phase.

During the merge phase, the algorithm maintains a set of clusters and a spanning tree on the vertex set of each cluster. Initially each vertex forms a singleton cluster. At each step of the merge phase, we choose an edge of minimum cost among all edges that are between two clusters and merge them by using the edge to connect their spanning trees.

Define the size of a cluster to be the number of vertices that it contains. During the course of the merge phase, the clusters grow in size. The collect phase is entered only when

(i) there exists a set of at most $\sqrt{k}$ clusters containing at least k vertices among themselves, and

(ii) no cluster has size k or more.

In the collect phase, we consider each cluster in turn as the root and perform a shortest-path computation between clusters using the weights on intercluster edges. We determine for each cluster C, the shortest distance d_C such that, within distance d_C from C, there exist at most $\sqrt{k}$ clusters whose sizes sum to at least k. Note that by the first precondition for starting the collect phase, the distance d_C is well defined. We choose the cluster C with the minimum value of d_C and connect it using shortest paths of length at most d_C to each of these $\sqrt{k}$ clusters. We prune edges from some of these shortest paths to output a tree of clusters whose sizes sum to k. We may do this since any cluster has less than k nodes at the start of this phase by the second precondition.

The merge phase of the algorithm continues to run until both the preconditions of the collect phase are satisfied. Beginning with the step of the merge phase after which both preconditions of the collect phase are satisfied, at each subsequent step, the algorithm forks off an execution of the collect phase for the current configuration of clusters. The merge phase continues to run until a cluster of size k or more is formed. Next, the merge phase prunes the edges of the spanning tree of the cluster whose size is between k and $2k$ so as to obtain a spanning tree of size exactly k. At this point, the merge phase terminates and outputs the spanning tree of the cluster of size k. Each forked execution of the collect phase outputs a spanning tree of size between k and $2k$ as well. The algorithm finally outputs the tree of least weight among all these trees. The algorithm is given as follows.

ALGORITHM MERGE-COLLECT

1. Initialize each vertex to be in singleton-connected components and the set of edges chosen by the algorithm to be ϕ. Initialize the iteration count $i = 1$.
2. Repeat until there exists a cluster whose size is between k and $2k$.
   (a) Let $VS_i = \{C_1 \ldots C_l\}$ denote the set of connected components at the start of this iteration. Assume that the components are numbered in nonincreasing order of their size.
   (b) Form an auxiliary graph $G(VS_i, E')$ where the edge (C_i, C_j) between two components is the minimum-cost edge in E whose endpoints belong to C_i and C_j, respectively.
   (c) Choose a minimum-cost edge (C_i, C_j) in $G(VS_i, E')$ and merge the corresponding clusters C_i and C_j.
   (d) $VS_{i+1} = VS_i - \{C_i\} - \{C_j\} \cup \{C_i \cup C_j\}$.
   Remark: This corresponds to one iteration of merge phase.
   (e) Let $j^* = \min\{j : \sum_{i=1}^{j} |C_i| \geq k\}$.
   (f) If $j^* \leq \sqrt{k}$, then $SOL_i = \text{Collect}(G(VS, E'))$.
   (g) $i = i + 1$.

3. Prune the edges of the cluster whose size is between k and $2k$ to obtain a tree with exactly k vertices. Denote the tree obtained by $MSOL$.
4. The output of the heuristic is the minimum valued tree among $MSOL$ and all the SOL_is.

PROCEDURE COLLECT($G(V, E)$)

1. For each cluster vertex C do
   (a) With the cluster C as the root, form a shortest path tree.
   (b) Let d_C be the minimum distance such that there is a set of at most $\sqrt{k}$ clusters within a distance of d_C from C containing at least k vertices.
   (c) Choose these clusters and join them to the root cluster by using the edges in the shortest path tree computed in Step 1(a).
   (d) Prune the edges of the tree to obtain a tree having exactly k nodes.
2. Output the tree corresponding to the choice of the root cluster C that minimizes d_C.

It is easy to see that there are at most $O(n)$ steps in the merge phase and hence at most this many instances of the collect phase to be run. Using Dijkstra's algorithm [15] in each collect phase, the whole algorithm runs in time $O(n^2(m + n \log n))$ where m and n denote the number of edges and nodes in the input graph, respectively. The running time of the collect phase dominates the running time of the merge phase.

3.2. The performance guarantee. Consider an optimal kMST of weight OPT. During the merge phase, nodes of this tree may merge with other nodes in clusters. We focus our attention on the number of edges of the optimal kMST that are *exposed*, i.e., remain as intercluster edges. We show that at any step in which a large number of edges of the kMST are exposed, every edge in the spanning tree of each cluster has small weight.

LEMMA 3.2. *If at the beginning of a step of the merge phase, an optimal kMST has at least x exposed edges (intercluster edges), then each edge in the spanning tree of any cluster at the end of the step has weight at most $\frac{OPT}{x}$.*

Proof. Since the edges are chosen in nondecreasing order of cost, it is clear that each edge in the spanning tree of any cluster at the end of the step has weight at most that of any intercluster edge. Since an optimal kMST has at least x exposed edges, one of these edges has weight at most $\frac{OPT}{x}$. Hence each edge in the spanning tree of any cluster at the end of the step has weight at most $\frac{OPT}{x}$. □

We now prove the performance guarantee in Theorem 3.1. The above lemma is useful as long as the number of exposed edges is high. Applying the lemma with $x = \sqrt{k}$ shows that every edge in the spanning tree of each cluster has weight at most $\frac{OPT}{\sqrt{k}}$. Consider the scenario when the merge phase runs to completion to produce a tree with at least k nodes even before the number of exposed edges falls below $\sqrt{k}$. In this case, since the resulting tree has at most k nodes, the cost of the tree is at most $\frac{OPT}{\sqrt{k}} \cdot k \leq 2\sqrt{k} \cdot OPT$.

Otherwise, the number of exposed edges falls below $\sqrt{k}$ before the merge phase runs to completion. However, in this case, note that both preconditions for the start of the collect phase will have been satisfied. Hence the algorithm must have forked off a run of the collect phase. We show that the tree output by this run has low weight. Consider a shortest-path computation of the collect phase rooted at a cluster containing a node of the optimal kMST. Then clearly, within a distance at most OPT, we find at most $\sqrt{k}$ clusters whose sizes sum to at least k. Since the number of exposed

edges is less than $\sqrt{k}$, the clusters containing nodes of the optimal tree form such a collection. Since there are at most $\sqrt{k}$ clusters to connect to, the weight of these connections is at most $\sqrt{k} \cdot OPT$. To complete the analysis we need to upper-bound the weight of the spanning trees within each of the clusters retained in the output solution. This is not hard since all edges in these clusters have weight at most $\frac{OPT}{\sqrt{k}}$ by Lemma 3.2. Since the size of the output tree is at most k (as a result of the pruning), the total weight of all the edges retained within these clusters is at most $\sqrt{k} \cdot OPT$. By summing the weight of these intracluster edges and the intercluster connections we show that the output tree has cost at most $2\sqrt{k} \cdot OPT$. This proves the performance ratio of $2\sqrt{k}$ claimed in Theorem 3.1.

The example in Fig. 2 shows that the performance ratio of the algorithm is $\Omega(\sqrt{k})$. The optimal kMST is the horizontal path, each edge of which has weight zero or $\frac{OPT}{\sqrt{k}}$. The horizontal path has $\sqrt{k}$ edges of weight $\frac{OPT}{\sqrt{k}}$ each. All zero-weight edges will be chosen first in the merge phase. The merge phase running to completion will extend each of the zero-weight upward-directed paths to include $\Omega(k)$ edges each of weight $\frac{OPT}{4\sqrt{k}}$ resulting in a tree of weight $\Omega(OPT \cdot \sqrt{k})$. The collect phases may output trees consisting of all the $(\sqrt{k}+1)$-sized clusters at the bottom of the figure, each of weight $\Omega(OPT \cdot \sqrt{k})$.

4. An approximation algorithm for points on the plane.

THEOREM 4.1. *There is a polynomial-time algorithm that, given n points in the Euclidean plane and a positive integer $k \leq n$, constructs a tree spanning at least k of these points such that the total length of the tree is at most $O(k^{1/4})$ times that of a minimum-length tree spanning any k of the points.*

In this section, we present a heuristic for the kMST problem for points on the plane and a proof of its performance guarantee. Let $S = \{s_1, s_2, \ldots, s_n\}$ denote the given set of points. For any pair of points s_i and s_j, let $d(i,j)$ denote the Euclidean distance between s_i and s_j.

4.1. The heuristic.

I. **For** each distinct pair of points s_i, s_j in S **do**

(1) Construct the circle C with diameter $\delta = \sqrt{3}d(i,j)$ centered at the midpoint of the line segment $\langle s_i, s_j \rangle$.

(2) Let S_C be the subset of S contained in C. If S_C contains fewer than k points, skip to the next iteration of the loop (i.e., try the next pair of points). Otherwise, do the following.

(3) Let Q be the square of side δ circumscribing C.

(4) Divide Q into k square cells each with side $= \delta/\sqrt{k}$.

(5) Sort the cells by the number of points from S_C they contain and choose the minimum number of cells so that the chosen cells together contain at least k points. If necessary, arbitrarily discard points from the last chosen cell so that the total number of points in all the cells is equal to k.

(6) Construct a minimum spanning tree for the k chosen points. (For the rectilinear case, construct a rectilinear minimum spanning tree for the k chosen points.)

(7) The solution value for the pair $\langle s_i, s_j \rangle$ is the length of this MST.

II. **Output** the smallest solution value found.

It is easy to see that the above heuristic runs in polynomial time. In the next section, we show that the heuristic provides a performance guarantee of $O(k^{1/4})$. We

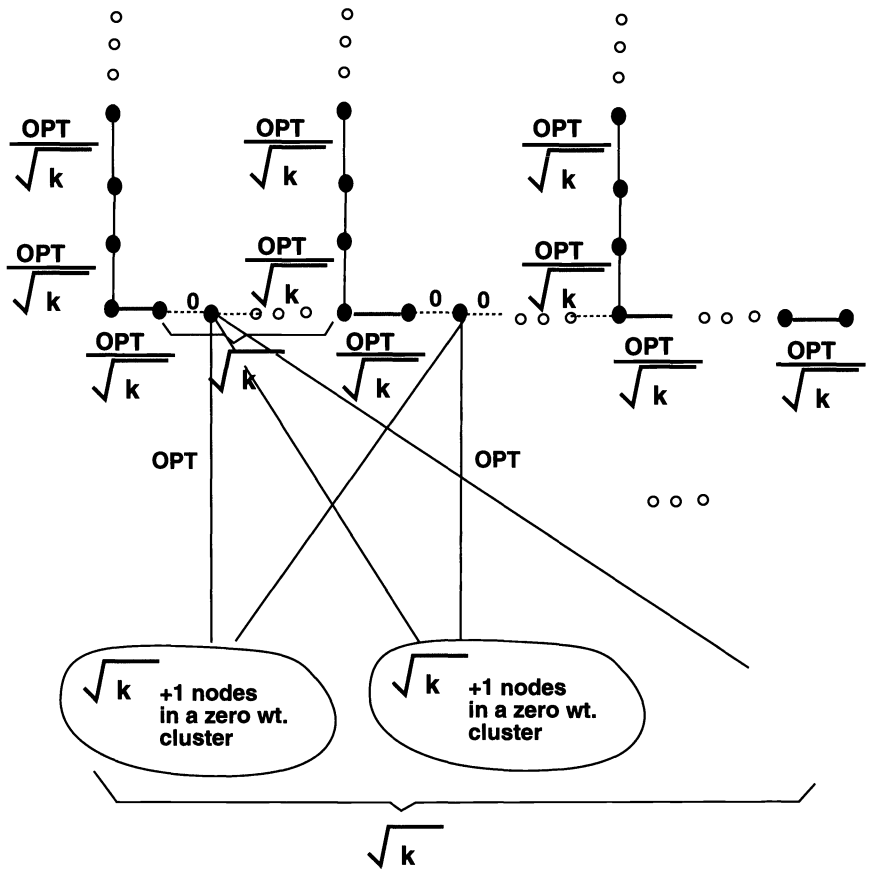


FIG. 2. *Example of a graph in which the algorithm in Theorem* 3.1 *outputs a tree of weight* $\Omega(OPT \cdot \sqrt{k})$.

begin with some lemmas.

4.2. The performance guarantee.

LEMMA 4.2. *Let S denote a set of points on the plane, with diameter Δ. Let a and b be two points in S such that $d(a,b) = \Delta$. Then the circle with diameter $\sqrt{3}\Delta$ centered at the midpoint of the line segment $\langle a,b\rangle$ contains S.*

Proof. Suppose there exists a point $p \in S$ not contained within the circle of diameter $\sqrt{3}\Delta$ centered at the midpoint of the line segment $\langle a,b\rangle$. If p lies on the perpendicular bisector of the line segment $\langle a,b\rangle$, then it is clear that $d(a,p) = d(b,p) > \Delta$, else p is closer to one of a and b than the other. Say p is closer to a; then it is easy to see that $d(b,p) > \Delta$. Thus, if there exists a point outside the circle, then it contradicts the fact that the diameter of the set S is Δ. Hence S must be contained within the circle. □

Lower bounds on an optimal kMST. The following lemma is used to establish a lower bound on OPT.

LEMMA 4.3. *Consider a square grid on the plane with the side of each cell being σ. Then the length of an MST for any set of t points, where each point is from a distinct cell, is $\Omega(t\sigma)$.*

Proof. Pick a point from the set and discard all points in the eight cells neighboring the cell containing the chosen point. Doing this repeatedly we choose a subcollection of $t/9 = \Omega(t)$ points such that the distance between any pair of points in the subcollection is at least σ. The lemma then follows from the observation that the minimum length of a tree spanning $\Omega(t)$ points that are pairwise σ-distant is $\Omega(t\sigma)$. □

Let P^* denote the set of points in an optimal solution to the problem instance. Let Δ denote the *diameter* of P^* (i.e., the maximum distance between a pair of points in P^*) and OPT denote the length of an MST for P^*. Consider an iteration in which the circle constructed by the heuristic is defined by two points a and b in P^* such that $d(a,b) = \Delta$. Let g be the number of square cells used by the heuristic in selecting k points in this iteration. To establish the performance guarantee of the heuristic, we show that the length of the MST constructed by the heuristic during this iteration is within a factor $O(k^{1/4})$ of OPT.

It is easy to see that $OPT \geq \Delta$ because Δ is the diameter of P^*.

Since the heuristic uses a minimum number (g) of square cells in selecting k points, the points in P^* must occur in g or more square cells. Note that the side of each square cell is $\frac{\sqrt{3}\Delta}{\sqrt{k}}$. This gives us the following corollary to Lemma 4.3.

COROLLARY 4.4.

$$OPT = \Omega(\frac{g\Delta}{\sqrt{k}}).$$

Upper bound on the cost of the heuristic. We now prove an upper bound on the cost of the spanning tree returned by the heuristic. For this, we need the following lemma.

LEMMA 4.5. *The length of a minimum spanning tree for any set of q points in a square with side σ is length $O(\sigma\sqrt{q})$.*

Proof. Paste a square grid over the square where each subcell in the grid has side $\frac{\sigma}{\sqrt{q}}$. Connect each point to a closest vertex in the grid. Consider the tree consisting of one vertical line, all the horizontal lines in the grid connected to the vertical line, and the vertical lines connecting each point to its nearest horizontal line (see Fig. 3). It is clear that the grid lines in the tree have total length $O(\sigma\sqrt{q})$ and the lines connecting the points to the grid have total length $q \cdot O(\frac{\sigma}{\sqrt{q}}) = O(\sigma\sqrt{q})$. This is a Steiner tree. But, it is a simple matter to observe that a spanning tree of at most twice the length can be obtained by shortcutting the Steiner tree. □

LEMMA 4.6. *The length of the spanning tree constructed by the heuristic is $O(\sqrt{g}\Delta)$.*

Proof. Let Q_i denote the set of points in the ith cell chosen by the heuristic, $1 \leq i \leq g$. Thus $\sum_{i=1}^{g} |Q_i| = k$. Consider the following two-stage procedure for constructing a spanning tree for the points in $\bigcup_{i=1}^{g} Q_i$.

Stage I. Construct a minimum spanning tree for the points in Q_i, $1 \leq i \leq g$. Note that the points in Q_i are within a square of side $\sqrt{3}\Delta/\sqrt{k}$. Using Lemma 4.5, the length of an MST for Q_i is $O(\frac{\Delta}{\sqrt{k}}\sqrt{|Q_i|})$. Thus, the total length of all the minimum spanning trees constructed in this stage is $O(\frac{\Delta}{\sqrt{k}}\sum_{i=1}^{g}\sqrt{|Q_i|}) = O(\sqrt{g}\ \Delta)$ by the Cauchy–Schwartz inequality.

Stage II. Connect the g spanning trees constructed in Stage I into a single spanning tree as follows. Choose a point arbitrarily from each Q_i ($1 \leq i \leq g$), and construct an MST for the g chosen points. Note that these g points are within a square of side $\sqrt{3}\ \Delta$. Thus, by Lemma 4.5, the length of the MST constructed in this stage is $O(\sqrt{g}\ \Delta)$ as well.

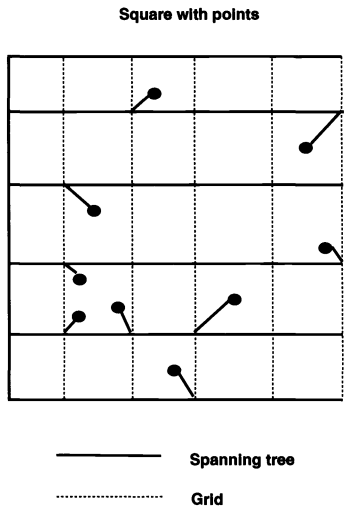


FIG. 3. *A spanning tree of length $O(\sigma\sqrt{q})$ on any q points in a square of side σ.*

Thus, the total length of the spanning tree constructed by the two-stage procedure is $O(\sqrt{g}\,\Delta)$. □

The final analysis. We are now ready to complete the proof of the performance bound. As argued above, $OPT = \Omega(\Delta)$, and from Corollary 4.4, $OPT = \Omega(\frac{g\Delta}{\sqrt{k}})$. Thus $OPT = \Omega(\max\{\Delta, \frac{g\Delta}{\sqrt{k}}\})$. Also from Lemma 4.6, the length of the spanning tree produced by the heuristic is $O(\sqrt{g}\,\Delta)$. Therefore, the performance ratio is $O(\min\{\sqrt{g}, \sqrt{k/g}\}) = O(k^{1/4})$ as claimed.

The example in Fig. 4 shows that the performance ratio of the heuristic is $\Omega(k^{1/4})$. The big square has side σ. Each cell of the square grid has side $\frac{\sigma}{\sqrt{k}}$. There are $\sqrt{k}$ points clustered closely together in each cell along the diagonal of the big square. In each of the $\sqrt{k}$ cells distributed uniformly throughout the big square there are $\sqrt{k}$ uniformly distributed points. The heuristic may pick up the points in the uniformly distributed cells, forming a tree of length $\Omega(\sigma \cdot k^{1/4})$, while the tree spanning the points along the diagonal has length $O(\sigma)$.

Observe that both our lower bounds on an optimal solution and the upper bound on the spanning tree obtained also apply to the case of constructing a rectilinear kMST. Hence it follows that the above approximation algorithm delivers a performance guarantee of $O(k^{1/4})$ for the rectilinear kMST problem too. This proves the following corollary.

COROLLARY 4.7. *There is a polynomial-time algorithm that, given n points in the plane and a positive integer $k \leq n$, constructs a rectilinear tree spanning at least k of these points such that the total length of the tree is at most $O(k^{1/4})$ times that of a minimum-length rectilinear tree spanning any k of the points.*

5. Polynomially solvable special cases.

5.1. kMST for treewidth-bounded (or decomposable) graphs. In this section, we give the details of our polynomial-time algorithm for the class of treewidth-

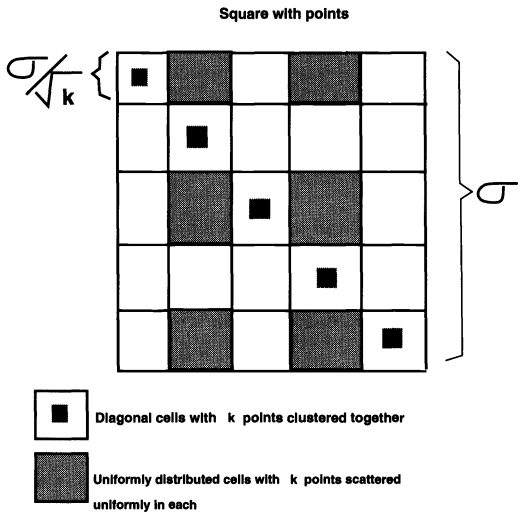


FIG. 4. *Example of a configuration of points on the plane in which the heuristic outputs a tree of length* $\Omega(OPT \cdot \sqrt{k})$.

bounded graphs. As mentioned earlier Arnborg et al. [5] showed that the class of treewidth-bounded graphs is the same as the class of decomposable graphs defined by Bern, Lawler, and Wong [8]. We use the characterization of Bern, Lawler, and Wong to explain our algorithm.

THEOREM 5.1. *For any class of decomposable graphs, there is an* $O(nk^2)$*-time algorithm for solving the kMST problem.*

In this section, we prove the above theorem. A class of *decomposable graphs* Γ is inductively defined as follows [8].

1. The number of primitive graphs in Γ is finite.
2. Each graph in Γ has an ordered set of special nodes called *terminals*. The number of terminals in each graph is bounded by a constant.
3. There is a finite collection of binary composition rules that operate only at terminals, either by identifying two terminals or adding an edge between terminals. A composition rule also determines the terminals of the resulting graph, which must be a subset of the terminals of the two graphs being composed.

Examples of decomposable graphs include trees, series-parallel graphs, and bounded-bandwidth graphs [8].

Let Γ be any class of decomposable graphs. The kMST problem for Γ can be solved optimally in polynomial time using dynamic programming. Following [8], it is assumed that a given graph G is accompanied by a parse tree specifying how G is constructed using the rules and that the size of the parse tree is linear in the number of nodes of G.

Consider a fixed class of decomposable graphs Γ. Suppose that G is a graph in Γ. Let π be a partition of a nonempty subset of the terminals of G. We define the following set of costs for G.

$Cost_i^{\pi}(G)$ = Minimum total cost of any forest containing a tree for each block of π, such that the terminal nodes occurring in each tree are exactly the members of the corresponding block of π, no pair of trees is connected, and the total number of edges in the forest is i $(1 \leq i < k)$.

$Cost_{k-1}^{\emptyset}(G)$ = Minimum cost of a tree within G containing $k-1$ edges and *no* terminal nodes of G.

For any of the above costs, if there is no forest satisfying the required conditions, the value of $Cost$ is defined to be ∞.

Note that because Γ is fixed, the number of cost values associated with any graph in the parse tree for G is $O(k)$. We now show how the cost values can be computed in a bottom–up manner, given the parse tree for G.

To begin with, since Γ is fixed, the number of primitive graphs is finite. For a primitive graph, each cost value can be computed in constant time, since the number of forests to be examined is fixed. Now consider computing the cost values for a graph G constructed from subgraphs G_1 and G_2, where the cost values for G_1 and G_2 have already been computed.

Let Π_{G_1}, Π_{G_2}, and Π_G be the set of partitions of a subset of the terminals of G_1, G_2, and G, respectively. Let A be the set of edges added to G_1 and G_2 by the composition rule R used in constructing G from G_1 and G_2. Corresponding to rule R, there is a partial function $f_R : \Pi_{G_1} \times \Pi_{G_2} \times 2^A \rightarrow \Pi_G$, such that a forest corresponding to partition π_1 in Π_{G_1}, a forest corresponding to partition π_2 in Π_{G_2}, and a subset $B \subseteq A$ combine to form a forest corresponding to partition $f_R(\pi_1, \pi_2, B)$ of G. Furthermore, if the forest corresponding to π_1 contains i edges and the forest corresponding to π_2 contains j edges, then the combined forest in G contains $i+j+|B|$ edges.

Similarly, there is a partial function $g_R : \Pi_{G_1} \times 2^A \rightarrow \Pi_G$, such that a forest corresponding to partition π_1 in Π_{G_1} and a subset $B \subseteq A$ combine to form a forest corresponding to partition $g_R(\pi_1, B)$ of G. If the forest corresponding to π_1 contains i edges, then the combined forest in G contains $i + |B|$ edges. There is also a similar partial function $h_R : \Pi_{G_2} \times 2^A \rightarrow \Pi_G$. Finally, there is a partial function $j_R : 2^A \rightarrow \Pi_G$.

Using functions f_R, g_R, h_R, and j_R, cost values for G can be computed from the set of cost values for G_1 and G_2. For instance, suppose that $f_R(\pi_1, \pi_2, B) = \pi$. Then a contributor to computing $Cost_i^{\pi}(G)$ is $Cost_t^{\pi_1}(G_1) + Cost_{i-t-|B|}^{\pi_2}(G_2) + w(B)$, for each t such that $1 \leq t \leq i - |B| - 1$. Here $w(B)$ is the total cost of all edges in B. The value of $Cost_i^{\pi}(G)$ is the minimum value among its contributors.

When all the cost values for the entire graph G have been computed, the cost of an optimal kMST is equal to $\min_{\pi \in \pi_G}\{Cost_{k-1}^{\pi}(G)\}$, where the forest corresponding to π consists of a single tree.

We now analyze the running time of the algorithm. For each graph occurring in the parse tree, there are $O(k)$ cost values to be computed. Each of the cost values can be computed in $O(k)$ time. As in [8], we assume that the size of the given parse tree for G is $O(n)$. Then the dynamic-programming algorithm takes time $O(nk^2)$. This completes the proof of Theorem 5.1.

It is also easy to see that a straightforward extension of the above algorithm works for the weighted case, when the edges of noninfinite weight form a decomposable graph.

5.2. kMST for points on the boundary of a convex region.

THEOREM 5.2. *There is a polynomial-time algorithm for solving the kMST problem for the case of points in the Euclidean plane that lie on the boundary of a convex region.*

We now restrict our attention to the case where we are given n points that lie on the boundary of a convex region and show that the kMST on these points can be computed in polynomial time using dynamic programming. We also provide a faster algorithm if the points are constrained to lie on the boundary of a circle.

LEMMA 5.3. *Any optimal kMST for a set of points in the plane is non-self-intersecting.*

Proof. Suppose an optimal kMST is self-intersecting; then let $\langle a, b\rangle$ and $\langle c, d\rangle$ be the intersecting line segments. On removing the edges $\langle a, b\rangle$ and $\langle c, d\rangle$ from the kMST we get three connected components; hence some two vertices, one from $\{a, b\}$ and one from $\{c, d\}$, must be in the same connected component. Say a and d are in the same connected component; then since in any convex quadrilateral the sum of two opposite sides is less than the sum of the two diagonals, replacing $\langle a, b\rangle$ and $\langle c, d\rangle$ by $\langle a, c\rangle$ and $\langle b, d\rangle$ we still get a tree spanning k nodes but with less weight. This contradicts the fact that the kMST with which we started out was optimal. Hence any optimal kMST on a set of points in the plane must be non-self-intersecting. □

LEMMA 5.4. *Given n points on the boundary of a convex polygon, no vertex in an optimal kMST of these points has degree greater than* 4.

Proof. Suppose there is a vertex v in an optimal kMST with degree greater than 4. Let $v_1, v_2, \ldots, v_d, d \geq 5$, be its neighbors in the optimal kMST as shown in Fig. 5. Using the well-known fact that any convex polygon lies entirely on one side of a supporting line, we have that $\angle v_1 v v_d \leq 180°$. By the pigeon-hole principle, there is an i such that $\angle v_i v v_{i+1} \leq 180°/(d-1) < 60°, 1 \leq i \leq d-1$, since d is at least 5. Thus in $\triangle v_i v v_{i+1}$, $\angle v_i v v_{i+1}$ is not the largest angle and $v_i v_{i+1}$ is not the largest side. Therefore replacing the larger of vv_i and vv_{i+1} in the optimal kMST with $v_i v_{i+1}$ we obtain a tree with lesser weight, contradicting the assumption that the kMST was optimal. This completes the proof. □

We now characterize the structure of an optimal solution in the following decomposition lemma and use it to define the subproblems that we need to solve recursively using dynamic programming. The next lemma intuitively points out that an optimal solution for the kMST problem for the whole polygon can be constructed from optimal solutions for smaller polygons obtained by triangulating the original polygon.

LEMMA 5.5 (decomposition lemma). *Let $v_0, v_1, \ldots, v_{n-1}$ be the vertices of a convex polygon in, say, clockwise order. Let v_i be a vertex of degree d_i in an optimal kMST. Note that $1 \leq d_i \leq 4$.*

If $d_i \geq 2$ let the removal of v_i from the optimal kMST produce connected components $C_1, C_2, \ldots, C_{d_i}$ (see Fig. 6). Let $|C_i|$ denote the number of vertices in component C_i. Then there exists a partition of $v_{i+1}, v_{i+2}, \ldots, v_{i-1}$ (indices taken mod n*), into d_i contiguous subsegments $S_1, S_2, \ldots, S_{d_i}$ such that $\forall j, 1 \leq j \leq d_i$, the optimal kMST induced on $S_j \bigcup \{v_i\}$ is an optimal $(|C_j|+1)$MST on $S_j \bigcup \{v_i\}$ among all such trees in which the degree of v_i is one.*

If $d_i = 1$, let v_j be v_i's neighbor in the optimal kMST. Let v_j be adjacent to d_{j1} vertices in $v_{i+1}, v_{i+2}, \ldots, v_{j-1}$ and d_{j2} vertices in $v_{j+1}, v_{j+2}, \ldots, v_{i-1}$. Let the optimal kMST contain $|C_1|$ vertices from the set $v_{i+1}, v_{i+2}, \ldots, v_{j-1}$ and $|C_2|$ vertices from the set $v_{j+1}, v_{j+2}, \ldots, v_{i-1}$. Then the optimal kMST induced on $v_{i+1}, v_{i+2}, \ldots, v_j$ is an optimal $(|C_1|+1)$MST on $v_{i+1}, v_{i+2}, \ldots, v_j$ with degree of $v_j = d_{j1}$ and the optimal

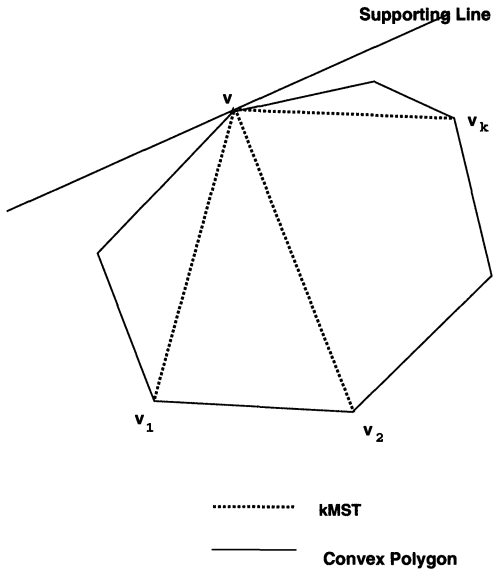


FIG. 5. *Points on a convex polygon.*

kMST induced on $v_j, v_{j+1}, \ldots, v_{i-1}$ *is an optimal* $(|C_2|+1)$*MST on* $v_j, v_{j+1}, \ldots, v_{i-1}$ *among all such trees with degree of* $v_j = d_{j2}$.

Proof. If $d_i \geq 2$, then it is easy to see that a partition of $v_{i+1}, v_{i+2}, \ldots, v_{i-1}$ into contiguous subsegments $S_1, S_2, \ldots, S_{d_i}$ exists such that $\forall j, 1 \leq j \leq d_i, C_j \subset S_j$, because the optimal kMST is non-self-intersecting by Lemma 5.3. Further, the optimal kMST induced on $S_j \bigcup \{v_i\}$ must be an optimal $(|C_j|+1)$MST on $S_j \bigcup \{v_i\}$ with degree of $v_i = 1$, for otherwise we could replace it getting a lighter kMST. The proof of the case when $d_i = 1$ is equally straightforward and is omitted. □

Thus the subproblems we consider are specified by the following four parameters: a size s, a vertex v_i, the degree d_i of v_i, and a contiguous subsegment $v_{k1}, v_{k1+1}, \ldots, v_{k2}$ such that $i \notin [k1 \ldots k2]$. A solution to such a subproblem denoted by $SOLN(s;\ v_i;\ d_i;\ v_{k1}, v_{k1+1}, \ldots, v_{k2})$ is the weight of an optimal sMST on $\{v_i, v_{k1}, v_{k1+1}, \ldots, v_{k2}\}$ in which v_i has degree d_i. Using the decomposition lemma above, we can write a simple recurrence relation for $SOLN(s; v_i; d_i; v_{k1}, v_{k1+1}, \ldots, v_{k2})$ as

$SOLN(s; v_i; d_i; v_{k1}, v_{k1+1}, \ldots, v_{k2}) =$

$$\begin{cases} \infty : \text{if } d_i = 0 \text{ or } s < d_i + 1 \text{ or } ((k2 - k1 + 1) \bmod n) + 1 < s, \\ \min\limits_{k'_0 = k1 < k'_1 \cdots < k'_{d_i} = k2} \ \min\limits_{s_1 \cdots + s_{d_i} = s + d_i - 1, s_j \geq 1} \Sigma_{1 \leq j \leq d_i} SOLN(s_j; v_i; 1; v_{k'_{j-1}}, \ldots, v_{k'_j}) \\ \qquad : \text{if } d_i \geq 2, \\ \min\limits_{j_0 = k1 \leq j_1 \leq j_2 = k2} \{w(v_i v_{j_1}) + \min_{0 \leq d_1 + d_2 \leq 3} \min_{s_1 + s_2 = s} \\ (SOLN(s_1; v_{j_1}; d_1; v_{j_0}, \ldots, v_{j_1 - 1}) + SOLN(s_2; v_{j_1}; d_2; v_{j_1+1}, \ldots, v_{j_2}))\}) : \text{if } d_i = 1. \end{cases}$$

Here $w(v_i v_j)$ is the cost of the edge (v_i, v_j). The optimal kMST is expressed as

$$\min_{1 \leq i \leq n} \min_{1 \leq d \leq 4} SOLN(k; v_i; d; v_{i+1}, v_{i+2}, \ldots, v_{i-1}).$$

Note that we have $O(kn^3)$ subproblems and each subproblem requires looking up the solution to at most $O(k^3 n^3)$ smaller subproblems. This yields a running

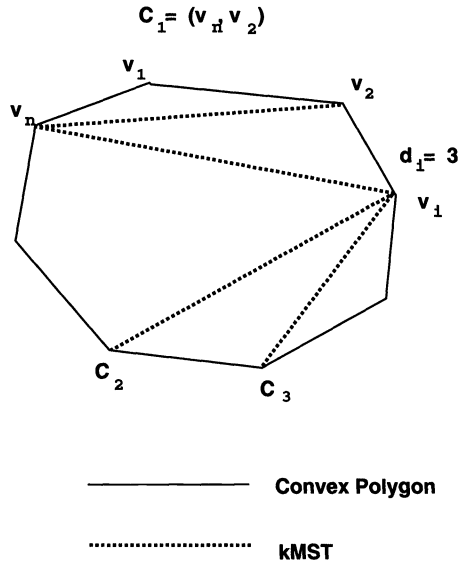


FIG. 6. *Decomposition.*

time of $O(k^4n^6)$. When $k = \Omega(\sqrt{n})$, this running time can be further improved by organizing the computation of the recurrences for the smaller subproblems better. Each subproblem specified by s, v_i, d_i, and the interval $v_{k1}, \ldots, v_{k2}$ can be solved by first computing a partition of the interval into at most four subintervals (exactly four when $d_i = 4$). For the first subinterval, we compute the best tree with $j - 1$ nodes from this subinterval and containing v_i so that it has degree one in this tree, for $1 \leq j \leq s$. This computation takes $O(nk)$ time since there are at most $s \leq k$ trees to be computed, and for each j there are at most n nodes with which v_i shares the single edge in the best tree. Next, we include the next subinterval and compute for $1 \leq j' \leq s$ the best tree on $j' - 1$ nodes containing v_i and nodes from these two subintervals, where v_i have degree two with one edge to a node in the first and one edge to a node in the second subinterval. This set of trees can also be computed in $O(nk)$ time given the set of trees for the first subinterval as follows: First, compute the best tree on j nodes for $1 \leq i \leq s$ containing v_i and nodes only in the second subinterval, where v_i has exactly one edge to a node in this subinterval, in $O(nk)$ time as before. Using these values and the analogous set of values for the first subinterval, the best j' trees for the first two subintervals can be obtained in $O(k^2) = O(nk)$ time since each of the $s \leq k$ trees requires looking up at most s different pairs of trees, one from each subinterval. This method can be extended to compute the solution for the whole set of four subintervals in $O(nk)$ time. Since there are $O(n^3)$ ways to partition a given interval into four subintervals, the recurrence for this subproblem can be solved in $O(kn^4)$ time. So the total time to solve one subproblem is $O(kn^4)$ time. Since there are a total of $O(kn^3)$ subproblems, the total running time of the algorithm is $O(k^2n^7)$.

We now provide a faster algorithm to find the optimal kMST in the case when all n points lie on a circle. We assume that no two points are diametrically opposite.

LEMMA 5.6. *Given n points $v_1, v_2, \ldots, v_n$ on a circle, no vertex in an optimal kMST has degree more than 2.*

Proof. Suppose point v_p in an optimal kMST has degree greater than 2. Then consider the diameter passing through v_p. At least two neighbors of v_p lie on one side of this diameter. Let these neighbors be v_q and v_r, where v_q is closer to v_p than v_r. Then since $\angle v_p v_q v_r$ is obtuse, we replace $v_p v_r$ by $v_q v_r$ to get a smaller tree. □

Lemma 5.6 implies that if the points lie on a circle, then every optimal kMST is a path. Moreover, if the path "zig-zags," then we replace the crossing edge with a smaller edge. Thus we have the following lemma.

LEMMA 5.7. *Given n points $v_1, v_2, \ldots, v_n$ on a circle, let a minimum length k-path on these points be $v_{i_1}, \ldots, v_{i_p}$. Then the line segment joining v_{i_1} and v_{i_p} along with the k-path forms a convex k-gon.*

Proof. By Lemma 5.6 the minimum-length k-path is also the minimum-length kMST. Suppose the line segment joining v_{i_1} and v_{i_p} along with the minimum k-path does not form a convex k-gon. Then there exists a zig-zag in the path as shown in Fig. 7. Say the center of the circle lies to the right of the edge $\langle a, b \rangle$; then we replace $\langle a, b \rangle$ by the edge $\langle b, c \rangle$ to get a smaller kMST, which contradicts the fact that the k-path with which we started was optimal. □

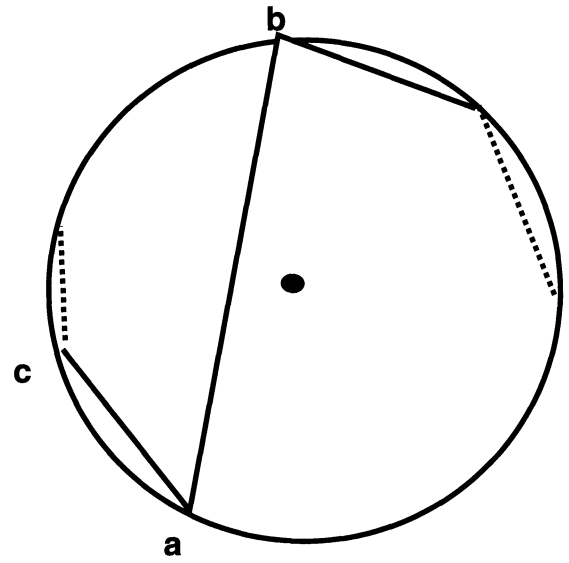


FIG. 7. *Illustration of Lemma* 5.7.

Lemmas 5.6 and 5.7 lead to a straightforward dynamic-programming algorithm to compute an optimal kMST for points on a circle: for each point on the circle compute the minimum-length i-path ($1 \leq i \leq k$), which lies completely on one side of the diameter passing through the point; then combine these solutions to find the optimal kMST. It is easy to see this algorithm takes $O(k^2 n)$ time.

6. Short trees and short small trees.

6.1. Short trees. In this subsection, we prove our results on short trees. First, we address the minimum-diameter k-tree problem: given a graph with nonnegative edge weights, find a tree of minimum diameter spanning at least k nodes.

THEOREM 6.1. *There is a polynomial-time algorithm for the minimum-diameter k-tree problem on graphs with nonnegative edge weights.*

Recall that the diameter of a tree is the maximum distance (path length) between any pair of nodes in the tree. We introduce the notion of subdividing an edge in a weighted graph. A subdivision of an edge $e = (u, v)$ of weight w_e is the replacement of e by two edges $e_1 = (u, r)$ and $e_2 = (r, v)$ where r is a new node. The weights of e_1 and e_2 sum to w_e. Consider a minimum-diameter k-tree. Let x and y be the endpoints of a longest path in the tree. The weight of this path, D, is the diameter of the tree. Consider the midpoint of this path between x and y. If it falls in an edge, we subdivide the edge by adding a new vertex as specified above. The key observation is that there exist at least k vertices at a distance at most $D/2$ from this midpoint. This immediately motivates an algorithm for the case when the weights of all edges are integral and bounded by a polynomial in the number of nodes. In this case, all such potential midpoints lie in half-integral points along edges of which there are only a polynomial number. Corresponding to each candidate point, there is a smallest distance from this point within which there are at least k nodes. We choose the point with the least such distance and output the breadth-first search (bfs) tree rooted at this point appropriately truncated to contain only k nodes.

When the edge weights are arbitrary, the number of candidate midpoints are too many to check in this fashion. However, we use a graphical representation of the distance of any node from any point along a given edge to bound the search for candidate points. We think of an edge $e = (u, v)$ of weight w as a straight line between its endpoints of length w. For any node x in the graph, consider the shortest path from x to a point along the edge e at distance ℓ ($\leq w$) from u. The length of this path is the minimum of $\ell + d(x, u)$ and $w - \ell + d(v, x)$. We plot this distance of the node x as a function of ℓ. The resulting plot is a piecewise linear bitonic curve that we call the roof curve of x in e (see Fig. 8). For each edge e, we plot the roof curves of all the vertices of the graph in e. For any candidate point in e, the minimum diameter of a k-tree centered at this point can be determined by projecting a ray upward from this point in the plot and determining the least distance at which it intersects the roof curves of at least k distinct nodes. The best candidate point for a given edge is one with the minimum such distance. Such a point can be determined by a simple line-sweep algorithm on the plot. Determining the best midpoint over all edges gives the midpoint of the minimum-diameter k-tree. This proves Theorem 6.1.

The following lemma gives yet another way to implement the polynomial-time algorithm for finding a tree of minimum diameter spanning k nodes.

LEMMA 6.2. *Given two vertices in a graph, v_i and v_j, such that every other vertex is within distance d_i of v_i or d_j of v_j, it is possible to find two trees, one rooted at v_i and of depth at most d_i and one rooted at v_j of depth at most d_j, that partition the set of all vertices.*

Proof. Consider the shortest-path trees T_i and T_j rooted at v_i and v_j of depth d_i and d_j, respectively. Every vertex occurs in one tree or both trees. Consider a vertex v_p that occurs in both trees. If it is the case that $d_i - \text{depth}_{T_i}(v_p)$ is greater than $d_j - \text{depth}_{T_j}(v_p)$, then the same is true of all descendants of v_p in T_j. Hence we can remove v_p and all its descendants from T_j since we are guaranteed that all these vertices occur in T_i. Repeating this procedure bottom-up we get two trees satisfying the required conditions and partitioning the vertex set. □

The above lemma motivates the following alternate algorithm for finding a minimum-diameter tree spanning at least k nodes. For each vertex v_i in the graph compute the shortest distance d_i such that there are k vertices within distance d_i of v_i. For each edge (v_i, v_j) compute the least $d_{ij}^i + d_{ij}^j$ such that there are k vertices within distance

d_{ij}^i of v_i or d_{ij}^j of v_j. Then compute the least of all the d_i's and $d_{ij}^i + d_{ij}^j + w(v_i, v_j)$'s, and this is the diameter of the k-tree with least diameter. It can be easily seen that the running time of the algorithm is $O(\min\{k^2, E\}E)$.

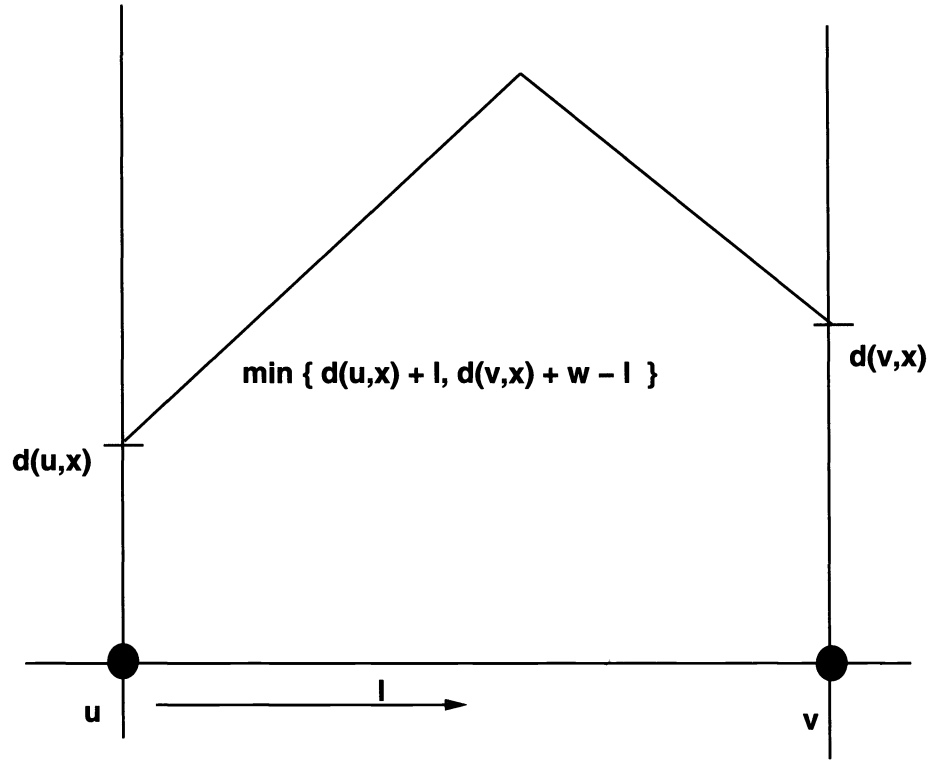


FIG. 8. *A roof curve of a node x in edge $e = (u, v)$.*

We now address the results in the third row of Table 1.

LEMMA 6.3. *If the r_{ij} values are drawn from the set $\{a, b\}$ and the d_{ij} values from $\{0, c\}$, then the minimum-communication-cost spanning tree can be computed in polynomial time.*

Proof. When the d_{ij} values are all uniform, Hu [22] observed that the Gomory–Hu cut tree with the r_{ij} values as capacities is a minimum-communication-cost tree. We can use this result to handle the case when zero-cost d_{ij} edges are allowed as well. We contract the connected components of the graph using zero-cost d_{ij} edges into supernodes. The requirement value r_{IJ} between two supernodes v_I and v_J is the sum of the requirement values r_{ij} such that $i \in v_I$ and $j \in v_J$. Now we find a Gomory–Hu cut tree between the supernodes using the r_{IJ} values as capacities. By choosing an arbitrary spanning tree of zero-d_{ij}-valued edges within each supernode and connecting them to the Gomory–Hu tree, we get a spanning tree of the whole graph. It is easy to verify that this is a minimum-communication-cost spanning tree in this case. □

LEMMA 6.4. *When all the d_{ij} values are uniform and there are at most two distinct r_{ij} values (say a and b), then the minimum-diameter-cost spanning tree can be computed in polynomial time.*

Proof. Let the higher of the two r_{ij} values be a. If the edges with requirement a

form a cyclic subgraph, then any spanning tree has diameter cost $2a$. In this case, any spanning star (a star is a rooted tree of depth 1) is an optimal solution. Otherwise, consider the forest of edges with requirement a. Determine a center for each tree in this forest. Consider the tree formed by connecting these centers in a star. The root of the star is a center of the tree of largest diameter in the forest. If the diameter cost of the resulting tree is less than $2a$, it is easy to see that this tree has optimum diameter cost. Otherwise any star tree on all the nodes has diameter cost $2a$ and is optimal. Note that we can extend this solution to allow zero-cost d_{ij} edges by using contractions as before. □

Now we address the results in the fourth row of Table 1.

LEMMA 6.5. *The minimum-diameter-cost spanning tree problem is NP-complete even when the r_{ij}'s and d_{ij}'s take on at most two distinct values.*

Proof. It is easy to see that the minimum-diameter-cost spanning tree problem is in NP. We now prove that it is NP-hard by using a reduction from an instance of 3SAT. Without loss of generality, we assume that all clauses in the given instance of 3SAT contain three distinct literals. We form a graph that contains a special node t (the "true" node), a node for each literal and each clause. We use two d_{ij} values, c and $5c$ where we assume $c \neq 0$. Each literal is connected to its negation with an edge of distance c. The true node is connected to every literal with an edge of distance c. Each clause is connected to the three literals that it contains with edges of distance c. All other edges in the graph have distance $5c$. Now we specify the requirements on the edges. We use requirement values from $\{a, 4a\}$, where $a \neq 0$. The requirement value of an edge between a literal and its negation is $4a$. The requirement value of all other edges is a (see Fig. 9). It is easy to check that there exists a spanning tree of this graph with diameter cost at most $4ac$ if and only if the 3SAT formula is satisfiable. □

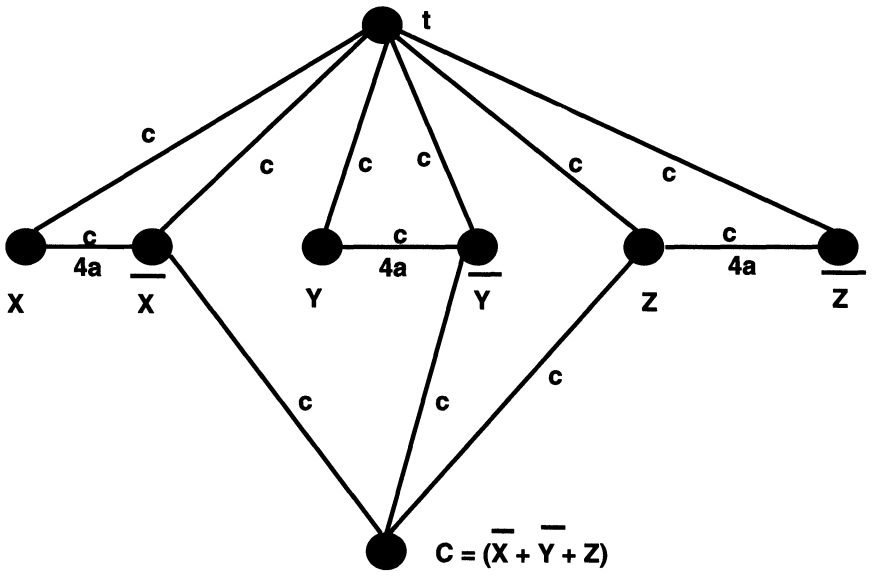


FIG. 9. *Reduction from an instance of 3SAT to the minimum-diameter-cost spanning tree problem.*

6.2. Short small trees.

THEOREM 6.6. *The minimum-communication k-tree problem and the minimum-diameter k-tree problem are both NP-hard to approximate within any factor even when all the d_{ij} values are one and the r_{ij} values are nonnegative.*

Proof. We prove the above theorem for the communication tree case. The proof of the other part is similar. Suppose there is a polynomial-time M-approximation algorithm for the minimum-communication-cost k-tree problem where all the d_{ij} values are one and all r_{ij} values are nonnegative. Then, we show that the k-independent set problem can be solved in polynomial time. The latter problem is well known to be NP-complete [19]. Given a graph G of the k-independent set problem, produce the following instance of the communication k-tree problem: $d_{ij} = 1$ for every pair of nodes i, j; assign $r_{ij} =$ one if (i, j) is *not* an edge in G and $Mk(k-1)+1$ otherwise. If G has an independent set of size k, then we form a star on these k nodes (choosing an arbitrary node as the root). In the star, the distance between any pair of nodes is at most 2 and the r value for each pair is 1. Thus, the communication cost of an optimum solution is at most $k(k-1)$. The approximation algorithm will return a solution of cost at most $Mk(k-1)$. The nodes in this solution are independent in G by the choice of r_{ij} for nonedges $(i, j) \in G$. On the other hand, if there is no independent set of size k in G, the communication cost of any k-tree is greater than $Mk(k-1)$. □

7. Closing remarks.

7.1. Future research. A natural question is whether there are approximation algorithms for the kMST problem that provide better performance guarantees than those presented in this paper. In this direction, Garg and Hochbaum [20] gave an $O(\log k)$-approximation algorithm for the kMST problem for points on the plane using an extension of our lower-bounding technique in §4. Blum, Chalasani, and Vempala [10] recently improved upon this to obtain a constant-factor approximation for points on the plane. Also, Awerbuch, Azar, Blum, and Vempala [1] obtained an $O(\log^2 k)$-approximation algorithm for the kMST problem. An interesting observation in this regard is the following: any edge in an optimal kMST is a shortest path between its endpoints. This observation allows us to assume without loss of generality that the edge weights on the input graph obey the triangle inequality. Although we have been unable to exploit the triangle inequality property in our algorithms, it is possible that this remark holds the key to improving our results.

Table 1 is incomplete. It would be interesting to know the complexity of the minimum-diameter-cost spanning tree problem when the distance values are uniform. Note that any star tree on the nodes provides a 2-approximation to the minimum-diameter-cost spanning tree in this case. The above problem can be shown to be polynomial-time equivalent to the following tree reconstruction problem: given integral nonnegative distances d_{ij} for every pair of vertices i, j, does there exist a spanning tree on these nodes such that the distance between i and j in the tree is at most d_{ij}?

7.2. Maximum acyclic subgraph. In the course of our research we considered the k-forest problem: given an undirected graph is there a set of k nodes that induces an acyclic subgraph? The optimization version of this problem is the maximum acyclic subgraph problem. Since this problem is complementary to the minimum feedback vertex set problem [19], NP-completeness follows. While the feedback vertex set problem is 4-approximable [7], we show that the maximum acyclic subgraph problem is hard to approximate within a reasonable factor using an approximation-preserving

transformation from the maximum independent set problem [6]. This same result was also derived in a more general form in [27].

THEOREM 7.1. *There is a constant $\epsilon > 0$ such that the maximum acyclic subgraph problem cannot be approximated within a factor $\Omega(n^\epsilon)$ unless $P = NP$.*

Proof. Note that any acyclic subgraph of size S contains a maximum independent set of size at least $S/2$ since acyclic subgraphs are bipartite and each partition is an independent set. Further, every independent set is also an acyclic subgraph. These two facts show that the existence of a ρ-approximation algorithm for the maximum acyclic subgraph problem implies the existence of a 2ρ-approximation algorithm for the maximum independent set problem. But by the result in [6] we know that there is a constant $\epsilon > 0$ such that the maximum independent set problem cannot be approximated within a factor $\Omega(n^\epsilon)$ unless $P = NP$. Hence, the same is true of the maximum acyclic subgraph problem. □

Acknowledgments. The authors thank Alex Zelikovsky and Naveen Garg for helpful conversations during the initial stages of the paper. They are grateful to Professor John Oomen for observing that our algorithm for points in the plane extends to the rectilinear case and to Professor Arie Tamir for his observations on §6. We thank the referees for detailed comments and suggestions that substantially improved the quality of presentation.

REFERENCES

[1] B. AWERBUCH, Y. AZAR, A. BLUM, AND S. VEMPALA, *Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen*, in Proc. 27th Annual ACM Symposium on Theory of Computing, Las Vegas, NV, 1995, pp. 277–283.

[2] D. ADOLPHSON AND T. C. HU, *Optimal linear ordering*, SIAM J. Appl. Math., 25 (1973), pp. 403–423.

[3] A. AGGARWAL, H. IMAI, N. KATOH, AND S. SURI, *Finding k points with minimum diameter and related problems*, J. Algorithms, 12 (1991), pp. 38–56.

[4] A. AGRAWAL, P. KLEIN, AND R. RAVI, *When trees collide: an approximation algorithm for the generalized Steiner tree problem on networks*, in Proc. 23rd Annual ACM Symposium on Theory of Computing, New Orleans, LA, 1991, pp. 134–144.

[5] S. ARNBORG, B. COURCELLE, A. PROSKUROWSKI, AND D. SEESE, *An algebraic theory of graph reduction*, J. Assoc. Comput. Mach., 40 (1993), pp. 1134–1164.

[6] S. ARORA, C. LUND, R. MOTWANI, M. SUDAN, AND M. SZEGEDY, *Proof verification and hardness of approximation problems*, in Proc. 33rd IEEE Symposium on the Foundations of Computer Science, Pittsburgh, PA, 1992, pp. 14–23.

[7] R. BAR-YEHUDA, D. GEIGER, J. NAOR, AND R. M. ROTH, *Approximation algorithms for the cycle-cover problem with applications to constraint satisfaction and Bayesian inference*, in Proc. Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, 1994, pp. 344–354.

[8] M. W. BERN, E. L. LAWLER, AND A. L. WONG, *Linear time computation of optimal subgraphs of decomposable graphs*, J. Algorithms, 8 (1987), pp. 216–235.

[9] A. BLUM, P. CHALASANI, D. COPPERSMITH, B. PULLEYBLANK, P. RAGHAVAN, AND M. SUDAN, *The minimum latency problem*, in Proc. 26th Annual ACM Symposium on Theory of Computing, Montreal, Canada, 1994, pp. 163–172.

[10] A. BLUM, P. CHALASANI, AND S. VEMPALA, *A constant-factor approximation for the k-MST problem in the plane*, in Proc. 27th Annual ACM Symposium on Theory of Computing, Las Vegas, NV, 1995, pp. 294–302.

[11] H. L. BODLAENDER, *Dynamic programming on graphs of bounded treewidth*, in Proc. 15th International Colloquium on Automata, Languages and Programming, Springer-Verlag, Berlin, New York, NY, 1988, pp. 105–118.

[12] P. M. CAMERINI AND G. GALBIATI, *The bounded path problem*, SIAM J. Alg. Discrete Methods, 3 (1982), pp. 474–484.

[13] P. M. CAMERINI, G. GALBIATI, AND F. MAFFIOLI, *Complexity of spanning tree problems: Part 1*, European J. Oper. Res., 5 (1980), pp. 346–352.

[14] N. CHRISTOFIDES, *Worst-case analysis of a new heuristic for the traveling salesman problem*, Report 338, GSIA, CMU, Pittsburgh, PA, 1976.

[15] E. W. DIJKSTRA, *A note on two problems in connexion with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[16] D. P. DOBKIN, R. L. DRYSDALE, AND L. J. GUIBAS, *Finding smallest polygons*, in Advances in Computing Research 1, JAI Press, Greenwich, CT, 1983, pp. 181–214.

[17] D. EPPSTEIN, *New algorithms for minimum area k-gons*, in Proc. 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, Orlando, FL, 1992, pp. 83–88.

[18] D. EPPSTEIN AND J. ERICKSON, *Iterated nearest neighbors and finding minimal polytopes*, in Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, Austin, TX, 1993, pp. 64–73.

[19] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman and Co., San Francisco, CA, 1979.

[20] N. GARG AND D. HOCHBAUM, *An $O(\log k)$ approximation algorithm for the k minimum spanning tree problem in the plane*, in Proc. 26th ACM Symposium on Theory of Computing, Montreal, Canada, 1994, pp. 432–438.

[21] R. E. GOMORY AND T. C. HU, *Multi-terminal network flows*, SIAM J. Appl. Math., 9 (1961), pp. 551–570.

[22] T. C. HU, *Optimum communication spanning trees*, SIAM J. Comput., 3 (1974), pp. 188–195.

[23] D. S. JOHNSON, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, *The complexity of the network design problem*, Networks, 8 (1978), pp. 279–285.

[24] P. KLEIN AND R. RAVI, *A nearly best-possible approximation for node-weighted Steiner trees*, in Proc. 3rd MPS conference on Integer Programming and Combinatorial Optimization, 1993, pp. 323–332.

[25] J. B. KRUSKAL, *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proc. Amer. Math. Soc., 7 (1956), pp. 48–50.

[26] D. LOZOVANU AND A. ZELIKOVSKY, *Minimal and bounded tree problems*, Tezele Congresului XVIII al Academiei Romano-Americane, Kishniev, 1993, p. 25.

[27] C. LUND AND M. YANNAKAKIS, *On the hardness of the maximum subgraph problems*, in Proc. 20th International Colloquium on Automata, Languages and Programming, Lund, Sweden, 1993, Springer-Verlag, Berlin, New York, NY, pp. 40–51.

[28] R. C. PRIM, *Shortest connection networks and some generalizations*, Bell System Tech. J., 36 (1957), pp. 1389–1401.

[29] R. RAVI, *Steiner Trees and Beyond: Approximation Algorithms for Network Design*, Ph.D. Thesis, Tech. report TR-CS-93-41, Department of Computer Science, Brown University, Providence, RI, September 1993.

[30] R. RAVI, M. V. MARATHE, S. S. RAVI, D. J. ROSENKRANTZ, AND H.B. HUNT III, *Many birds with one stone: multi-objective approximation algorithms*, in Proc. 25th Annual ACM Symposium on the Theory of Computing, San Diego, CA, 1993, pp. 438–447.

[31] R. RAVI, R. SUNDARAM, M. V. MARATHE, D. J. ROSENKRANTZ, AND S. S. RAVI, *Spanning trees short or small*, in Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, 1994, pp. 546–555.

[32] N. ROBERTSON AND P. SEYMOUR, *Graph minors IV: Tree-width and well-quasi-ordering*, J. Combin. Theory Ser. B, 48 (1990), pp. 227–254.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 201–209, May 1996

004

ON THE NONEXISTENCE OF PERFECT CODES IN THE JOHNSON SCHEME*

TUVI ETZION†

Abstract. Although it was conjectured by Delsarte in 1973 that no nontrivial perfect codes exist in the Johnson scheme, only very partial results are known. In this paper we considerably reduce the range in which perfect codes in the Johnson scheme can exist; e.g., we show that there are no nontrivial perfect codes in the Johnson graph $J(2w+p, w)$, p prime. We give theorems about the structure of perfect codes if they exist. This involved structure gives more evidence in support of the belief that no nontrivial perfect codes exist in the Johnson scheme.

Key words. perfect code, Steiner system, Hamming scheme, Johnson graph, Johnson scheme

AMS subject classifications. 05B15, 05B30, 51E10, 94B25

1. Introduction. Perfect codes always draw the attention of coding theorists and mathematicians. They are defined on large varieties of metrics, e.g., Hamming, Johnson, and Lee [7]. Although a lot of results are known regarding the Hamming metric [4], [7], and the Lee metric [5], [7] (and we don't give the enormous number of references in order to save space), only a few results are known on perfect codes in the Johnson metric.

In the Johnson scheme, we are given two integers, n and w, such that $0 \leq w \leq n$. Given a binary code C, all its codewords have length n and constant weight w. Two codewords u and v are in distance (*J-distance*) d apart if there are exactly d positions in which u has 1 value and v has 0 values. Obviously, there are exactly d other positions in which u has 0 values and v has 1 value. With the Johnson scheme we associate the Johnson graph $J(n, w)$. The vertex set V_w^n of the Johnson graph consists of all w-subsets of a fixed n-set. Two such w-subsets are adjacent if and only if their intersection has size $w-1$. A code C of such w-subsets is called e-perfect code in $J(n, w)$ (or in the Johnson scheme) if the e-spheres of all the codewords of C form a partition of V_w^n. In other words, C is an e-perfect code if for each element $v \in V_w^n$ there exists a unique element $c \in C$, such that the J-distance between v and c is less than or equal to e. There are some trivial perfect codes in $J(n, w)$.

1. V_w^n is 0-perfect.
2. Any $\{v\}$, $v \in V_w^n$, is w-perfect.
3. If $n = 2w$, w odd, any pair of disjoint w subsets is e-perfect with $e = \frac{1}{2}(w-1)$.

Delsarte [3, p. 55] conjectured that $J(n, w)$ doesn't contain nontrivial perfect codes. Bannai [1] proved the nonexistence of e-perfect codes in $J(2w-1, w)$ and $J(2w+1, w)$ for $e \geq 2$. Hammond [6] extended the result and showed that $J(n, w)$ cannot contain a nontrivial perfect code for $n \in \{2w-2, 2w-1, 2w+1, 2w+2\}$. Generalizations of Lloyd's theorem [1], [3], didn't lead to significant results. A significant improvement was made by Roos [8] who showed the following result.

THEOREM 1.1. *If an e-perfect code in $J(n, w)$, $n \geq 2w$, exists, then $n \leq (w-1)(2e+1)/e$.*

* Received by the editors October 10, 1994; accepted for publication (in revised form) April 5, 1995. This research was supported in part by Science and Engineering Research Council (United Kingdom) grant GR/K01605.

† Computer Science Department, Royal Holloway College, University of London, Egham, Surrey TW20 0EX, UK. The author is on leave of absence from the Computer Science Department, Technion–Israel Institute of Technology, Haifa 32000, Israel.

In this paper we make a considerable improvement on the range in which perfect codes cannot exist. We show that there are strong connections between perfect codes and Steiner systems. If nontrivial perfect code exists then many Steiner systems are embedded in it. They are embedded in such an involved way that it seems impossible that such a structure can exist. The necessary conditions on the existence of these Steiner systems reduce the range in which these perfect codes can exist.

This paper is organized as follows. In §2 we introduce the necessary notation and definitions on codes and Steiner systems that are needed for our discussion. We also give some simple results that are essential for the discussion. In §3 we give two theorems that connect the existence of perfect codes with the existence of Steiner systems. The proofs of these theorems reveal the involved structure of the perfect codes. In §4 we prove that, except for the trivial perfect codes, there cannot exist perfect codes which are also Steiner systems. In §5 we give a concept similar to the weight distribution on e-perfect codes in the Johnson scheme. Using this concept we will show that there are no e-perfect codes in $J(2w+e+1,w)$. In §6 we examine the theorems of §§3, 4, and 5 to show that the range in which e-perfect codes exist is considerably reduced. We also explore the involved structure obtained in §3.

2. Notation, definitions, and preliminaries. Perfect codes in the Johnson scheme have a strong connection to constant weight codes and Steiner systems. For this purpose we need to use the Hamming metric. Two binary words u and v of the same length n have Hamming distance (*H-distance*) d if they differ in exactly d positions. Note that two words $u,v \in V_w^n$, have J-distance d if and only if their H-distance is $2d$. A code C has minimum H-distance d if for any two codewords $u,v \in C$, the H-distance between u and v is at least d.

LEMMA 2.1. *If C is an e-perfect code in the Johnson scheme then its minimum H-distance is $4e+2$.*

Proof. Since C is an e-perfect code, it follows that the e-spheres of two words with J-distance less than $2e+1$ have nonempty intersection. Hence, the minimum J-distance of the code is $2e+1$ and its minimum H-distance is $4e+2$. □

An (n,d,w) code is a code of length n, constant weight w to all the codewords, and minimum H-distance d. $A(n,d,w)$ denote the maximum size of an (n,d,w) code. An extensive survey on the lower bounds on $A(n,d,w)$ can be found in [2].

LEMMA 2.2. *If C is an e-perfect code in $J(n,w)$ then $A(n,4e+2,w)=|C|$.*

Proof. Assume C is an e-perfect code in $J(n,w)$. By Lemma 2.1 C has minimum H-distance $4e+2$, and hence it is an $(n,4e+2,w)$ code. Given an $(n,4e+2,w)$ code, in the Johnson scheme, its minimum J-distance is $2e+1$ and hence the e-spheres around its codewords are disjoint. The lemma follows from the facts that all e-spheres in $J(n,w)$ have the same size and in an e-perfect code they form a partition of V_w^n. □

A Steiner system $S(t,k,n)$ is a collection of k-subsets (called *blocks*) taken from an n-set such that each t-subset of the n-set is contained in exactly one block. The following theorem is well known, e.g., [7, p. 60].

THEOREM 2.3. *A necessary condition that a Steiner system $S(t,k,n)$ exists is that the numbers $\binom{n-i}{t-i}/\binom{k-i}{t-i}$ be integers for $0 \le i \le t$.*

Henceforth, let $N=\{1,2,\ldots,n\}$ be the n-set. From a Steiner system $S(t,k,n)$ we construct constant weight code on n coordinates as follows. From each block B we construct a codeword with 1's in the positions of B and 0's in the positions of $N \setminus B$. This construction leads to the following well-known theorem [2].

THEOREM 2.4. *$A(n, 2(k-t+1), k) = \frac{n(n-1)\cdots(n-t+1)}{k(k-1)\cdots(k-t+1)}$ if and only if a Steiner system $S(t,k,n)$ exists.*

From Theorem 2.4 and Lemma 2.1 we immediately infer the following result.

LEMMA 2.5. *If C is an e-perfect code in $J(n,w)$ and is also a Steiner system then it is a Steiner system $S(w-2e, w, n)$.*

COROLLARY 2.6. *If C is an e-perfect code in $J(n,w)$ which is not a Steiner system $S(w-2e,w,n)$ then there exists at least one set of $w-2e$ coordinates which are not contained in any codeword.*

LEMMA 2.7. *The complement of an e-perfect code in $J(n,w)$ is an e-perfect code in $J(n, n-w)$.*

Proof. This lemma is a simple observation from the fact that $J(n,w)$ and $J(n,n-w)$ are isomorphic under the mapping which maps each vertex to its complement. □

Finally, we need a few more definitions which we will use in the proofs of the nonexistence theorems in the sequel. For a given partition of N into two subsets A and B such that $|A| = k$ and $|B| = n-k$, let *configuration* (i,j) consist of all vectors with weight i in the positions of A and weight j in the positions of B. For an e-perfect code C in $J(n,w)$ we say that $u \in C$ *J − cover* $v \in V_w^n$ if the J-distance between u and v is less than or equal to e. For a given two subsets u and v we say that u *C − cover* v if v is a subset of u (this is our usual understanding of the word cover).

In the sequel we will use a mixed language of set and vector notations. It should be understood from the context which one we are using and the translation between the two different notations.

3. Perfect codes and Steiner systems. In this section we will prove that if there exists an e-perfect code in the Johnson scheme, then many Steiner systems are embedded in it. This fact will force the necessary conditions for the existence of these Steiner systems also to become necessary conditions for the existence of the e-perfect codes. The involved way in which these Steiner systems are embedded in the perfect codes will make it reasonable to believe that except for the trivial perfect codes no other e-perfect codes exist in the Johnson scheme.

THEOREM 3.1. *If an e-perfect code in $J(n,w)$ exists, then a Steiner system $S(e+1, 2e+1, w)$ exists.*

Proof. Assume C is an e-perfect code in $J(n,w)$. We partition N into two subsets A and B, such that $|A| = w$, $|B| = n-w$, and the vector of the $(w,0)$ configuration is a codeword. This codeword J-covers exactly all the vectors of all configurations $(w-x, x)$, where $0 \le x \le e$. Since C is e-perfect code and all vectors of all configurations $(w-x,x)$, $0 \le x \le e$, are covered, it follows that C does not contain any codeword of any configurations $(w-x,x)$, where $1 \le x \le 2e$. Therefore, all words of configuration $(w-e-1, e+1)$ must be J-covered by codewords from configuration $(w-2e-1, 2e+1)$. Consider now all $\binom{w}{e+1}$ vectors in configuration $(w-e-1, e+1)$ with $e+1$ 1's in $e+1$ fixed positions of B. These vectors are J-covered by codewords from configuration $(w-2e-1, 2e+1)$ with $2e+1$ 1's in positions of B which C-covers the $e+1$ fixed positions. Let C_1 be this set of codewords. Each subset of $e+1$ 0's in A with these $e+1$ fixed positions in B must be C-covered and no subset can be C-covered twice (since the code is perfect). Hence, the complement of the A part of C_1 forms a Steiner system $S(e+1, 2e+1, w)$. □

By using Lemma 2.7 we also have the following corollary.

COROLLARY 3.2. *If an e-perfect code in $J(n,w)$ exists, then a Steiner system $S(e+1, 2e+1, n-w)$ exists.*

THEOREM 3.3. *If an e-perfect code in $J(n,w)$, which is not a Steiner system $S(w-2e,w,n)$, exists then for some k, $0 \le k \le e-1$, a Steiner system $S(2,2e-k+1,n-w+2e-2k)$ exists.*

Proof. The proof will be given in some kind of inductive approach. Assume C is an e-perfect code in $J(n,w)$ which is not a Steiner system $S(w-2e,w,n)$. We partition N into two subsets, A_0 and B_0, such that $|A_0| = w-2e$, $|B_0| = n-w+2e$, and there are no codewords in C from configuration $(w-2e,2e)$, but there is at least one codeword from configuration $(w-2e-1,2e+1)$. This can be done as a simple consequence from Corollary 2.6. For a given k, $0 \le k \le e-2$, assume N is partitioned into two subsets A_k and B_k, such that $|A_k| = w-2e+2k$, $|B_k| = n-w+2e-2k$, there are no codewords in C from any configuration $(w-2e+i,2e-i)$, $k \le i \le 2k$, but there is at least one codeword from configuration $(w-2e+k-1,2e-k+1)$. Let C_k be the set of codewords from configuration $(w-2e+k-1,2e-k+1)$. In the B_k part of C_k we search for two coordinates in which each codeword has at least one 0. If none exist then the B_k part forms a Steiner system $S(2,2e-k+1,n-w+2e-2k)$ (note, that if two codewords of C_k have two 1's in the same two coordinates of the B_k part their H-distance will be $4e$, contradicting Lemma 2.1). If these two coordinates exist, we join them to A_k to obtain A_{k+1} and $B_{k+1} = N \setminus A_{k+1}$. Now, $|A_{k+1}| = w-2e+2(k+1)$, $|B_{k+1}| = n-w+2e-2(k+1)$, and there are no codewords in C from any configuration $(w-2e+i,2e-i)$, $k+1 \le i \le 2(k+1)$, but there is at least one codeword from configuration $(w-2e+k,2e-k)$. If $k = e-2$ and we obtain $|A_{e-1}| = w-2$, $|B_{e-1}| = n-w+2$, there are no codewords in C from any configuration $(w-2e+i,2e-i)$, $e-1 \le i \le 2e-2$, but there is at least one codeword in C from configuration $(w-e-2,e+2)$. This means that vectors of configuration $(w-2,2)$ can be J-covered only by codewords of configuration $(w-e-2,e+2)$. Since each vector of configuration $(w-2,2)$ is J-covered exactly once, it follows that the B_{e-1} part of the codewords from configuration $(w-e-2,e+2)$ forms a Steiner system $S(2,e+2,n-w+2)$. □

COROLLARY 3.4. *If an e-perfect code in $J(n,w)$, which is not a Steiner system $S(n-w-2e,n-w,n)$, exists then for some k, $0 \le k \le e-1$, a Steiner system $S(2,2e-k+1,w+2e-2k)$ exists.*

4. Only trivial Steiner systems are perfect codes. As said before, for $n = 2w$, w odd, any pair of disjoint w-subsets is e-perfect with $e = \frac{1}{2}(w-1)$. These two w-subsets form a Steiner system $S(1,w,n)$. For any n and $1 \le w \le n$, V_w^n is 0-perfect, and it forms a Steiner system $S(w,w,n)$. A natural question is whether there exist more perfect codes which are also Steiner systems. The answer to this question is our next theorem. But, first we need the following simple lemma.

LEMMA 4.1. *If a Steiner system $S(w-k,w,n)$, $k \ge 1$, exists then $n \ge 2w$.*

Proof. Assume $n < 2w$ and a Steiner system $S(w-k,w,n)$, $k \ge 1$, exists. The number of blocks in this system is

$$\frac{\binom{n}{w-k}}{\binom{w}{w-k}} = \frac{n! \cdot k!}{(n-w+k)! \cdot w!}.$$

The number of blocks in a packing of $(n-w)$-subsets of N in which each $(n-w-k)$-subset of N is contained in at most one block is less than or equal to

$$\frac{\binom{n}{n-w-k}}{\binom{n-w}{n-w-k}} = \frac{n! \cdot k!}{(w+k)! \cdot (n-w)!}.$$

If $n < 2w$ and $k \geq 1$ then obviously

$$\frac{\binom{n}{w-k}}{\binom{w}{w-k}} > \frac{\binom{n}{n-w-k}}{\binom{n-w}{n-w-k}}.$$

But since the complement of the code derived from the Steiner system $S(w-k, w, n)$ is a packing of $(n-w)$-subsets of N in which each $(n-w-k)$-subset of N is contained at most in one block, we have a contradiction. Hence, if a Steiner system $S(w-k, w, n)$, $k \geq 1$, exists then $n \geq 2w$. □

THEOREM 4.2. *Except for the Steiner systems $S(1, w, n)$ and $S(w, w, n)$, there are no more Steiner systems which are also perfect codes in the Johnson scheme.*

Proof. Assume C is an e-perfect code in $J(n, w)$ which is also a Steiner system. Since C is e-perfect it follows by Lemma 2.1 that C has minimum H-distance $4e+2$, and by Lemma 2.5 it is a Steiner system $S(w-2e, w, n)$. Now, we partition N into two subsets A and B such that $|A| = w - 2e + 1$, $|B| = n - w + 2e - 1$, and there is no codeword in C from configuration $(w - 2e + 1, 2e - 1)$. Since C is a Steiner system $S(w-2e, w, n)$ and no word in C is from configuration $(w - 2e + 1, 2e - 1)$ it follows that there are $w - 2e + 1$ codewords in C from configuration $(w - 2e, 2e)$. Since the minimum H-distance of C is $4e + 2$ it follows that the $2e$ elements in B, of any two codewords from configuration $(w - 2e, 2e)$, must be disjoint. Hence, we have $n \geq (w - 2e + 1) + (w - 2e + 1)2e = (w - 2e + 1)(2e + 1)$. By Lemma 4.1, $n \geq 2w$, and hence by Theorem 1.1 we have $(w-1)(2e+1)/e \geq n$, and therefore

$$(w-1)(2e+1)/e \geq (w-2e+1)(2e+1)\ .$$

We now distinguish between two cases.

Case 1. For $e > 1$ this implies $2e + 1 \geq w$ Therefore, the intersection between any two codewords is empty since the minimum H-distance is $4e + 2$. Thus, the code contains two codewords, $n = 2w$, and the Steiner system is $S(1, w, n)$.

Case 2. For $e = 1$ this implies $n = 3w - 3$. By Theorem 3.1 a Steiner system $S(2, 3, w)$ exists and hence by Theorem 2.3, $w \equiv 1$ or 3 (mod 6). Therefore, we have $n \equiv 0 \pmod 6$. By Corollary 3.2, Steiner system $S(2, 3, n - w)$ exists also and hence by Theorem 2.3, $n - w \equiv 1$ or 3 (mod 6). By Theorem 3.3 a Steiner system $S(2, 3, n - w + 2)$ exists also and hence by Theorem 2.3, $n - w + 2 \equiv 1$ or 3 (mod 6) which implies that $n - w \equiv 1 \pmod 6$. Since $n \equiv 0 \pmod 6$, it follows that $w \equiv 5 \pmod 6$, a contradiction.

Thus, no nontrivial perfect code is a Steiner system. □

5. No e-perfect codes in $J(2w + e + 1, w)$. By Theorem 3.1 and Corollary 3.2, if an e-perfect code exists in $J(n, w)$ then Steiner systems $S(e+1, 2e+1, w)$ and $S(e+1, 2e+1, n-w)$ exist. By the divisibility conditions of Theorem 2.3 this implies that $e+1$ divides $w - e$ and $n - w - e$, i.e., $n - w \equiv w \equiv e \pmod{e+1}$. This implies that e-perfect codes might exist in $J(2w + e + 1, w)$. In this section we prove that no nontrivial e-perfect codes exist in $J(2w + e + 1, w)$. This result and the results of the previous sections enable us to show many Johnson graphs in which no nontrivial perfect codes exist. The proof will proceed in a few steps which also show some properties of e-perfect codes if they exist. Assume C is an e-perfect code in $J(n, w)$ and N is partitioned into two parts A and B such that $|A| = w$ and $|B| = n - w$. Let $\{D_{(i,j)} : 0 \leq i, j,\ i + j = w\}$ denote the *configuration distribution* of the code; i.e., $D_{(i,j)}$ denote the number of codewords from configuration (i, j).

THEOREM 5.1. *There are exactly $e+1$ different configuration distributions for an e-perfect code. If W_k, $0 \le k \le e$, is the set of the kth configuration distribution then W_k contains $D_{(w-k,k)}$ and $D_{(w-2e-1+k,\ 2e+1-k)}$ as the only nonzero elements among $D_{(w-i,i)}$, $0 \le i \le 2e+1-k$.*

Proof. Let k be the smallest integer such that C has a codeword from configuration $(w-k,k)$. Since we must J-cover the vector from configuration $(w,0)$, it follows that $0 \le k \le e$. Since by Lemma 2.1 the minimum H-distance of C is $4e+2$, it follows that there is exactly one codeword from configuration $(w-k,k)$ and no codewords from any configuration $(w-j,j)$, $k+1 \le j \le 2e-k$. The codeword from configuration $(w-k,k)$ J-covers all vectors from configurations $(w-i,i)$ for all i, $0 \le i \le e-k$. Vectors from configurations $(w-e+k-1, e-k+1)$ are J-covered by the codeword from the $(w-k,k)$ configuration if $k > 0$, and the rest, which are the most, can be J-covered only by codewords from configuration $(w-2e-1+k, 2e+1-k)$. Note, that we can always partition N into A and B such that the first codeword will have $w-k$ 1's in A and k 1's in B, and hence C contains a codeword from configuration $(w-k,k)$. To complete the proof we have to show that once we are given k, $0 \le k \le e$, such that a codeword from configuration $(w-k,k)$ is in the code (i.e., $D_{(w-k,k)} = 1$, $D_{(w-i,i)} = 0$, $0 \le i \le 2e-k$, $i \ne k$), then the configuration distribution is determined. The proof is by induction; assume we have determined all the values $D_{(w-i,i)}$, $0 \le i \le r$, for some r, $r \ge 2e-k$, and all vectors from configurations $(w-j,j)$, $0 \le j \le r-e$, are J-covered by codewords from configurations $(w-i,i)$, $0 \le i \le r$. To evaluate $D_{(w-r-1,r+1)}$ notice that by considering how vectors of configuration $(w-r+e-1, r-e+1)$ are J-covered we have

$$\binom{w}{r-e+1}\binom{n-w}{r-e+1} = \sum_{i=r-2e+1}^{r+1} C_{(w-i,i)}^{(w-r+e-1,r-e+1)} \cdot D_{(w-i,i)}$$

where $C_{(x_1,y_1)}^{(x_2,y_2)}$ is the number of vectors from configuration (x_2,y_2) which are J-covered by a codeword from configuration (x_1,y_1). Hence we have

$$D_{(w-r-1,r+1)} = \frac{\left[\binom{w}{r-e+1}\binom{n-w}{r-e+1} - \sum_{i=r-2e+1}^{r} C_{(w-i,i)}^{(w-r+e-1,r-e+1)} \cdot D_{(w-i,i)}\right]}{C_{(w-r-1,r+1)}^{(w-r+e-1,r-e+1)}}$$

and hence $D_{(w-r-1,r+1)}$ is determined, and all vectors from configurations $(w-j,j)$, $0 \le j \le r-e+1$, are J-covered by codewords from configurations $(w-i,i)$, $0 \le i \le r+1$.

Thus, there are exactly $e+1$ different configuration distributions for e-perfect codes. □

LEMMA 5.2. *In an e-perfect code in $J(2w+e+1,w)$ the intersection between any two codewords is at least e.*

Proof. Assume C is an e-perfect code in $J(2w+e+1,w)$, N is partitioned into two parts A and B such that $|A| = w$, $|B| = w+e+1$, and C contains the vector v, from configuration $(0,w)$, which ends with $e+1$ 0's as a codeword. The only vectors from configuration $(0,w)$ which are not J-covered by this codeword are the $\binom{w}{e+1}$ vectors which end with $e+1$ 1's. Since by Lemma 2.1 any codeword which J-covers some of these vectors should have H-distance at least $4e+2$ from v, it follows that these vectors are J-covered by codewords from configuration $(e, w-e)$, which end with $e+1$ 1's. Moreover, note that since $D_{(0,w)} = 1$ all the configuration distribution of C is determined (as in the proof of Theorem 5.1). Now, by Theorem 5.1, for this

configuration distribution we have $D_{(w-k,k)} = 1$ for exactly one k, $0 \leq k \leq e$, and for $i \neq k$, $0 \leq i \leq 2e-k$, $D_{(w-i,i)} = 0$. If $k < e$ we can exchange one column from A, with 1 in the codeword from configuration $(w-k,k)$, with a column from B, with 0's in the codewords from configurations $(w-k,k)$ and $(0,w)$. The obtained e-perfect code C' has $D_{(0,w)} = 1$ and $D_{(w-k-1,k+1)} = 1$, $k+1 \leq e$, a contradiction to the fact that $D_{(0,w)} = 1$ determines all the configuration distribution. Thus, $D_{(w-e,e)} = 1$ and $D_{(w-i,i)} = 0$ for $0 \leq i \leq e-1$. Now, assume that there exists a codeword from configuration $(w-k-r,\ k+r)$, $k < e$, $r > 0$, which intersects the codeword from configuration $(0,w)$ in exactly k positions. Again, we can exchange r columns from A, with 0's in the codeword from configuration $(w-k-r,\ k+r)$, with r columns from B with 1's in this codeword and 0's in the codeword from configuration $(0,w)$. The obtained e-perfect code C' have $D_{(0,w)} = 1$ and $D_{(w-k,k)} = 1$, for $k < e$, a contradiction. Now, note that any codeword can be chosen as the codeword from configuration $(0,w)$ and hence the intersection of any two codewords is at least e. □

THEOREM 5.3. *There is no e-perfect code in* $J(2w+e+1,w)$.

Proof. Assume C is an e-perfect code in $J(2w+e+1,w)$ and N is partitioned into two parts A and B such that $|A| = w$, $|B| = w+e+1$, and C contains the vector from configuration $(w,0)$ as a codeword. By Lemma 5.2 the intersection between any two codewords is at least e and hence C cannot contain a codeword from any configuration $(i,w-i)$, $0 \leq i \leq e-1$. Therefore, the $\binom{w+e+1}{w}$ vectors from configuration $(0,w)$ are J-covered only by codewords from configuration $(e,w-e)$. Moreover, every set of $e+1$ positions in the B part must be C-covered by the 0's of exactly one codeword from configuration $(e,w-e)$. Let C_1 be the set of codewords from configuration $(e,w-e)$. Thus, the complement of the B part of C_1 forms a Steiner system $S(e+1,2e+1,w+e+1)$.

Each set of e columns of the B part of C_1 has exactly $\frac{w+1}{e+1}$ rows with e 0's, since the complement of the B part of C_1 is an $S(e+1,2e+1,w+e+1)$. By exchanging any e columns from B with e columns of A that contain e 1's from the codewords of C_1 we obtain an e-perfect code C'. Since C has a codeword from configuration $(w,0)$ it follows that C' has a codeword from configuration $(w-e,e)$. Also, note that C' contains a codeword from configuration $(0,w)$. Clearly, not all e columns of A have e 1's in the codewords of C_1. If we exchange any e columns of B with e columns of A that do not contain e 1's in the codewords of C_1 we obtain a code C'' with a codeword from configuration $(w-e,e)$ but no codeword from configuration $(0,w)$. This is in contradiction to Theorem 5.1 that all codes with a codeword from configuration $(w-e,e)$ have the same configuration distribution.

Thus, there is no e-perfect code in $J(2w+e+1,w)$. □

Another interesting consequence from Theorem 5.1 is on the structure of e-perfect codes in $J(2w,w)$. We show now that these codes, if they exist, are *self-complement*; i.e., the complement of the code is equal to the code.

THEOREM 5.4. *An e-perfect code in* $J(2w,w)$ *is self-complement.*

Proof. Let C be an e-perfect code in $J(2w,w)$, and assume N is partitioned into two parts A and B such that $|A| = |B| = w$ and the vector from configuration $(w,0)$ is a codeword. By Theorem 5.1 for exactly one k, $0 \leq k \leq e$, we have $D_{(k,w-k)} = 1$ and for $i \neq k$, $0 \leq i \leq 2e-k$, $D_{(i,w-i)} = 0$. If $k > 0$ then we can exchange one column from A with 0 in the codeword from configuration $(k,w-k)$ with a column from B with 0 in the codeword from configuration $(k,w-k)$ to obtain a new e-perfect code C'. In C' we have $D_{(w-1,1)} = 1$ and $D_{(k,w-k)} = 1$ in contradiction to the unique configuration distribution when $D_{(k,w-k)} = 1$, $0 \leq k \leq e$, obtained in Theorem 5.1.

Thus, $k = 0$ and C is self-complement. □

6. Applications. The theorems obtained in §§3, 4, and 5 make it possible to reduce the range in which perfect codes in the Johnson scheme can exist. If an e-perfect code in $J(n, w)$ exists then by Theorem 3.1 and Corollary 3.2 Steiner systems $S(e+1, 2e+1, w)$ and $S(e+1, 2e+1, n-w)$ exist. By the divisibility conditions of Theorem 2.3 we have that $e+1$ should divide $w-e$ and $n-w-e$ and hence we have $w \equiv e \pmod{e+1}$. This condition itself limits the range in which e-perfect codes can exist. Combining this condition with the nonexistence of e-perfect codes in $J(2w+e+1, w)$ obtained in Theorem 5.3 we have the following theorem.

THEOREM 6.1. *There are no perfect codes in $J(2w+p, w)$, p prime.*

In fact, we can obtain many more results similar to Theorem 6.1; e.g., there are no perfect codes in $J(2w+2p, w)$, p prime, $p \neq 3$ or there are no perfect codes in $J(2w+3p, w)$, p prime, $p \neq 2$, $p \neq 3$, and $p \neq 5$, and other similar theorems. The proofs involve carefully examining the divisibility conditions of Theorem 2.3 for $S(e+1, 2e+1, w)$ and $S(e+1, 2e+1, n-w)$, and using Theorem 5.3. Theorem 6.1 immediately implies the results of Bannai [1] and Hammond [6], that there are no nontrivial perfect codes in $J(2w-2, w)$, $J(2w-1, w)$, $J(2w+1, w)$, and $J(2w+2, w)$. Together with the other divisibility conditions we have that the range is considerably reduced. By Theorem 4.2 we know that a nontrivial e-perfect code cannot be a Steiner system. Hence, we can apply Corollary 3.4 and obtain that if there exists an e-perfect code in $J(n, w)$ then there exists a Steiner system $S(2, 2e+1-k, w+2e-k)$ for some k, $0 \leq k \leq e-1$. This implies that C is an e-perfect code in $J(n, w)$ if w is an admissible value for the necessary conditions for the existence of $S(e+1, 2e+1, w)$, and also an admissible value for the necessary conditions of one Steiner system $S(2, 2e+1-k, w+2e-k)$. The same results are applied also on $n-w$ instead of w. Checking all these conditions we obtain that for $e = 1$, we must have $n-w \equiv w \equiv 1 \pmod 6$, for $e = 2$, we have $n-w \equiv w \equiv 2,\ 17,\ 26,\ 41$, or $50 \pmod{60}$, and so on. Compiling all this data, we also have found that there are no nontrivial perfect codes in $J(2w-r, w)$ and $J(2w+r, w)$ for all $1 \leq r \leq 14$ with possible exceptions for $r = 6$, 9, and 12. This comes together with modulos conditions imposed on w and $n-w$ for any e-perfect code in $J(n, w)$. Assume N is partitioned into two subsets A and B, such that $|A| = w$, $|B| = n-w$, and the vector of the $(w, 0)$ configuration is a codeword. By considering the way in which the vectors of the configuration $(w-e-2, e+2)$ are J-covered we can get some more divisibility conditions that rule out some of the combinations between w and $n-w$. We can proceed to get more divisibility conditions. But no other Johnson graph $J(n, w)$ was ruled out as a candidate to contain nontrivial perfect codes. Similarly, other results can be obtained from the configuration distribution, but the outcome is less significant.

From the proof of Theorem 3.1 we can see the involved structure of e-perfect codes in $J(n, w)$. If we partition N into two subsets A and B, such that $|A| = w$ and $|B| = n-w$ and the vector of the $(w, 0)$ configuration is a codeword, then in the codewords of configuration $(w-2e-1, 2e+1)$ we have in the A part a complement of a Steiner system $S(e+1, 2e+1, w)$ for each set of codewords which C-cover any fixed $e+1$ positions of B. Similarly, we can obtain (and this can be an alternative proof for Corollary 3.2) that in the B part there is a Steiner system $S(e+1, 2e+1, n-w)$ for each set of codewords for which the complements C-cover any fixed $e+1$ positions in A. This involved structure, together with the Steiner system of Theorem 3.3 and Corollary 3.4, seems to be impossible to achieve. This is without taking into consideration all the other configurations which are becoming more and more complicated. Hence, we

get more confidence in the belief in the conjecture that no nontrivial perfect codes exist in the Johnson scheme.

Acknowledgment. The author would like to thank Prof. Chris Mitchell for his hospitality in Royal Holloway College, which made it possible to obtain these results.

REFERENCES

[1] E. BANNAI, *Codes in bi-partite distance-regular graphs*, J. London Math. Soc., 2 (1977), pp. 197–202.

[2] A. E. BROUWER, J. B. SHEARER, N. J. A. SLOANE, AND W. D. SMITH, *A new table of constant weight codes*, IEEE Trans. Inform. Theory, IT-36 (1990), pp. 1334–1380.

[3] P. DELSARTE, *An algebraic approach to association schemes of coding theory*, Philips J. Res., 10 (1973).

[4] T. ETZION AND A. VARDY, *Perfect codes: Constructions, properties and enumeration*, IEEE Trans. Inform. Theory, IT-40 (1994), pp. 754–763.

[5] S. W. GOLOMB AND L. R. WELCH, *Perfect codes in the lee metric and the packing of polyominos*, SIAM J. Appl. Math, 18 (1970), pp. 302–317.

[6] P. HAMMOND, *On the non-existence of perfect and nearly perfect codes*, Discrete Math., 39 (1982), pp. 105–109.

[7] F. J. MACWILLIAMS AND N. J. A. SLOANE, *The Theory of Error Correcting Codes*, North-Holland, Amsterdam, 1977.

[8] C. ROOS, *A note on the existence of perfect constant weight codes*, Discrete Math., 47 (1983), pp. 121–123.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 210–224, May 1996

005

A GRAPH-COLORING RESULT AND ITS CONSEQUENCES FOR POLYGON-GUARDING PROBLEMS*

FRANK HOFFMANN† AND KLAUS KRIEGEL†

Abstract. The following graph-coloring result is proved: let G be a 2-connected, bipartite, and plane graph. Then one can triangulate G in such a way that the resulting graph is 3-colorable. Such a triangulation can be computed in $O(n^2)$ time. This result implies several new upper bounds for polygon guarding problems, including the first nontrivial upper bound for the rectilinear prison yard problem. (1) $\lfloor \frac{n}{3} \rfloor$ vertex guards are sufficient to watch the interior of a rectilinear polygon with holes. (2) $\lfloor \frac{5n}{12} \rfloor + 3$ vertex guards ($\lfloor \frac{n+4}{3} \rfloor$ point guards) are sufficient to simultaneously watch both the interior and exterior of a rectilinear polygon. Moreover, a new lower bound of $\lfloor \frac{5n}{16} \rfloor$ vertex guards for the rectilinear prison yard problem is shown and proved to be asymptotically tight for the class of orthoconvex polygons.

Key words. graph coloring, visibility in polygons

AMS subject classifications. 05C15, 52A45

1. Introduction.

The original art gallery problem raised by Klee asks how many guards are sufficient to watch the interior of an n-sided simple polygon. In 1975, Chvátal [2] gave the answer, proving that $\lfloor \frac{n}{3} \rfloor$ guards are always sufficient and sometimes necessary. Since then many results have been published studying variants of the problem or analyzing algorithmic aspects; see [12], [13], [15] for a detailed discussion. All notations not explicitly defined below are used as in [12].

One of the main questions still open in this field is the so-called prison yard problem for simple rectilinear polygons (see [13]);[1] i.e., one wants to determine the minimal number of vertex guards sufficient to simultaneously watch both the interior and exterior of any n-sided simple rectilinear polygon.

The prison yard problem for general simple polygons has been completely settled by Füredi and Kleitman, proving that $\lceil \frac{n}{2} \rceil$ vertex guards for convex and $\lfloor \frac{n}{2} \rfloor$ vertex guards for any nonconvex simple polygon are sufficient; see [4]. As mentioned in [4] this does not imply new bounds for the rectilinear case. Here, the only upper bound known has been the rather trivial $\lfloor \frac{7n}{16} \rfloor + 5$-bound (see [12]) which can be obtained by combining the $\lfloor \frac{n}{4} \rfloor$-result for the interior (see [5]) with the $\lceil \frac{n}{4} \rceil + 1$ vertex guards for the exterior of an n-sided rectilinear polygon.

We are going to derive several new bounds for the original rectilinear prison yard problem as well as for the stronger "prison problem," where the guards have to watch not only the inside and outside of the yard but also all cells of the prison. The key tools to prove the new upper bounds are coloring and multicoloring arguments. The new graph coloring result shown in §2 is probably also of some independent interest. It shows that one can triangulate a 2-connected, bipartite, and plane graph in such a way that the resulting graph is 3-colorable. Further, we give a characterization of all triangulations leading to 3-colorable graphs. An $O(n^2)$ time algorithm which generates such triangulations is sketched.

* Received by the editors March 30, 1994; accepted for publication (in revised form) May 15, 1995. A preliminary version of this paper appeared in *Lecture Notes in Comput. Sci.*, 762 (1993), pp. 78–87. This research was supported by the ESPRIT Basic Research Action project ALCOM II.

† Institut für Informatik, Freie Universität Berlin, Takustrasse 9, D-14195 Berlin, Germany (hoffmann@inf.fu-berlin.de and kriegel@inf.fu-berlin.de).

[1] Rectilinear polygons have also been called orthogonal and isothetic.

In §3 we apply this 3-coloring result to guarding problems by a suitable modelling of rectilinear polygons. In §4 we establish lower bounds for the vertex guard number in staircase-like and in orthoconvex prison yards. Then in §5 we use a new multicoloring technique to prove these bounds to be asymptotically tight for the described polygon classes. Table 1 summarizes upper and lower bounds on guard numbers for rectilinear polygons. Compare with [12] for previous bounds.

TABLE 1.

Polygon type	Problem	Guard type	Upper bound		Lower bound
			Previous	New	
with holes	*art gallery*	*vertex*	$\lfloor \frac{3n}{8} \rfloor$	$\lfloor \frac{n}{3} \rfloor$	$\lfloor \frac{2n}{7} \rfloor$
staircase	*prison yard*	*vertex*	–	$\lfloor \frac{3n}{10} \rfloor + 2$	$\lfloor \frac{3n}{10} \rfloor$
orthoconvex	*prison yard*	*vertex*	–	$\lfloor \frac{5n}{16} \rfloor + 2$	$\lfloor \frac{5n}{16} \rfloor$
simple	*prison yard*	*vertex*	$\lfloor \frac{7n}{16} \rfloor + 5$	$\lfloor \frac{5n}{12} \rfloor + 2$	$\lfloor \frac{5n}{16} \rfloor$
simple	*prison yard*	*point*	$\lfloor \frac{7n}{16} \rfloor + 5$	$\lfloor \frac{n+4}{3} \rfloor$	$\lceil \frac{n}{4} \rceil + 1$
with h holes	*prison*	*vertex*	–	$\lfloor \frac{5n-4h}{12} \rfloor + 2$	–

At first glance some of the new bounds such as $\frac{5n}{16}$ seem to be curious. One advantage of our paper is that we provide rather natural explanations for these fractional numbers. We conclude in §6 by discussing algorithmic aspects and posing a few related questions.

2. A result on 3-colorable plane graphs. We assume that the reader is familiar with basic definitions and facts about planar graphs. For more details see, e.g., [11]. Let $G = (V, E)$ be a 2-connected, plane (i.e., embedded planar) graph. The embedding in the plane of G determines a set of faces including an exterior face. It is well known that each face boundary of G corresponds to a directed cycle such that the face lies on the left side of it. A *diagonal* is an edge which does not belong to E and connects two vertices on a facial cycle. G is *triangulated* if it has no diagonals; i.e., all facial cycles have length 3.

A *triangulation* of a plane graph is the augmentation of a set of diagonals such that the resulting *triangulation graph* is plane and triangulated. In this section we are going to prove and discuss the following result.

THEOREM 2.1. *Let G be a plane, 2-connected, and bipartite graph. Then there exists a triangulation of G such that the triangulation graph is 3-colorable.*

The proof consists of two lemmas. The first one is due to Whitney and can be proved by standard induction arguments. For an elegant proof see [8].

LEMMA 2.2. *A triangulated plane graph is 3-colorable iff all vertices have even degree.*

A triangulation of a plane graph G will be called *even* if in the triangulation graph each vertex has even degree.

LEMMA 2.3. *Let $G = (V, E)$ be a plane, 2-connected, and bipartite graph. Then G has an even triangulation.*

Proof. Since G is bipartite, all its facial cycles have even length. By adding diagonals (if necessary) we can assume, without loss of generality, that all facial cycles are cycles of length 4. Each graph of this type has a straight-line embedding in the plane such that each interior face is a convex quadrilateral and the exterior face is the

complement of a convex quadrilateral. Therefore we will denote by Q the set of all faces of G. Further, for any face $q \in Q$, the set of the four vertices on the boundary cycle of q is denoted by V_q and we set $Q_v = \{q \in Q | v \in V_q\}$. We fix a 2-coloring c of G with colors 0 and 1.

For a face $q \in Q$, the diagonal joining the two 0-colored vertices in V_q is called *0-diagonal*; the other diagonal joining the two 1-colored vertices in V_q is the *1-diagonal*. Now, a triangulation of G is nothing other than choosing for each face either the 0-diagonal or the 1-diagonal and adding it to the graph. We note that a diagonal of the exterior face cannot be drawn as a straight line within the given plane embedding of G. The graph obtained, however, is planar and thus there is also a straight-line embedding of it.

More formally, let us introduce for each $q \in Q$ a $\{0,1\}$-valued variable x_q. We fix a 1–1 correspondence between all triangulations of G and all evaluations of the variables x_q via the following condition for a triangulation T:

$$\forall q \in Q\text{:} \quad T \text{ contains the 0-diagonal of } q \iff x_q = 1.$$

The following observation is straightforward: if $c(v)$ is the color of a vertex $v \in V_q$, then the term $x_q + c(v) \pmod 2$ describes the increase of the degree of v by the diagonal of q which is chosen with respect to the value of x_q. Throughout this proof all additions should be understood as additions mod 2. The existence of an even triangulation is equivalent to the condition that the following system of equations has a solution in $GF(2)$:

$$deg(v) + \sum_{q \in Q_v} (x_q + c(v)) \;=\; 0 \qquad (\forall v \in V)$$

or, equivalently,

$$\textbf{(*)} \qquad \sum_{q \in Q_v} x_q \;=\; deg(v) + |Q_v| c(v) \qquad (\forall v \in V).$$

We want to illustrate the proof by the example in Figure 1. It shows the Schlegel diagram of a cube with a 0/1-coloring of its vertices. The six faces of the graph have variables $x_1, \ldots, x_6$. The dotted lines represent exactly those diagonals which correspond to the indicated evaluation of the variables. It is easy to check that the represented triangulation is even and that, indeed, **(*)** is satisfied.

Note that if $|V| = n$ then by Euler's formula $|Q| = n - 2$. Thus the system **(*)** is overdetermined and has an $n \times (n-2)$ matrix $\mathbf{A}$ of coefficients. The system has a solution iff the rank of $\mathbf{A}$ is equal to the rank of the augmented matrix $\overline{\mathbf{A}}$; see, e.g., [16]. Here, augmentation means attaching the vector of constants from the right side of **(*)** as an additional column to $\mathbf{A}$.

Recall that the rank of a matrix is equal to the maximum number of linear independent rows. Therefore it is sufficient to show that any linear dependence of rows in $\mathbf{A}$ implies a linear dependence of the corresponding rows in $\overline{\mathbf{A}}$. Since over $GF(2)$ sums are the only possible linear combinations, we have to prove the following.

CLAIM. *If for some set of vertices $U \subseteq V$ it holds that $\sum_{v \in U} \sum_{q \in Q_v} x_q \equiv 0$, then*

$$\sum_{v \in U} (deg(v) + |Q_v| c(v)) = 0.$$

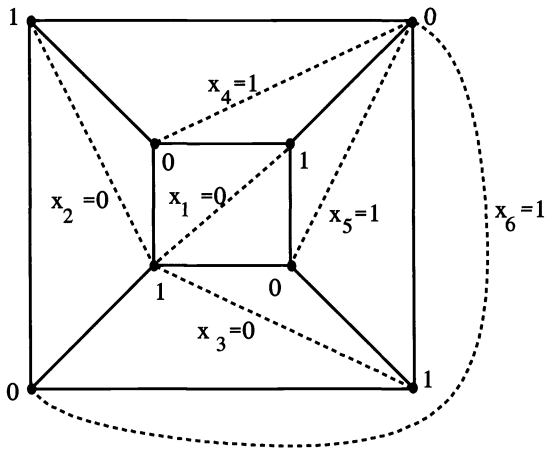


FIG. 1. *An even triangulation of the Schlegel diagram of the cube.*

Here the symbol $\equiv$ in the first sum means that each variable x_q occurs an even number of times in equations corresponding to vertices from U.

Since x_q can only occur in the four equations corresponding to vertices in V_q, it follows from the assumption of the claim that for any $q \in Q$ the number $|V_q \cap U|$ is $0, 2$, or 4. So we can subdivide Q according to this cardinality into Q_0, Q_2, and Q_4.

Let us return to our example in Figure 1. Suppose the set U in the claim is the set of the four inner vertices. Then Q_4 consists only of the face in the middle of the drawing; the other four interior faces form the set Q_2, and the only element of Q_0 is the exterior face.

Now we prove the claim by showing that the sums $\Sigma_1 = \sum_{v \in U} deg(v)$ and $\Sigma_2 = \sum_{v \in U} |Q_v| c(v)$ are both zero.

1) Since G is planar and 2-connected, we know that the degree of a vertex v equals the cardinality of the set Q_v. Using this fact and changing the order of summation we get

$$\Sigma_1 = \sum_{v \in U} |Q_v| = \sum_{v \in U} \sum_{q \in Q_v} 1 = \sum_{q \in Q} \sum_{v \in V_q \cap U} 1 = \sum_{q \in Q} |V_q \cap U|.$$

As we have already mentioned, all summands are even numbers and consequently $\Sigma_1 = 0$.

2) We start as above changing the order of summation.

$$\Sigma_2 = \sum_{v \in U} |Q_v| c(v) = \sum_{v \in U} \sum_{q \in Q_v} c(v) = \sum_{q \in Q} \sum_{v \in V_q \cap U} c(v).$$

Now we split the sum into subsums over Q_0, Q_2, and Q_4. Moreover, we subdivide the subsum over Q_2 according to whether the two vertices in $V_q \cap U$ lie on a diagonal or on an edge of q. So we have

$$\Sigma_2 = \sum_{q \in Q_0} \sum_{v \in V_q \cap U} c(v) + \sum_{q \in Q_2^{diag}} \sum_{v \in V_q \cap U} c(v) + \sum_{q \in Q_2^{edge}} \sum_{v \in V_q \cap U} c(v) + \sum_{q \in Q_4} \sum_{v \in V_q \cap U} c(v).$$

Obviously the first sum is zero and can be deleted. We also delete the sum over Q_2^{diag} since any summand of it has either the form $1+1$ or $0+0$. Finally, in the sum over Q_4, each summand has the form $1+0+1+0$. Deleting this sum also we obtain

$$\Sigma_2 = \sum_{q \in Q_2^{edge}} \sum_{v \in V_q \cap U} c(v) = \sum_{q \in Q_2^{edge}} (1+0) = |Q_2^{edge}| \pmod 2.$$

It remains to show that $|Q_2^{edge}|$ is even. Consider the subgraph of G induced by U. Replacing any (undirected) edge in it by a pair of oppositely directed edges we obtain a set $\vec{E_U}$ of even cardinality. Recall that a face $q \in Q$ is uniquely determined by its directed facial cycle. We denote the set of the four directed edges on the cycle by $\vec{E_q}$. Since $|V_q \cap U|$ is even, the number $|\vec{E_q} \cap \vec{E_U}|$ is either 4 (iff $q \in Q_4$) or 1 (iff $q \in Q_2^{edge}$), or otherwise 0. Let us cancel from $\vec{E_U}$ all directed edges which arise from some $\vec{E_q}$ such that $|\vec{E_q} \cap \vec{E_U}| = 4$. Clearly, the set obtained in this way is of even cardinality. Each directed edge in this set determines a face $q \in Q_2^{edge}$ and vice versa. This completes the proof. □

To make the last step in the proof more transparent let us once more return to Figure 1, with U being the set of the four inner vertices. Then Q_2^{edge} consists of the four interior faces which surround the middle face and $\vec{E_U}$ consists of eight directed edges obtained by traversing the middle cycle in both directions. Since the middle face has four vertices with U in common, we have to cancel its facial cycle, i.e., the counterclockwise-directed cycle. Now it is easy to observe that each of the remaining directed edges uniquely determines one of the faces from Q_2^{edge}.

The remaining part of this section is devoted to the combinatorial characterization of all even triangulations and to algorithmic aspects of Theorem 2.1. Since both topics are not essential for the rest of the paper, we only sketch them.

It is easy to find graphs with more than one even triangulation. Let us consider the even triangulation given in Figure 1. If we flip the diagonals in the four faces which surround the middle face, then the new triangulation is even too.

More generally, let us assume that $G = (V, E)$ is a 2-connected, bipartite, and plane graph such that all its facial cycles have length 4. Then the dual graph G^* is 4-regular. If we start a walk in this graph, at any time we have four possibilities for the next step: to go left, straight ahead, right, or back. A closed walk consisting of straight steps only is called a *straight walk* or *S-walk*. Note that a face either occurs on two different S-walks or it occurs twice on a single S-walk. For the Schlegel diagram of the cube we have three S-walks, each of them being a cycle. The graph in Figure 2 shows an S-walk which is not a cycle because it crosses itself.

Suppose an even triangulation of G is given. If we run once around an S-walk flipping the diagonal in each face visited (if a face is visited twice the diagonal will be the same in the end), then the triangulation obtained this way is even. The next result shows that the converse is also true.

THEOREM 2.4. *If T and T' are even triangulations of G, then there is a collection of S-walks such that by flipping the diagonals of T along these walks we obtain T'.*

We can prove this theorem by formalizing the problem as a system of linear equations over $GF(2)$. The consistency of the system can be shown by some counting arguments. However, the underlying combinatorial arguments are much more involved than those in the proof of Lemma 2.3 and we have omitted them. Details are in [7].

Using Theorem 2.4 we are able to determine a subset $Q_{val} \subset Q$ with the property that for each choice of diagonals for Q_{val} there is exactly one even triangulation of

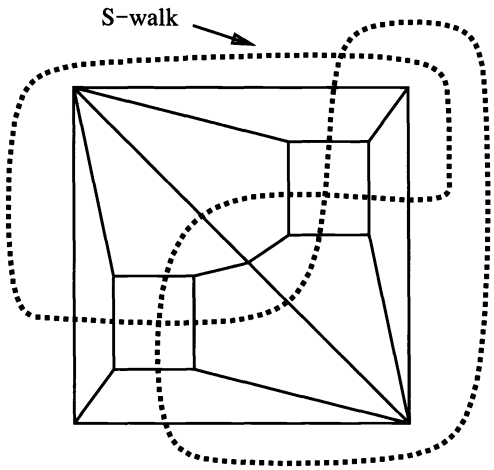


FIG. 2. *Example of an S-walk.*

the whole graph which contains the chosen diagonals. In terms of the system (*), this means that a given evaluation of the variables $\{x_q | q \in Q_{val}\}$ can be extended to a solution of (*) in a unique way only.

Finally, we briefly discuss the problem of algorithmically finding an even triangulation more efficiently than by using a general (superquadratic) method for solving the linear system of equations (*). We have shown that the system has a solution. Its underlying structure is a planar graph, so we could hope that the generalized nested dissection technique of Lipton, Rose, and Tarjan (see [10]) applies. This would imply an $O(n^{3/2})$-algorithm. However, this special version of the Gaussian elimination method requires positive definite, symmetric systems. In contrast, our system is not symmetric (even the matrix of the system is not quadratic) and the solution is, in general, not unique. By adding dummy variables and dummy equations we could transform the system into a symmetric one, but this would enlarge the solution space and hence there is no chance of obtaining a positive definite system in this way.

On the other hand, we can fix an evaluation of the variables $\{x_q | q \in Q_{val}\}$ to obtain a more restricted system with a unique solution. This new system has the disadvantage of being highly nonsymmetric. However, we can now adapt some ideas from the nested dissection approach to it, as outlined below.

We start with a suitable numbering of the faces which can be obtained by applying the vertex numbering algorithm from [10] to the dual graph G^*. This algorithm uses the planar separator theorem recursively and runs in $O(n \log n)$ time. Based on this numbering we can design a substitution scheme with the following properties:

- any substitution is applied to at most $O(\sqrt{n})$ equations;
- the length of each substitution is bounded by $O(\sqrt{n})$.

The first point follows immediately from the properties of the face numbering, whereas the second point requires a more involved machinery including Theorem 2.4 and the construction of the set Q_{val}. We reach the following result.

THEOREM 2.5. *An even triangulation of a 2-connected, bipartite, and plane graph can be computed in $O(n^2)$ time.*

Once again, the complete algorithm together with its running time analysis can be found in [7].

3. A graph model for the prison yard problem and general upper bounds. The idea for our graph model is based on the following nice and simple proof of the classical art gallery theorem of Fisk [3]: consider an arbitrary geometric triangulation of a given simple polygon. It is well known that the graph formed by all polygon edges and all diagonals in the triangulation is 3-colorable. Clearly, any triangle of the 3-colored graph contains each color. Therefore, choosing guard positions corresponding to the smallest color class implies that $\leq \lfloor \frac{n}{3} \rfloor$ vertex guards can watch the polygon.

In [5] Kahn, Klawe, and Kleitman applied a similar idea to rectilinear polygons. They proved that any rectilinear polygon (possibly with holes) has a convex quadrilateralization, i.e., a decomposition into convex 4-gons (subsequently called *quadrilaterals*) using only diagonals (called *chords*) of the polygon. Moreover, it is easy to observe that for simple rectilinear polygons the graph consisting of all polygon edges, all chords, and both inner diagonals of all quadrilaterals is 4-colorable. Hence they obtained an $\lfloor \frac{n}{4} \rfloor$-upper bound for the rectilinear art gallery problem.

However, the argument cannot be applied in the case of rectilinear polygons with holes since the graphs are, in general, no longer 4-colorable. But in this situation, Theorem 2.1 now states that we can select one diagonal per quadrilateral such that the graph formed by all polygon edges, all chords, and the selected diagonals is 3-colorable. Again, since each quadrilateral contains each of the three colors, we have proved the following result.

THEOREM 3.1. $\lfloor \frac{n}{3} \rfloor$ *vertex guards are sufficient to solve the art gallery problem for rectilinear polygons with holes.*

We note that this is an improvement on the previously known $\frac{3n}{8}$-bound, which was obtained by converting a polygon with holes into a 1-connected one by adding h edges ($2h$ new vertices) [12] and then applying the $\frac{n}{4}$-result of [5]. Observe that the guards also watch the interior of the holes if we start from quadrilateralized holes.

As shown below, we can modify this graph model in a way which allows us to apply our 3-coloring result to prison yard-type problems also. Let P be a simple n-sided rectilinear polygon. Without loss of generality we can assume that it is in general position; see [5], [12].

We start by constructing its orthoconvex hull $C(P)$, i.e., the smallest point set containing P such that its intersection with any horizontal or vertical line is convex (see Figure 3a). The boundary of $C(P)$ partitions the exterior region of P into the exterior region of $C(P)$ and those connected components of $C(P) \setminus P$ which are different from the interior region of P. These components will be called *pockets*. Since all pockets are rectilinear polygons, they can be quadrilateralized just as well as we quadrilateralize the polygon P itself. The hull construction requires the insertion of some additional vertices (u in our example in Figure 3a). However, using an idea from [9] we can shift these vertices to neighboring polygon corners on the boundary of $C(P)$ in such a way that the resulting polygon $C^*(P)$ is also orthoconvex and the quadrilateralizability of the pockets is not destroyed (see the dashed line in Figure 3a).

Clearly, $C^*(P)$ is bounded by four *extremal* edges (northernmost, westernmost, southernmost, easternmost) which are cyclically connected by monotone staircases. So the exterior of $C^*(P)$ can be covered by four halfplanes defined by the extremal edges and the *cones* defined by all concave vertices on the staircases. Such a cone is

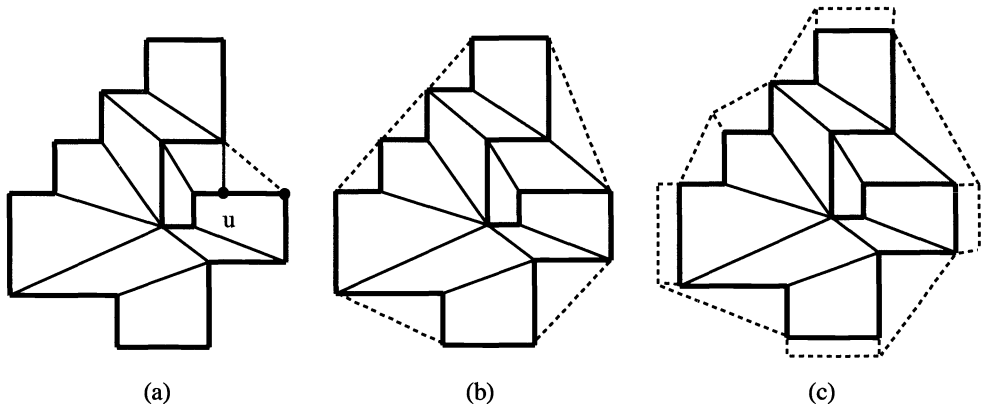


FIG. 3. *Modelling the prison yard problem.*

exactly the set of points in the exterior visible from the vertex.

Let $G(P)$ be the following planar graph (Figure 3b). Its vertex set is the set of all polygon vertices, its edge set consists of all polygon edges, all quadrilateralization chords, and edges connecting pairs of consecutive convex corners (separated by a concave corner) on boundary staircases in $C^*(P)$.

We say that a subset D of the vertex set *dominates* $G(P)$ if each quadrilateral, each triangle over a staircase, and each of the four extremal edges contains at least one vertex from D. In this context the prison yard problem now reads as follows: find a small dominating set for $G(P)$.

The idea is to find a dominating set by applying Theorem 2.1. It is therefore necessary to modify $G(P)$ such that it becomes bipartite. In particular, all convex regions (also the exterior cones and halfplanes!) will be represented by convex quadrilaterals. This can be done by inserting additional vertices and edges.

We start as before by constructing $C^*(P)$. Then we use eight new vertices to obtain a copy of each extremal edge as indicated in Figure 3c. Finally, for any monotone boundary staircase of $C^*(P)$ which contains more than one convex vertex (we do not count the vertices on extremal edges), we copy every second one of them. This allows us to replace each boundary triangle in $G(P)$ by a quadrilateral in the new graph $G^*(P)$; compare Figure 3c. The number of additionally inserted vertices is bounded by $\lfloor \frac{n-12}{4} \rfloor + 8$. Therefore the total number of vertices $G^*(P)$ is bounded by $\lfloor \frac{5n}{4} \rfloor + 5$.

By Theorem 2.1 we can 3-color the graph. Each color class is a dominating vertex set for $G^*(P)$ and we choose the smallest one. This set may possibly contain vertices which are not vertices of P; i.e., newly inserted vertices are chosen as guard positions. But, obviously, these guards can be shifted onto the corresponding original polygon vertices and we obtain a dominating set for $G(P)$. Therefore we have shown the following result.

THEOREM 3.2. *For any simple rectilinear polygon on n vertices, $\lfloor \frac{5n}{12} \rfloor + 2$ vertex guards are sufficient to solve the prison yard problem.*

In fact we observe that we can easily incorporate polygons with holes into the above construction. We define the *prison problem* as follows: let a rectilinear polygon

P with h rectilinear holes $P_1, \ldots, P_h$ be given, having in total of n vertices. The prison problem is to select a set of vertex guards (or point guards) such that any point in the plane can be watched from one of the selected vertices (points). That is, we want to simultaneously watch the exterior of the polygon, its interior, and the interior of all its holes with the polygon edges being obstacles for visibility.

A graph $G^*(P, P_1, \ldots, P_h)$ modelling the prison problem can be constructed as follows:

(1) quadrilateralize the holes $P_1, \ldots, P_h$;

(2) quadrilateralize the interior of P;

(3) proceed with the exterior of P as in the construction of $G^*(P)$.

Clearly, P has at most $n-4h$ vertices and hence the number of additional vertices for the construction of $G^*(P)$ is bounded by $8 + \lfloor \frac{n-4h-12}{4} \rfloor$. Thus, $G^*(P, P_1, \ldots, P_h)$ has at most $\lfloor \frac{5n-4h}{4} \rfloor + 5$ vertices.

COROLLARY 3.3. *Let P be a rectilinear polygon with h holes on n vertices.*

(i) $\lfloor \frac{5n-4h}{12} \rfloor + 2$ *vertex guards are sufficient to solve the prison problem.*

(ii) $\lfloor \frac{n+4}{3} \rfloor$ *point guards are sufficient to solve the prison problem for P.*

Proof. (i) Apply Theorem 2.1 to the graph $G^*(P, P_1, \ldots, P_h)$.

(ii) Let R be a rectangle enclosing P. We consider R together with P as a polygon P' having P as a hole. After quadrilateralizing P' as well as the original P together with all its holes the resulting graph has $n+4$ vertices and fulfills the assumptions of Theorem 2.1. □

Observe that in Corollary (ii) we get at most two point guards not sitting on polygon vertices. The best upper bound for point guards (even in the case $h = 0$) until now has been the same as for the vertex guard version: $\lfloor \frac{7n}{16} \rfloor + 5$. The spiral polygon in this case gives an $(\lceil \frac{n}{4} \rceil + 1)$-lower bound; see [12].

4. Lower bounds for the prison yard problem. Any simple convex nonrectilinear polygon requires $\lceil \frac{n}{2} \rceil$ vertex guards to solve the prison yard problem. What are the candidates for lower bound examples in the rectilinear world?

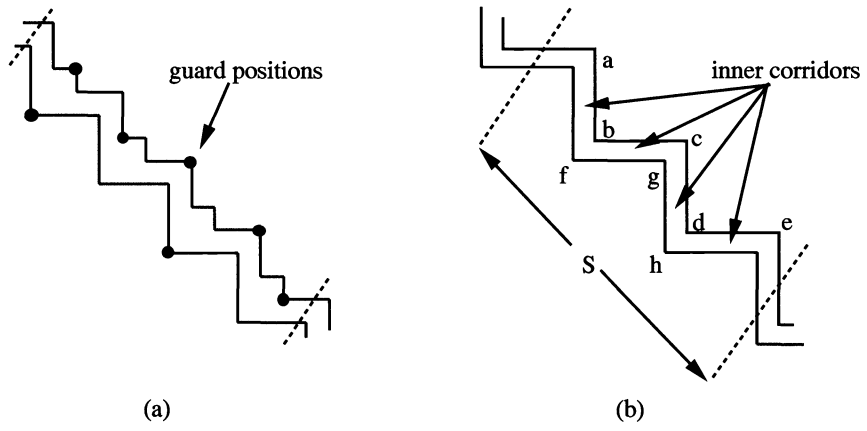


FIG. 4. *Dorward's example and lower bound illustration.*

Figure 4a shows an example of a rectilinear polygon due to Dorward; see [13]. Periodically repeating the guarding positions indicated in Figure 4a we see that

$\lceil \frac{7n}{24} \rceil + 2$ watchmen are sufficient, which is an improvement on the $\lceil \frac{n}{3} \rceil$ guarding in [13]. Let P_0 be the simplest possible staircase polygon; see Figure 4b.

PROPOSITION 4.1. *The prison yard P_0 requires $\lceil \frac{3n}{10} \rceil$ vertex guards.*

Proof. Consider a segment S on 10 vertices as indicated in Figure 4b. Assume that two guards are sufficient for S. To watch the triangles abc, cde, and fgh there must be one guard sitting in c and one in f, g, or h. There are also four inner corridors in S to be watched. This is impossible with one of the two guards sitting in c. Finally, vertex guards placed outside S cannot help watch these three triangles and the four inner corridors inside S. □

In the next section we will show that $\lceil \frac{3n}{10} \rceil + 2$ vertex guards are also sufficient for any staircase polygon. Surprisingly, there are other orthoconvex polygons which require even more guards.

Let P_1 be the pyramid in Figure 5. Recall that a horizontal pyramid is a rectilinear polygon with a horizontal edge (bottom edge) the length of which equals the sum of the lengths of all other horizontal edges; see [12]. We can assume that its edge lengths are chosen in such a way that to watch an inner quadrilateral (the quadrilateralization is unique!) one has to choose one of its vertices as the guard position. $\frac{5n}{16}$ guards are sufficient (up to an additive constant) by periodically repeating the 10 guard positions (on a segment of 32 vertices) as indicated in Figure 5.

PROPOSITION 4.2. *The prison yard P_1 requires $\lceil \frac{5n-10}{16} \rceil$ vertex guards.*

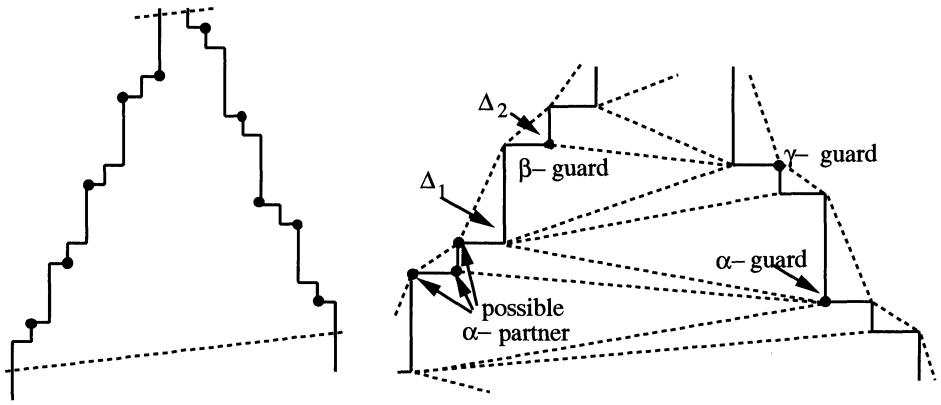


FIG. 5. $\frac{5n}{16}$ *vertex guards are necessary.*

Proof. Assume that g guards solve the problem. We can distinguish three types of guards; see Figure 5. A guard stationed in a concave corner such that he can watch four inner quadrilaterals is called an α-guard. Observe that each such guard must have one "partner" on the other side watching the opposite $\triangle_2$-triangle. Let us call two such guards an α-pair. An α-guard pair watches altogether at most four inner quadrilaterals, two $\triangle_1$-triangles, and one $\triangle_2$-triangle.

Among all other guards who are not part of α-pairs we distinguish between β- and γ-guards. β-guards are sitting in concave corners. They each watch two quadrilaterals, zero $\triangle_1$-triangles, and one $\triangle_2$-triangle. Finally, γ-guards sit in convex corners and each of them watches one quadrilateral, one $\triangle_1$-triangle, and one $\triangle_2$-triangle. Each guard is either a β- or γ-guard or belongs to an α-pair. We know that our guarding

set consists of $g = 2a + b + c$ guards of α-, β-, and γ-type. In total there are $\frac{n-2}{2}$ quadrilaterals and $\frac{n-2}{4}$ triangles of each type in P_1. Since we have assumed that the guards solve the problem, we have

$$4a + 2b + c \geq (n-2)/2, \tag{1}$$

$$2a + c \geq (n-2)/4, \tag{2}$$

$$a + b + c \geq (n-2)/4. \tag{3}$$

Multiplying (3) by 2 and adding (1) and (2) gives $4g \geq \frac{5(n-2)}{4}$, but this implies the lower bound. □

5. Special upper bounds. In this section we show that the lower bounds derived for rectilinear staircase polygons and for orthoconvex rectilinear polygons are tight up to an additive constant. To this end we extend the concept of finding a dominating set for the graph $G(P)$ (as defined in §3) via graph coloring to *labellings* and *multicolorings*.

Let us consider k different colors. A function which labels any vertex of a graph $G(P)$ with a certain set of colored pebbles is called a *k-labelling*. A k-labelling is a *k-multicoloring* if pebbles in adjacent vertices have disjoint color sets. A k-labelling is called *l-uniform* if each vertex gets a pebble set of cardinality l. In this notation a proper graph coloring with k colors is a 1-uniform k-multicoloring.

A k-multicoloring *dominates* the graph $G(P)$ if for each color the set of all vertices labelled with a pebble of this color dominates $G(P)$. Therefore a dominating k-multicoloring of $G(P)$ which uses in total $f(n)$ pebbles implies the existence of an $\lfloor \frac{f(n)}{k} \rfloor$ solution of the prison yard problem for P. Both upper bounds which we derive below are proved using multicoloring arguments.

THEOREM 5.1. $\lfloor \frac{3n}{10} \rfloor + 2$ *guards are sufficient to solve the prison yard problem for rectilinear staircase polygons on n vertices.*

Proof. Let P denote such a polygon, and assume it has north–west orientation (see Figure 6). First we note that the quadrilateralization of P is unique and its weak dual graph is a path $W = q_1, q_2, \ldots, q_{(n-2)/2}$. Each chord of the quadrilateralization connects a convex with a concave vertex and each quadrilateral has a diagonal connecting two convex vertices (called *convex* diagonal).

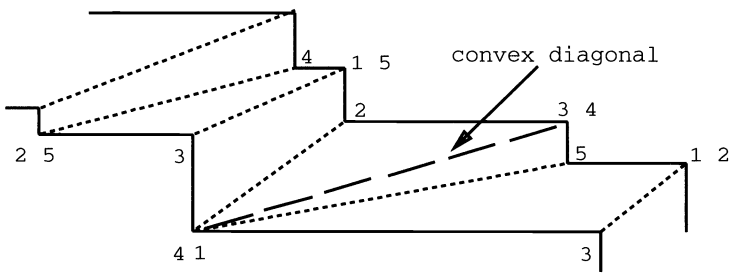


FIG. 6. *5-multicoloring of a staircase polygon.*

Let (d_i), $i = 1, 2, \ldots, n-1$ be the following sequence of polygon edges, diagonals, and chords which we obtain traversing W. We start with the bottom polygon edge d_1 in q_1 followed by the convex diagonal of q_1. d_i is the common edge of $q_{(i-1)/2}$ and

$q_{(i+1)/2}$ for an odd $i \geq 1$, otherwise it is the convex diagonal of $q_{i/2}$. Finally, d_{n-1} is the top edge of P. The d_i's induce a canonical vertex numbering in P. Starting with d_2 each d_i encounters exactly one new vertex, i.e., v_{i+1}. Let Q^i denote the polygon generated by the first i quadrilaterals.

We show that there is a greedy algorithm which, following the path W, constructs a dominating 5-multicoloring of $G(P)$ with the following properties:

- both the north–westernmost vertex v_n and the south–easternmost vertex v_2 are labelled by four pebbles;
- each other convex vertex is colored by two pebbles;
- each concave vertex is colored by one pebble.

While building this multicoloring we maintain the following *invariant*:

- each convex diagonal contains exactly three colors, i.e., there is one common color on both sides of the diagonal.

We start as follows. Color the left vertex v_1 on the bottom edge d_1 by two colors and its right vertex v_2 by one pebble with a third color. Complete the multicoloring of q_1 by repeating one color from v_1 in v_3 together with a pebble having the fourth color. One pebble with the fifth color is put on v_4.

Assume that we have already correctly colored Q^i. The next vertex to be colored is v_{2i+3}. It closes a triangle already labelled by three different colors; hence it gets the remaining two colors. v_{2i+4} is colored by the fifth color not used before in q_{i+1}.

Finally, to get a dominating multicoloring, on v_n we have to place the remaining three colors to dominate both the northern and western extremal edge of P, and similarly we have to place three more pebbles on v_2. Why does this scheme work correctly?

Let $\chi(j)$ denote the set of colors placed on vertex v_j. Assuming q_i is colored correctly we know for its convex diagonal (v_k, v_{2i+1}) that $|\chi(k) \cap \chi(2i+1)| = 1$. Now the algorithm colors v_{2i+3} by two pebbles such that $|\chi(2i+3) \cup \chi(2i+2) \cup \chi(l)| = 5$, where v_l is that vertex of the ith convex diagonal which is in a common exterior triangle with v_{2i+3}. But then for the other vertex v_m of the ith diagonal it follows that $|\chi(2i+3) \cap \chi(m)| = 1$ (so the invariant holds for q_{i+1}) and we can indeed color v_{2i+4} by the fifth color not being an element of the set $\chi(2i+2) \cup \chi(2i+3) \cup \chi(m)$ which has cardinality 4.

In total we use $2\frac{n+4}{2} + \frac{n-4}{2} + 4 = \frac{3n+12}{2}$ pebbles. Consequently, there exists a dominating color class of size $\leq \lfloor \frac{3n}{10} \rfloor + 2$. □

Figure 6 shows part of a 5-multicoloring (with colors 1,...,5) obtained by this algorithm. In a similar way we prove the following statement. The greedy algorithm used will be only slightly more difficult.

THEOREM 5.2. $\lfloor \frac{5n}{16} \rfloor + 2$ *guards are always sufficient to solve the prison yard problem for orthoconvex rectilinear polygons.*

Proof. We give the proof for (horizontal) pyramids only. The result then follows for an arbitrary orthoconvex polygon P by decomposing it into at most two pyramids and one staircase polygon. For the staircase part of P we extend the 5-multicoloring constructed in Theorem 5.1 to an 8-multicoloring by adding an independent dominating 3-multicoloring. Using Theorem 2.1, this 3-multicoloring can be chosen to be 1-uniform for all vertices not on extremal edges.

For a pyramid P we again consider the weak dual path $W = q_1, q_2, \ldots, q_{(n-2)/2}$ of its unique quadrilateralization. We will construct a dominating 8-multicoloring of $G(P)$ with the following properties:

- one of the bottom edge vertices has five pebbles, the other has six pebbles;
- one of the top edge vertices has four pebbles, the other has five pebbles;

- any other convex vertex has three pebbles, each concave one gets two pebbles.

Again, the existence of such a dominating 8-multicoloring can be shown by a greedy algorithm along W. Let d_0 be the bottom edge and d_i for $1 \leq i < \frac{n-2}{2}$ denote the common edge of the quadrilaterals q_i and q_{i+1}. We duplicate both bottom edge vertices and introduce dummy (zero length) horizontal edges on both sides. Now we are in a situation where on both sides of any d_i, $i \geq 0$ there are two horizontal edges. Let $\bar{d}_i$ denote the path consisting of these two horizontal edges with d_i in the middle. $\bar{d}_i$ starts and ends with a convex vertex and has two concave middle vertices. During the algorithm we maintain the following invariant:

- denoting the colors by $1, 2, \ldots, 8$ the color pattern is (modulo a permutation among the eight colors) of the form $123 - 45 - 16 - 247$. In particular, on each $\bar{d}_i$ one color is missing.

First initialize the coloring on $\bar{d}_0$ using this pattern. Having colored the first i quadrilaterals, $\bar{d}_{i+1}$ has two new vertices, one concave and the other convex. The convex one closes an exterior triangle which already has five colors because of the color pattern of $\bar{d}_i$. So it gets the remaining three colors. Now it is not hard to see that q_{i+1} already has six different colors, so we can put the remaining two on the new concave vertex of $\bar{d}_{i+1}$, which then also fulfills the invariant condition.

Finally, after having dominated the two top exterior triangles on both sides, we have to put three more pebbles to the top edge and one more pebble to $\bar{d}_0$. We end up with a dominating 8-multicoloring which gives the bound claimed. □

6. Conclusions, algorithmic aspects, and related problems. In this paper we have examined coloring and multicoloring techniques to solve various art gallery-type problems on rectilinear polygons. In particular, we have proved several new upper and lower bounds for the prison yard problem as summarized in Table 1. Our 3-coloring result in Theorem 2.1 might also be of independent interest. We now want to add a few more remarks.

Coloring versus multicoloring. For the result proved in Theorem 2.1 it is essential that all inner faces of the graph G are 4-cycles. The result is not true if, as in the graph $G(P)$ in §3, there are also triangles. So the result cannot be used to prove, for example, an $\lfloor \frac{n}{3} \rfloor$-bound for the prison yard problem.

On the other hand, it seems curious to use five or eight dominating color classes when constructing *one* dominating vertex set as in the previous section. The point here is that in the constructed multicoloring all convex (concave) vertices not on extremal edges get the same number of pebbles. So it is trivial to count the pebbles because the number of convex (concave) vertices does not depend on the special shape of the polygon. Of course we could try to directly construct, i.e., in a greedy way, a dominating set. However, even in such a regular example as Figure 7 it would be hard to give a good estimate of its size. We think that replacing the multicoloring argument used in Theorem 5.2 by an 8-labelling one can show that the following conjecture is correct.

CONJECTURE 6.1. *There is an absolute constant c such that any rectilinear prison yard can be watched by $\frac{5n}{16} + c$ vertex guards.*

We mention another possible application of the multicoloring technique: namely, there is some evidence that for the rectilinear art gallery problem in the presence of holes, the $\frac{2n}{7}$-lower bound in [6] for the vertex guard number is tight. At first glance the bound seems to be nonintuitive. However, the next conjecture (if true) gives a satisfying answer.

CONJECTURE 6.2. *For each quadrilateralized rectilinear polygon, possibly with*

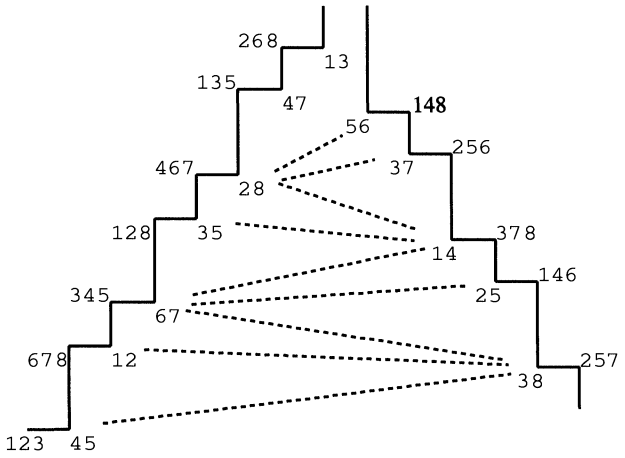


FIG. 7. *8-multicoloring of the lower bound example.*

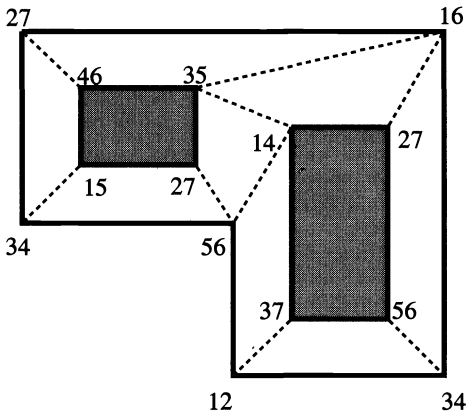


FIG. 8. *Illustration of Conjecture* 6.2.

holes, there is a 2-uniform dominating 7-multicoloring.

We illustrate this conjecture in Figure 8 by showing such a 7-multicoloring of the lower bound example from [6].

Further, it would be interesting to find applications of Theorem 2.1 or of some multicoloring/labelling to nonrectilinear art gallery-type problems; compare [4], [14], and Chapter 5.2 in [12].

Algorithmic aspects. All upper bound results proved in this paper can be converted into efficient algorithms. Since our coloring and multicoloring methods require quadrilateralized polygons, it should be mentioned that based on Chazelle's polygon triangulation algorithm [1] one can quadrilateralize simple rectilinear polygons in linear time. In contrast to this the $O(n \log n)$ bound is optimal for quadrilateralizing rectilinear polygons with holes. It is straightforward that for staircase polygons and

orthoconvex polygons the greedy algorithms from Theorems 5.1 and 5.2 imply a linear time solution of the prison yard problem.

Since all general upper bound results in §4 are based on the 3-coloring result (Theorem 2.1), their algorithmic complexity will be dominated by the $O(n^2)$ algorithm for finding even triangulations. However, we believe that this bound can be improved.

CONJECTURE 6.3. *There is a $O(n^{3/2})$ time algorithm which computes an even triangulation for a given 2-connected, bipartite, and plane graph.*

Acknowledgments. The authors would like to thank Joseph O'Rourke for his many valuable comments which considerably improved the readability of this paper.

REFERENCES

[1] B. CHAZELLE, *Triangulating simple polygons in linear time*, Discrete Comput. Geom., 6 (1991), pp. 485–523.
[2] V. CHVÁTAL, *A combinatorial theorem in plane geometry*, J. Combin. Theory Ser. B, 18 (1975), pp. 39–41.
[3] S. FISK, *A short proof of Chvátal's watchman theorem*, J. Combin. Theory Ser. B, 24 (1978), p. 374.
[4] Z. FÜREDI AND D. KLEITMANN, *The prison yard problem*, Combinatorica, 14 (1994), pp. 287–300.
[5] J. KAHN, M. KLAWE, AND D. KLEITMAN, *Traditional galleries require fewer watchmen*, SIAM J. Alg. Disc. Meth., 4 (1983), pp. 194–206.
[6] F. HOFFMANN, *On the rectilinear art gallery problem*, in Proc. International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science 443, Warwick, England, 1990, pp. 717–728.
[7] F. HOFFMANN AND K. KRIEGEL, *A Graph Coloring Result and its Consequences for Polygon Guarding Problems*, Tech. report TR-B-93-08, Inst. f. Informatik, Freie Universität Berlin, 1993.
[8] L. LOVÁSZ, *Combinatorial Problems and Exercises*, North Holland, Amsterdam, 1979.
[9] A. LUBIW, *Decomposing polygonal regions into convex quadrilaterals*, in Proc. 1st ACM Symp. Comp. Geometry, Baltimore, MD, 1985, pp. 97–106.
[10] R. J. LIPTON, D. J. ROSE, AND R. E. TARJAN, *Generalized nested dissection*, SIAM J. Numer. Anal., 16 (1979), pp. 346–358.
[11] T. NISHIZEKI AND N. CHIBA, *Planar Graphs: Theory and Algorithms*, North Holland, Amsterdam, 1988.
[12] J. O'ROURKE, *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, 1987.
[13] ———, *Computational geometry column* 15, SIGACT News, 23 (1992), pp. 26–28.
[14] T. SHERMER, *Triangulation Graphs that Require Extra Guards*, Computer Graphics Tech. report No. 3D-13, New York Institute of Technology, 1984.
[15] ———, *Recent results in art galleries*, in IEEE Proceedings, 80, 1992, pp. 1384–1399.
[16] E. H. THOMPSON, *An Introduction to the Algebra of Matrices with Some Applications*, Adam Hilger LTD, London, 1969.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 225–232, May 1996

006

CLASSIFYING HYPERPLANES IN HYPERCUBES*

OSWIN AICHHOLZER† AND FRANZ AURENHAMMER‡

Abstract. We consider hyperplanes spanned by vertices of the unit d-cube. We classify these hyperplanes by parallelism to coordinate axes, by symmetry of the d-cube vertices they avoid, as well as by so-called hull-honesty. (Hull-honest hyperplanes are those whose intersection figure with the d-cube coincides with the convex hull of the d-cube vertices they contain; they do not cut d-cube edges properly.) We describe relationships between these classes and give the exact number of hull-honest hyperplanes in general dimensions. An experimental enumeration of all spanned hyperplanes up to dimension eight showed us the intrinsic difficulty of developing a general enumeration scheme. Motivation for considering such hyperplanes stems from coding theory, from linear programming, and from the theory of machine learning.

Key words. d-cube, hyperplane cuts, hyperplane enumeration

AMS subject classifications. 68R05, 52A25

1. Introduction.

Among the simplest high-dimensional geometric objects is the d-dimensional hypercube (d-cube), which is the vector sum of d mutually orthogonal line segments of equal length. The d-cube is a convex polytope that, when having unit edge length, may be expressed as $C^d = [0,1]^d$. Sometimes C^d is called the measure polytope, suggested by its use as the unit of content. Despite its simple definition, C^d has been an object of study from different points of view. The theory of convex polytopes provides classical results concerning sections and projections of hypercubes; see Coxeter [7] and Grünbaum [12]. Purely combinatorial properties of C^d, mainly involving certain subgraphs formed by its edges and vertices (the latter are just the various d-tuples of binary digits), have been investigated extensively in coding theory and in communication theory; see, e.g., [5, 9, 10, 25]. Many easily stated questions concerning the geometry of C^d are still unsettled. A long-standing elementary conjecture on hypercube space fillings (Keller's conjecture) has been recently disproved by Lagarias and Shor [16].

In recent years, increased interest in problems involving the placement of hyperplanes in hypercubes can be observed. Among other areas, motivation stems from coding theory [25], from linear programming [11, 23], and from the theory of machine learning [2, 14, 20, 24]. Again, various elementary questions turned out to be surprisingly difficult. Let us mention just a few of them. What is the minimum number of (nonaxis-parallel) hyperplanes that cover all the 2^d vertices of C^d ? How many edges of C^d may be cut by a single hyperplane? Which hyperplane intersects C^d in a $(d-1)$-polytope of maximum volume? (This was solved in 1986 by Ball [3]; see also [4].) How many and which types of hyperplanes can be spanned by vertices of C^d ?

The present paper is devoted to the last question. Let us denote with $\mathcal{H}(C^d)$ the set of all hyperplanes that can be spanned by (d affinely independent) vertices of C^d. An attempt is made to characterize and classify the hyperplanes in $\mathcal{H}(C^d)$. An

* Received by the editors February 23, 1994; accepted for publication (in revised form) May 25, 1995.

† Institute for Theoretical Computer Science, Graz University of Technology, Klosterwiesgasse 32/2, A-8010 Graz, Austria (oaich@igi.tu-graz.ac.at). The research of this author was supported by the Austrian Ministry of Science and the Jubiläumsfond der Österreichischen Nationalbank.

‡ Institute for Theoretical Computer Science, Graz University of Technology, Klosterwiesgasse 32/2, A-8010 Graz, Austria (auren@igi.tu-graz.ac.at). Part of this work was done while this author was with the Institute for Information Processing and Computer Supported New Media.

obvious criterion is *parallelism*, where we distinguish between k-parallel hyperplanes (being parallel to exactly $k > 0$ coordinate axes) and skew (0-parallel) hyperplanes. A concept that turns out to be important is *symmetry*, which is fulfilled by hyperplanes that halve the set of d-cube vertices they avoid.

We show that k-parallel hyperplanes in $\mathcal{H}(C^d)$ correspond to skew ones in $\mathcal{H}(C^{d-k})$, and that skew and asymmetric hyperplanes in $\mathcal{H}(C^d)$ correspond to skew and symmetric ones in $\mathcal{H}(C^{d+1})$ in a unique manner. This tells us that the skew and symmetric type is the only really interesting one. The enumeration of all hyperplanes of this type, however, is complicated by the following unpleasant phenomenon. In each further dimension new kinds of hyperplanes appear that seem not to be accessible by using the results for lower dimensions. In fact, a theorem by Naumann [19] gives some evidence for the intrinsic complexity of $\mathcal{H}(C^d)$. Every $(d-1)$-polytope is obtainable as the intersection of a $(d-1)$-dimensional hyperplane and a hypercube of sufficiently high dimension. We were able to enumerate experimentally the sets up to $\mathcal{H}(C^8)$ but did not succeed in developing an enumeration scheme for $\mathcal{H}(C^d)$ for general d. We rediscovered an upper bound on the size of the normal vector of hyperplanes in $\mathcal{H}(C^d)$ which we needed to speed up the enumeration. Generally, we count the contributions of this paper among the first steps towards a systematic study of the set $\mathcal{H}(C^d)$.

The situation gets strikingly simpler if we restrict attention to hyperplanes in $\mathcal{H}(C^d)$ that do not (properly) cut d-cube edges. Those might be called *hull-honest* hyperplanes as their intersection figure with C^d coincides with the convex hull of the vertices of C^d they contain. We give an easy-to-use characterization of hull-honest hyperplanes that leads us to their exact total number in general dimensions.

The following notation will be used throughout. A *hyperplane* H in Euclidean d-space $\mathbf{R}^d$ is fixed by a pair (v, b), with $v \in \mathbf{R}^d, b \in \mathbf{R}$, and $H = \{x \in \mathbf{R}^d | v \cdot x = b\}$. For brevity, we shall also write $H = (v, b)$. It follows from linear algebra (Cramer's rule) that there is always an integer solution for v provided $H \in \mathcal{H}(C^d)$. It will be assumed that v is an integer vector and as short as possible; i.e., the greatest common divisor (gcd) of its entries is one. For convenience, we will not distinguish between a vector and its transposed form.

With V^d we denote the vertex set of C^d. Since C^d was defined to be the unit d-cube, $V^d = \{0,1\}^d$, which is the set of all the binary strings of length d. The *Hamming distance* of two vertices $x, y \in V^d$ is the number of coordinates where x differs from y. Note that x and y are the two endpoints of an edge of C^d iff their Hamming distance is one (and occurs at the ith bit iff the edge is parallel to the ith axis of $\mathbf{R}^d$). By symmetry of C^d, for each vertex $x = (x_1, x_2, \ldots, x_d) \in V^d$ there is a vertex $\overline{x} = (1 - x_1, 1 - x_2, \ldots, 1 - x_d) \in V^d$ which we call the *antipode* of x in C^d.

2. Parallel and skew hyperplanes. Our set of interest, $\mathcal{H}(C^d)$, can be partitioned into classes by considering parallelism of its hyperplanes to coordinate axes. Let us call a hyperplane $H \in \mathbf{R}^d$ *k-parallel* $(0 \le k \le d-1)$ if H is parallel to exactly k axes of $\mathbf{R}^d$, that is, to exactly k spanning edges of C^d. 0-parallel hyperplanes are called *skew*. The following is a trivial but useful observation.

Observation 1. $H = (v, b) \in \mathcal{H}(C^d)$ is k-parallel iff exactly k entries of v vanish.

For $d > 0$ and $0 \le k < d$, let $e_k(d)$ denote the number of k-parallel hyperplanes in $\mathcal{H}(C^d)$. It is possible to relate e_k to e_0 (the number of skew hyperplanes) in a simple way.

Observation 2. For $d > 0$ and $0 \le k < d$ we have $e_k(d) = \binom{d}{k} e_0(d-k)$.

Proof. Let $H = (v, b) \in \mathcal{H}(C^d)$ be k-parallel. Then k out of d entries of v are zero and there are $\binom{d}{k}$ possibilities to place them. The remaining entries of v define a

vector of dimension $d-k$ without zero entries which obviously is the normal vector of a skew hyperplane in $\mathcal{H}(C^{d-k})$. The number of such hyperplanes was defined to be $e_0(d-k)$. □

Note that we can thus restrict attention to skew hyperplanes, as each nonskew hyperplane has its skew analogue in some dimension lower.

3. Symmetric hyperplanes. Next we study ways in which a hyperplane $H \in \mathcal{H}(C^d)$ may partition the vertex set V^d. To this end, we call H a (j,p,k)-*plane* if it contains $p \geq d$ vertices and splits the rest into subsets of cardinalities j and k where $j \leq k$.

One might conjecture that a valid triple (j,p,k) uniquely determines a hyperplane in $\mathcal{H}(C^d)$ up to hypercube symmetries. This is false already in dimension 5 where hyperplanes with a fixed triple (j,p,k) may have a different intersection with the d-cube. For instance, the two hyperplanes $H = ((-3,1,1,1,1),0)$ and $H' = ((-1,-3,1,2,2),0)$ in $\mathcal{H}(C^5)$ are both (11, 5, 16)-planes. However, H cuts 10 edges of C^5 in their interior while H' does this 13 times. Five is the lowest dimension where this phenomenon occurs, which turns out to be frequent in higher dimensions. This already reveals part of the difficulty of classifying the hyperplanes in $\mathcal{H}(C^d)$.

A *symmetric* hyperplane in $\mathbf{R}^d$ is one which passes through the center $(\frac{1}{2},\ldots,\frac{1}{2})$ of C^d, that is, a hyperplane $(v,b) \in \mathbf{R}^d$ with $\sum_{i=1}^d v_i = 2b$. For symmetric hyperplanes H, every two antipodal vertices $x, \overline{x} \in V^d$ either both lie in H or they lie on opposite sides of H. Symmetric (j,p,k)-planes thus have $j = k$; that is, the vertices of the d-cube which are avoided by the hyperplane are partitioned into equal-sized subsets.

There is a bijection between asymmetric skew (j,p,k)-planes in $\mathcal{H}(C^d)$ and symmetric skew $(j+k,2p,j+k)$-planes in $\mathcal{H}(C^{d+1})$.

THEOREM 1. *The number of asymmetric skew hyperplanes in $\mathcal{H}(C^d)$ that contain exactly p vertices of C^d is equal to the number of symmetric skew hyperplanes in $\mathcal{H}(C^{d+1})$ with exactly $2p$ vertices of C^{d+1}.*

Proof. There is an obvious bijection between hyperplanes $(v,b) \in \mathbf{R}^d$ and symmetric hyperplanes $(w,b) \in \mathbf{R}^{d+1}$, given by $w_i = v_i$ for $i = 1,\ldots,d$ and $w_{d+1} = 2b - \sum_{i=1}^d v_i$. Note that, under this correspondence, (v,b) is asymmetric and skew if and only if (w,b) is symmetric and skew. Furthermore, it is easy to check that a vertex x of C^d lies in (v,b) if and only if the antipodal vertex pair $(x,0)$ and $(\overline{x},1)$ of C^{d+1} lies in (w,b). Hence (v,b) is an asymmetric and skew hyperplane in $\mathcal{H}(C^d)$ only if (w,b) is a symmetric and skew hyperplane in $\mathcal{H}(C^{d+1})$. The number of vertices of C^{d+1} that lie in (w,b) is exactly twice the number of vertices of C^d that lie in (v,b). □

The merit of Theorem 1 is that attention can be restricted to skew symmetric hyperplanes for further classification. Note also that the number of vertices covered by a symmetric hyperplane $H \in \mathcal{H}(C^d)$ is at least $2(d-1)$.

4. Hull-honest hyperplanes. A completely different way to classify the hyperplanes in $\mathcal{H}(C^d)$ is by considering their surface of intersection with C^d. To this end, $H \in \mathcal{H}(C^d)$ is called a *hull-honest* hyperplane if the convex hull of $H \cap V^d$ coincides with the polytope $H \cap C^d$. Expressed in different terms, H cuts edges of the d-cube only at their vertices but not in their interior. In fact, in dimensions 2 (square) and 3 (cube) each spanned line (plane) is hull honest. But starting with dimension 4, spanned hyperplanes can properly intersect edges of the d-cube, and the convex hull of all the vertices lying on the hyperplane is no longer the surface

of intersection of hyperplane and d-cube. Hull-honest hyperplanes seem to be the only type in $\mathcal{H}(C^d)$ which has been studied previously, motivated by the intersection polytopes they generate [7, 6]. We now give an easy-to-use criterion to characterize hull-honest hyperplanes, and we also give the exact number of such hyperplanes in general dimensions.

THEOREM 2. *Let H be a hyperplane in $\mathcal{H}(C^d)$ and let $v = (v_1, \ldots, v_d)$ be the (shortest integer) normal vector of H. Then H is hull honest iff $v \in \{-1, 0, 1\}^d$.*

Proof. The assertion is trivial for $d = 1$, so let $d \geq 2$ below.

Now assume that H is hull honest. We first prove $v \in \{-1, 0, 1\}^d$ under the assumption that H contains the origin.

We claim that if v_i and v_j are nonzero entries of v of opposite sign then $|v_j| = |v_i|$. Suppose for contradiction that $|v_j| > |v_i|$. Then the adjacent vertices $x = (0, \ldots, 0_i, \ldots, 1_j, \ldots, 0)$ and $y = (0, \ldots, 1_i, \ldots, 1_j, \ldots, 0)$ lie on opposite sides of H and, thus, H intersects the edge between them.

Thus $|v_j| = |v_i|$ whenever v_i and v_j have opposite sign. Since not all nonzero entries of v have the same sign (since H contains the origin but also other vertices), we conclude that all nonzero entries of v must have the same absolute value A. But $v_1, \ldots, v_d$ have gcd 1, so we have $A = 1$ and $v \in \{-1, 0, 1\}^d$.

To cover hull-honest hyperplanes $H = (v, b)$ that avoid the origin, we choose some vertex x of C^d in H and consider $\mathbf{R}^d$ as being spanned by the d-cube edges emanating from x. This transformation affects only the signs of the entries of v, leaving their absolute values unchanged.

For the "if" part, let $H = (v, b)$ have $v \in \{-1, 0, 1\}^d$. Suppose H is not hull honest. Then there is at least one edge of C^d which is intersected by H. The vertices of this edge, x and y, lie on opposite sides of H; hence, $v \cdot x > b$ and $v \cdot y < b$. Since x, y, and v are all integers, we have $v \cdot x \geq b + 1$ and $v \cdot y \leq b - 1$ which implies $v \cdot x \geq v \cdot y + 2$. On the other hand, x and y have Hamming distance one. Say they differ at position i. This gives $|v \cdot x - v \cdot y| = |v_i|$; hence $|v_i| \geq 2$. This contradiction proves that H is hull honest. □

For skew hyperplanes no entry of a normal vector is zero. Skew hull-honest hyperplanes, therefore, always have a normal vector with entries from $\{-1, 1\}$. We will now use this fact to derive their exact number in general dimensions. Intuitively speaking, we look at the 2^{d-1} main diagonals of the d-cube and show that there are precisely $d - 1$ hull-honest hyperplanes perpendicular to a diagonal.

THEOREM 3. *Let v be a vector in $\{-1, 1\}^d$, $d \geq 2$, and let b be an integer. Then the hyperplane (v, b) is spanned by a set of vertices of C^d if and only if b satisfies $n(v) - d < b < n(v)$, where $n(v)$ is the number of 1's in v.*

Before proving the theorem, we use it to count the number of skew hull-honest hyperplanes in $\mathcal{H}(C^d)$.

COROLLARY 1. *The number of skew hull-honest hyperplanes in $\mathcal{H}(C^d)$ is given by*

$$h_0(d) = \begin{cases} 2, & d = 1, \\ (d-1)2^{d-1}, & d \geq 2. \end{cases}$$

Proof. The case $d = 1$ is obvious, so let $d \geq 2$. By Theorems 2 and 3, we only need to count the number of distinct hyperplanes of the form (v, b) with $v \in \{-1, 1\}^d$ and $n(v) - d < b < n(v)$. For each $v \in \{-1, 1\}^d$, there are exactly $d - 1$ integers b within these bounds. Thus there are $2^d(d-1)$ pairs (v, b) satisfying the requirements. Each skew hull-honest hyperplane is counted twice because (v, b) and $(-v, -b)$ give the same hyperplane. □

Proof of Theorem 3. Note first that if b lies outside the given range, then (v, b) contains at most one vertex of C^d, and so the only if part of the theorem follows.

For the if part, define, for $v \in \{-1, 1\}^d$, the vertex $x(v)$ of C^d such that $x_i(v) = \frac{1}{2}(1 + v_i)$. Let $X(v, k)$ denote the set of all vertices of C^d that differ from $x(v)$ in exactly k coordinates, i.e., whose Hamming distance from $x(v)$ is exactly k. Observe that $X(v, k)$ is contained in the hyperplane $(v, n(v) - k)$. Furthermore, it is an easy exercise to show that for k between 1 and $d - 1$, the vertices in $X(v, k)$ span the hyperplane. Thus for $n(v) - d < b < n(v)$ the hyperplane (v, b) is spanned by the vertices in $X(v, n(v) - b)$. □

Note that a skew and hull-honest hyperplane $(v, b) \in \mathcal{H}(C^d)$ thus contains precisely $\binom{d}{n(v)-b}$ vertices of C^d. (In fact, the corresponding intersection figures turn out to be well-known polytopes [7] and to have interesting properties [6, 15].) Moreover, up to hypercube symmetries, there is a unique symmetric, skew, and hull-honest hyperplane in every even dimension and no such one in odd dimensions.

For $d \geq 1$, let $h(d)$ denote the total number of hull-honest hyperplanes in $\mathcal{H}(C^d)$, and let $h_k(d)$ count those that are k-parallel $(0 \leq k \leq d - 1)$.

THEOREM 4.

$$h_k(d) = \begin{cases} \binom{d}{k}(d - k - 1)2^{d-k-1} & \text{for } 0 \leq k \leq d - 2, \\ 2d & \text{for } k = d - 1, \end{cases}$$

$$h(d) = \frac{3^d(2d - 3)}{6} + 2d + \frac{1}{2}.$$

Proof. According to Observation 2 on the relation between skew and k-parallel hyperplanes we have $e_k(d) = \binom{d}{k} e_0(d - k)$. The proof of Observation 2 shows that hull honesty is not affected by the construction of k-parallel hyperplanes as we only add zeros to the normal vector of the original skew hyperplane. Therefore the relation also applies to hull-honest hyperplanes: $h_k(d) = \binom{d}{k} h_0(d - k)$. By using the result of Corollary 1 for $h_0(d)$ we get the formula for $h_k(d)$ as claimed above. Finally, summing overall values of k gives the total number of hull-honest hyperplanes in $\mathcal{H}(C^d)$:

$$h(d) = \sum_{k=0}^{d-2} \left[\binom{d}{k}(d - k - 1)2^{d-k-1} \right] + \underbrace{2d}_{k=d-1}.$$

This simplifies to $h(d) = \frac{3^d(2d-3)}{6} + 2d + \frac{1}{2}$ by routine manipulation using the binomial theorem. □

5. Exhaustive enumeration of hyperplanes. To get an idea of the behavior of the numbers and types of hyperplanes in $\mathcal{H}(C^d)$ for dimensions $d \geq 4$ we did some numerical investigations which are briefly reported below. For a more complete description and discussion see Aichholzer [1].

In order to make the enumeration of $\mathcal{H}(C^d)$ efficient we considered the following question. For a hyperplane $(v, b) \in \mathcal{H}(C^d)$, how large can the magnitude of the entries of v be? (Recall that we assumed v to be an integer and as short as possible.) A similar problem was considered in the threshold circuit literature many years ago, and the upper bound $d^{d/2}$ is well known. Actually, the bound there is slightly weaker, $(d + 1)^{(d+1)/2}$, since it matters from which side the determining points of a threshold hyperplane come from.

TABLE 1
Upper bound of Fact 1 versus the determinant of $(0,1)$*-matrices.*

Dimension d	1	2	3	4	5	6	7	8	9	10
$2 \cdot \left(\frac{d+1}{4}\right)^{(d+1)/2}$	1.00	1.30	2.00	3.49	6.75	14.18	32.00	76.89	195.3	521.6
$\max \lvert\det(\{0,1\}^{d\times d})\rvert$	1	1	2	3	5	9	32	56	144	320
enumerative max. vector entry	1	1	1	2	3	5	9	18	42	≥ 98 ≤ 144

We found the improved bound below which, as we have recently learned, was already given by Muroga, Toda, and Takasu [18] and can be found in more detail in the book of Muroga [17]. Actually, this improvement was rediscovered several times (cf. the survey papers of Orponen [21] and Parberry [22]), most recently by Håstad [14].

FACT 1. *Let* A *be a* $(0,1)$*-matrix of size* $d \times d$. *Then* $|\det(A)| \leq 2\left(\frac{d+1}{4}\right)^{(d+1)/2}$ *which is sharp for* $d = 2^k - 1$, $k \in \mathbf{N}$.

THEOREM 5. *Let* $H = (v, b) \in \mathcal{H}(C^d)$ *and* $v = (v_1, \ldots, v_d)$,

$$|v_i| \leq 2^{-(d-1)} d^{\frac{d}{2}} \quad \text{for } i = 1, \ldots, d,$$

$$|b| \leq 2^{-d}(d+1)^{\frac{d+1}{2}}.$$

For threshold circuits the bound for the weights v_i and the threshold level b is the same, whereas we can bound v_i stronger than b for the reason pointed out above.

The bound in Fact 1 is achieved via a transformation from Hadamard matrices of dimension $d+1$. A *Hadamard matrix* M is an $n \times n$ matrix with entries from $\{-1, 1\}$ which satisfies $MM^T = M^T M = nI_n$. In [13] it is shown that there always exists a Hadamard matrix when $n = 2^k$, and its existence is conjectured for every $n = 4k$. (This conjecture has been verified for $n < 428$; see [8, 13].) Depending on d mod 4, some better bounds on the determinant of $(0,1)$-matrices of order d and therefore for Theorem 5 are known; see [8] and references therein.

The bound on v in Theorem 5 is not tight for dimensions $d = 2^k$, as one might conjecture from Fact 1. We can give a counterexample for dimension 8: The bound on order-7 matrix determinants is 32 and has to be tight by Fact 1. This is also the bound on normal vector entries in dimension 8 that results from Theorem 5. But by exhaustive enumerative computation of $\mathcal{H}(C^8)$ the largest normal vector entry was 18. This is somewhat surprising because testing all those hyperplanes also means testing all linear systems of order 7 with coefficients in $\{0, 1\}$ and with unique solutions. By Cramer's rule each solution coefficient is the ratio of two integers, each having an absolute value ≤ 18 (and not 32, to which the corresponding determinants could increase). Given the importance of linear systems with coefficients in $\{0, 1\}$ in many fields, the gap between maximal solution coefficients and maximal determinant values is worth mentioning. See [8] for some related work in the context of determinant functions.

Table 1 displays the upper bound of Fact 1 versus the determinant of $(0,1)$-matrices and the size of normal vectors, respectively, as far as we computed them.

TABLE 2
Table of the number of different kinds of hyperplanes in the d-cube.

Dimension d	Number of hyperplanes in $\mathcal{H}(C^d)$	Number of skew hyperplanes $e_0(d)$	Number of different types of skew hyperplanes	Number of skew hull-honest hyperplanes $h_0(d)$
1	2	2	1	2
2	6	2	1	2
3	20	8	1	8
4	140	88	3	24
5	3254	2704	8	64
6	252434	234688	35	160
7	71343208	69640192	219	384
8	86246755608	85682904704	1293	896
9	?	?	?	2048
10	?	?	?	4608

Using the above bounds we computed all spanned hyperplanes in dimensions up to 8. Their numbers turned out to increase superexponentially with d (roughly 2^{d^2}), as one would expect from the trivial upper bound $\binom{2^d}{d}$. (Each d-tuple chosen from the 2^d vertices of C^d potentially gives rise to a hyperplane.) An exhaustive enumeration for dimensions higher than 8 currently seems to be out of reach. For $\mathcal{H}(C^8)$, our program, though exploiting theoretical results reported in the present paper and in [1], needed about 12 days on a DEC 5000/240, and for $\mathcal{H}(C^9)$ we estimate a running time of about 35 years.

The experimental results, on the one hand, led us to observe structural properties of $\mathcal{H}(C^d)$ which could be made rigorous but, on the other hand, showed us the intrinsic difficulty of classifying d-cube hyperplanes. In each further dimension we encountered new types of hyperplanes which could not be brought into connection with hyperplanes in lower dimensions. Although we only needed to deal with skew symmetric hyperplanes, this did not really help since the skew type prevails over all other types, and symmetry gains only one dimension. For instance, in $\mathcal{H}(C^8)$ about 99.3 percent of all hyperplanes are skew.

Most hyperplanes are spanned by very few vertices. In $\mathcal{H}(C^8)$ about 30 percent cover only the minimum of 8 vertices. A correlation between the number of vertices covered by a hyperplane, its degree of parallelism, and the size of its normal vector can be observed: The smaller the normal vector and the higher the degree of parallelism, the more vertices may be covered. Saks [24] showed that the maximum number of vertices covered by a skew hyperplane is $\binom{d}{\lfloor \frac{d}{2} \rfloor}$. Section 4 on hull-honest hyperplanes tells us that this bound is attainable. It is not difficult to deduce that a k-parallel hyperplane thus can cover not more than $2^k\binom{d-k}{\lfloor \frac{d-k}{2} \rfloor}$ vertices. Note that $(d-1)$-parallel hyperplanes cover exactly half of the 2^d vertices of C^d, the maximum possible.

Some detailed enumerative results are listed in Table 2.

Acknowledgments. We would like to thank Michael Formann, Philip M. Long, and Lorenz Wernisch for discussions and helpful comments on the presented topic. We also thank Wolfgang Maass and Pekka Orponen for pointing us to literature on related topics in threshold logic. The help of an anonymous referee in simplifying several of our proofs is gratefully acknowledged.

REFERENCES

[1] O. AICHHOLZER, *Hyperebenen in Hyperkuben—eine Klassifizierung und Quantifizierung*, Diplomarbeit, Inst. for Theor. Comput. Sci., Graz Univ. of Technology, Graz, Austria, 1992.

[2] N. ALON AND Z. FÜREDI, *Covering the cube by affine hyperplanes*, European J. Combin., 14 (1993), pp. 79–83.

[3] K. BALL, *Cube slicing in R^n*, Proc. Amer. Math. Soc., 97 (1986), pp. 465–473.

[4] ———, *Volumes of sections of cubes and related problems*, Lecture Notes in Math. 1376, 1989, pp. 251–260.

[5] E. R. BERLEKAMP, *Algebraic Coding Theory*, McGraw-Hill, New York, 1968.

[6] D. CHAKERIAN AND D. LOGOTHETTI, *Cube slices, pictorial triangles, and probability*, Math. Mag., 64 (1991), pp. 219–241.

[7] H. S. M. COXETER, *Regular Polytopes*, Dover Publications, New York, 1963, 1973.

[8] R. CRAIGEN, *The range of the determinant function on the set of $n \times n$ $(0,1)$-matrices*, J. Combin. Math. Combin. Comput., 8 (1990), pp. 161–171.

[9] E. N. GILBERT, *Gray codes and paths on the n-cube*, Bell Systems Tech. J., 37 (1958), pp. 1–12.

[10] R. L. GRAHAM AND H. O. POLLAK, *On the addressing problem for loop switching*, Bell Systems Tech. J., 50 (1971), pp. 2495–2519.

[11] M. GRÖTSCHEL AND M. W. PADBERG, *Polyhedral aspects of the travelling salesman problem*, in The Travelling Salesman Problem, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, eds., John Wiley and Sons, London, New York, 1985.

[12] B. GRÜNBAUM, *Convex Polytopes*, Interscience, New York, 1967.

[13] M. HALL JR., *Combinatorial constructions, studies in combinatorics*, MAA Stud. Math., 17 (1978), pp. 218–253.

[14] J. HÅSTAD, *On the size of weights for threshold gates*, SIAM J. Discrete Math., 7 (1994), pp. 484–492.

[15] L. S. JOEL, D. R. SHIER, AND M. L. STEIN, *Planes, cubes and center-representable polytopes*, Amer. Math. Monthly, 84 (1977), pp. 360–363.

[16] J. C. LAGARIAS AND P. W. SHOR, *Keller's Cube-Tiling Conjecture is False in High Dimensions*, DIMACS Tech. Rep. 92-13, 1992.

[17] S. MUROGA, *Threshold Logic and its Applications*, Wiley-Interscience, New York, 1971.

[18] S. MUROGA, I. TODA, AND S. TAKASU, *Theory of majority decision elements*, J. Franklin Inst., 271 (1961), pp. 376–418.

[19] H. NAUMANN, *Beliebige konvexe Polytope als Schnitte und Projektionen höherdimensionaler Würfel, Simplices und Masspolytope*, Mathematische Zeitschrift, 65 (1956), pp. 91–103.

[20] P. E. O'NEIL, *Hyperplane cuts of an n-cube*, Discrete Math., 1 (1971), pp. 193–195.

[21] P. ORPONEN, *Computational complexity of neural networks: A survey*, Nordic J. Comput., 1 (1994), pp. 94–110.

[22] I. PARBERRY, *Circuit Complexity and Neural Networks*, The MIT Press, Cambridge, MA, 1994.

[23] G. REINELT, *The linear ordering problem: Algorithms and applications*, Research and Exposition in Mathematics 8, Helderman-Verlag, Berlin, 1985.

[24] M. E. SAKS, *Slicing the hypercube*, in Surveys in Combinatorics, K. Walker, ed., London Mathematical Society Lecture Note Series 187, Cambridge University Press, 1993, pp. 211–256.

[25] N. J. A. SLOANE, *A short course on error correcting codes*, in CISM Courses and Lectures 188, Springer, Wien, New York, 1975.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 233–257, May 1996

007

PACKING STEINER TREES: SEPARATION ALGORITHMS *

M. GRÖTSCHEL†, A. MARTIN†, AND R. WEISMANTEL†

Abstract. In this paper, we investigate separation problems for classes of inequalities valid for the polytope associated with the Steiner tree packing problem, a problem that arises, e.g., in very large-scale integration (VLSI) routing. The separation problem for Steiner partition inequalities is $\mathcal{NP}$-hard in general. We show that it can be solved in polynomial time for those instances that come up in switchbox routing. Our algorithm uses dynamic programming techniques. These techniques are also applied to the much more complicated separation problem for alternating cycle inequalities. In this case, we can compute in polynomial time, given some point y, a lower bound for the gap $\alpha - a^T y$ over all alternating cycle inequalities $a^T x \geq \alpha$. This gives rise to a very effective separation heuristic. A by-product of our algorithm is the solution of a combinatorial optimization problem that is interesting in its own right: find a shortest path in a graph where the "length" of a path is its usual length minus the length of its longest edge.

Key words. dual graph, dynamic programming, multicuts, separation, shortest path, Steiner tree

AMS subject classifications. 90C, 90C27

1. Introduction.

To introduce the problem we are considering, let us begin with a few definitions. We are given a graph $G = (V, E)$. If T is a subset of V, then an edge set $S \subseteq E$ is called a *Steiner tree in G for T* if the subgraph induced by S contains a path from s to t for every pair s, t of nodes in T. We will call the elements of T *terminals* and T *terminal set* or *net*. We are further given a list $\mathcal{N} = \{T_1, \ldots, T_N\}$, $N \geq 1$, of nets, i.e., subsets of V, and, moreover, for each edge $e \in E$, a positive *capacity* $c_e \in \mathbb{N}$. A *Steiner tree packing* is an N-tuple $(S_1, \ldots, S_N)$ of edge sets $S_k \subseteq E$ such that each set S_k is a Steiner tree in G for T_k, $k = 1, \ldots, N$, and such that each edge $e \in E$ is contained in at most c_e of these Steiner trees. The *Steiner tree packing problem* is the task to decide whether, for a given graph $G = (V, E)$ with edge capacities $c_e \in \mathbb{N}$ and for a given net list $\mathcal{N}$, a Steiner tree packing exists. The ultimate goal of the investigation is to find a minimum weight Steiner tree packing with respect to some given weight function on the edges.

In [GMW92b] we have shown how the Steiner tree packing problem can be employed to model various versions of the routing problem in VLSI design. We have demonstrated that a cutting plane method based on polyhedral investigations can be successfully utilized for the optimal solution of small real routing problems and that good lower bounds on the optimum solution value can be computed in acceptable running time. The cornerstone of our cutting plane algorithm, introduced in [GMW92a], is an effective implementation of exact and heuristic separation routines for various classes of inequalities that are valid and under mild assumptions facet defining for the associated Steiner tree packing polyhedron. The design and investigation of these separation algorithms are the subject of this paper.

2. The polyhedral approach and some basic results.

In this section we define the Steiner tree packing polyhedron and describe some basic polyhedral results. We start by introducing some graph-theoretic notation.

* Received by the editors October 20, 1993; accepted for publication (in revised form) May 25, 1995. This research was partially supported by Science Program SC1-CT91-620 of the European Community.

† Konrad-Zuse-Zentrum für Informationstechnik Berlin, Heilbronner Straße 10, D-10711 Berlin, Germany.

We denote graphs by $G = (V, E)$, where V is the node set and E the edge set. All graphs we consider are undirected, loopless, and finite. For a given edge set $F \subseteq E$, we denote by $V(F)$ all nodes that are incident to an edge in F. An edge e with endnodes u and v is also denoted by uv. Given two node sets $U, W \subseteq V$, we denote by $[U : W]$ the set of edges in G with one endnode in U and the other in W. For a node set W, we also use $E(W)$ instead of $[W : W]$. A set of node sets $V_1, \ldots, V_p \subset V$, $p \geq 2$, is called a *partition of* V if all sets V_i are nonempty, the node sets are mutually disjoint, and the union of these sets is V. (Note that we use "$\subset$" to denote strict set theoretic containment.) If $V_1, \ldots, V_p$ is a partition of V, then $\delta(V_1, \ldots, V_p)$ denotes the set of edges in G whose endnodes are in different sets. We call $\delta(V_1, \ldots, V_p)$ a *multicut (with p shores) induced by* $V_1, \ldots, V_p$. For $W \subset V$, $W \neq \emptyset$, we write $\delta(W)$ instead of $\delta(W, V \setminus W)$ and call this set the *cut induced by* W. We abbreviate $\delta(\{v\})$ by $\delta(v)$. For an edge set F, we define $d_F(v) := |\delta(v) \cap F|$, which is the *degree* of v in the subgraph (V, F) of G. With a planar graph G we always associate a fixed embedding of G in the plane. The set of edges that are incident to the outer face of a planar graph $G = (V, E)$ will be denoted by $\mathrm{O}_G(E)$. For a subset of edges $S \subseteq E$, we define $\mathrm{O}_G(S) := \mathrm{O}_{(V(S),S)}(S)$; i.e., $\mathrm{O}_G(S)$ denotes the set of outer face edges of the graph induced by S.

Let $K = (v_0, e_1, v_1, e_2, \ldots, v_{l-1}, e_l, v_l)$ be a sequence of nodes and edges, where each edge e_i is incident with the nodes v_{i-1} and v_i for $i = 1, \ldots, l$, and where the edges are pairwise disjoint and the nodes distinct (except possibly v_0 and v_l). K is called a *path* (or a *$[v_0, v_l]$-path*) if $v_0 \neq v_l$ and a *cycle* if $v_0 = v_l$ and $l \geq 2$. The nodes $v_1, \ldots, v_{l-1}$ of a path K are called the *inner nodes* of K. Each edge that connects two nodes of a cycle (path) K and that is not in K is called a *diagonal of* K. We say that two diagonals uv and $u'v'$ *cross with respect to* K if the corresponding nodes appear in the sequence u, u', v, v' or u, v', v, u' by walking along the cycle (path). Similarly, we call two sets of diagonals F_1 and F_2 *cross-free* if, for all $e_1 \in F_1$ and $e_2 \in F_2$, e_1 and e_2 do not cross. Otherwise, F_1 and F_2 are *crossing*. For our purposes, it is convenient to consider a path P or a cycle C, respectively, as a subset of the edge set. We call an edge set B a *tree* if $(V(B), B)$ is connected and contains no cycle. The *leaves* of a tree B are the nodes that are incident to exactly one edge of B.

Note that a Steiner tree is not a tree, in general. (Our Steiner trees are supersets of "ordinary" Steiner trees. We employ this slight change of the more standard definition, since it simplifies a number of technicalities of our polyhedral investigations.) A Steiner tree that is a tree and whose leaves are terminals is called an *edge-minimal Steiner tree*. We call an edge e in a graph G, given some net T, a *Steiner bridge* if every Steiner tree for T in G contains e.

We now introduce a polytope associated with the Steiner tree packing problem. We are given a graph $G = (V, E)$ with capacities $c_e \in \mathbb{N}$ for all $e \in E$ and a net list $\mathcal{N} = \{T_1, \ldots, T_N\}$, $N \geq 1$. We will denote an *instance of the Steiner tree packing problem* by the triple $(G, \mathcal{N}, c)$. Let $\mathbb{R}^{\mathcal{N} \times E}$ denote the $N \cdot |E|$-dimensional vector space $\mathbb{R}^E \times \cdots \times \mathbb{R}^E$, where the components of each vector $x \in \mathbb{R}^{\mathcal{N} \times E}$ are indexed by x_e^k for $k \in \{1, \ldots, N\}$, $e \in E$. Moreover, for a vector $x \in \mathbb{R}^{\mathcal{N} \times E}$ and $k \in \{1, \ldots, N\}$, we denote by $x^k \in \mathbb{R}^E$ the vector $(x_e^k)_{e \in E}$, and we simply write $x = (x^1, \ldots, x^N)$ instead of $x = ((x^1)^T, \ldots, (x^N)^T)^T$. For a subset $E' \subseteq E$ and a vector $a \in \mathbb{R}^{\mathcal{N} \times E}$, we define a vector $a|_{E'} \in \mathbb{R}^{\mathcal{N} \times E'}$ by $(a|_{E'})_e^k := a_e^k$ for all $k = 1, \ldots, N$ and $e \in E'$. For an edge set $F \subseteq E$, $\chi^F \in \mathbb{R}^E$ denotes the *incidence vector of* F, i.e., $\chi_e^F := 1$ if $e \in F$ and $\chi_e^F := 0$ otherwise. Conversely, for each 0/1-vector $x \in \mathbb{R}^E$, the set $I_x := \{e \in E \mid x_e = 1\}$ is called the *incidence set of* x. The *incidence vector of a*

Steiner tree packing $(S_1, \ldots, S_N)$ is denoted by $(\chi^{S_1}, \ldots, \chi^{S_N})$.

The *Steiner tree packing polyhedron* $\mathrm{STP}(G, \mathcal{N}, c)$ is the convex hull of all incidence vectors of Steiner tree packings. It is easy to see that the following holds:

$$
\begin{aligned}
\mathrm{STP}(G,\mathcal{N},c) = \mathrm{conv}\Big\{ x \in \mathbb{R}^{\mathcal{N}\times E} \mid \\
\text{(i)} \quad & \sum_{e\in\delta(W)} x_e^k \geq 1 \quad \text{for all } W \subset V,\ W \cap T_k \neq \emptyset, \\
& \qquad\qquad (V \setminus W) \cap T_k \neq \emptyset,\ k = 1, \ldots, N, \\
\text{(ii)} \quad & \sum_{k=1}^{N} x_e^k \leq c_e \quad \text{for all } e \in E, \\
\text{(iii)} \quad & 0 \leq x_e^k \leq 1 \quad \text{for all } e \in E,\ k = 1, \ldots, N, \\
\text{(iv)} \quad & x_e^k \in \{0,1\} \quad \text{for all } e \in E,\ k = 1, \ldots, N \Big\}.
\end{aligned}
\tag{2.1}
$$

The inequalities (2.1) (i) are called *Steiner cut inequalities*, inequalities (2.1) (ii) are called *capacity inequalities*, and the ones in (2.1) (iii) are called *trivial inequalities*. In case $N = 1$, the Steiner tree packing polyhedron is also called the *Steiner tree polyhedron*. Note that (2.1) (i)–(iv) yields an integer programming formulation of the weighted Steiner tree packing problem.

We close this section by listing some polyhedral results that are of importance for the remainder of the paper. The reader interested in the proofs of these results is referred to [GMW92a].

First, the problem of deciding whether, for some given $l \in \mathbb{N}$, the dimension of the Steiner tree packing polyhedron is at least l is $\mathcal{NP}$-complete. This follows from the fact that the Steiner tree packing problem itself is $\mathcal{NP}$-complete. (See, for instance, [KL84], [S87].) Due to this fact, we have decided to study the Steiner tree packing polyhedron for problem instances for which the dimension can easily be determined and to look for facet-defining inequalities for these special instances. The justification of the choice to be described below can be found in [GMW92a].

We restrict ourselves to considering instances $(G, \mathcal{N}, c)$, where the graph G is complete, the net list $\mathcal{N} = \{T_1, \ldots, T_N\}$ is *disjoint* (i.e., $T_i \cap T_j = \emptyset$ for all $i, j \in \{1, \ldots, N\}$, $i \neq j$), and the capacities are equal to one ($c = \mathbf{1}$). It can easily be verified that the corresponding Steiner tree packing polyhedron $\mathrm{STP}(G, \mathcal{N}, \mathbf{1})$ is full-dimensional in this case. Lemma 2.1 shows how validity results for the Steiner tree packing polyhedron for some graph can be transformed to validity results for the Steiner tree packing polyhedron for the graph obtained by deleting some edge or splitting some node and, thus, by repeated application, how validity results for the complete graph can be transformed to the general case.

LEMMA 2.1. *Let $(G, \mathcal{N}, c)$ be an instance of the Steiner tree packing problem.*

(a) *(Deleting an edge.) Let $a^T x \geq \alpha$ be a valid inequality of $STP(G, \mathcal{N}, c)$ and suppose $f \in E$ is deleted from G. Then $\hat{a}^T x \geq \alpha$ is a valid inequality of $STP(G \setminus f, \mathcal{N}, c|_{E\setminus\{f\}})$ where $\hat{a}_e^k = a_e^k$ for all $e \in E \setminus \{f\}$, $k \in \{1, \ldots, N\}$ (where $G \setminus f$ denotes the graph that is obtained by deleting edge f).*

(b) *(Splitting a node.) Let $f \in E$ and let $\hat{a}^T x \geq \alpha$ be a valid inequality of $STP(G \,/\, f, \hat{\mathcal{N}}, \hat{c})$. ($G \,/\, f$ denotes the graph that is obtained by shrinking edge f; $\hat{\mathcal{N}}$ and $\hat{c}$ denote the corresponding net list and capacity vector defined*

on G / f.) Then $a^T x \geq \alpha$ defines a valid inequality for $STP(G, \mathcal{N}, c)$ with $a_e^k = \hat{a}_e^k$ for all $e \in E \setminus \{f\}$, $k \in \{1, \ldots, N\}$ and $a_f^k = 0$ for all $k = 1, \ldots, N$.

The next theorem shows that each nontrivial facet-defining inequality of the Steiner tree polyhedron can be lifted to yield a facet-defining inequality of the Steiner tree packing polyhedron.

THEOREM 2.2. *Let $G = (V, E)$ be the complete graph with node set V and let $\mathcal{N} = \{T_1, \ldots, T_N\}$, $N \geq 2$, be a disjoint net list. Let $\bar{a}^T x \geq \alpha$, $\bar{a} \in \mathbb{R}^E$, be a nontrivial facet-defining inequality of $STP(G, \{T_l\}, \mathbf{1})$ for some $l \in \{1, \ldots, N\}$. Then $a^T x \geq \alpha$ defines a facet of $STP(G, \mathcal{N}, \mathbf{1})$, where $a \in \mathbb{R}^{\mathcal{N} \times E}$ denotes the vector with $a_e^l = \bar{a}_e$, $a_e^k = 0$ for all $k = 1, \ldots, N, k \neq l$, $e \in E$.*

Theorem 2.2 implies that in order to obtain a complete description of some Steiner tree packing polyhedron STP $(G, \mathcal{N}, c)$, at least all "individual" Steiner tree polyhedra STP $(G, \{T\}, c), T \in \mathcal{N}$, must be known completely. Of course, this knowledge will hardly do. There are many classes of inequalities that combine at least two nets. We call such inequalities *joint*. In [GMW92a] and [GMW95], several classes of joint inequalities are described.

Polyhedral results such as the ones mentioned above are utilized algorithmically by means of separation algorithms in the framework of a cutting plane method. We will discuss separation problems for classes of inequalities valid for the Steiner tree packing polyhedron and separation algorithms for these classes in the subsequent sections.

3. Separation of the Steiner partition inequalities. Let a graph $G = (V, E)$ and a set of terminals $T \subseteq V$, $|T| \geq 2$ be given. A partition $V_1, \ldots, V_p$, $p \geq 2$, of V is called a *Steiner partition (with respect to T)* if $V_i \cap T \neq \emptyset$ for $i = 1, \ldots, p$. The inequality

$$x(\delta(V_1, \ldots, V_p)) \geq p - 1$$

induced by a Steiner partition $V_1, \ldots, V_p$ is called a *Steiner partition inequality*. (Note that a Steiner cut inequality is the special case where $p = 2$.) A Steiner partition inequality is valid for STP $(G, \{T\}, \mathbf{1})$, because each element of the partition contains terminals and a Steiner tree must span all these terminals, thus "crossing" the multicut at least $p - 1$ times. The separation problem for this class of inequalities can be formulated as follows.

PROBLEM 3.1 (Separation problem for the Steiner partition inequalities). *Let a graph $G = (V, E)$, a terminal set $T \subseteq V$, and a vector $y \in \mathbb{R}^E$ with $0 \leq y_e \leq 1$ for $e \in E$, be given. Decide whether y satisfies all Steiner partition inequalities. If not, find a Steiner partition inequality that y violates.*

Problem 3.1 is $\mathcal{NP}$-hard in general (cf. [GMS92]). Restricting Problem 3.1 to Steiner cut inequalities, i.e., the case $p = 2$, the separation problem can be solved in polynomial time by min-cut computations using any of the many polynomial time max-flow algorithms; see [AMO93]. We show now that if we restrict the graph G to be planar and the set of terminals T to lie on the outer face of G, Problem 3.1 can be solved in time polynomial in the size of G and the encoding length of y. In the following we describe this algorithm.

It is shown in [GM90] that the following conditions are necessary and sufficient for a Steiner partition inequality induced by $V_1, \ldots, V_p$ to be facet-defining, provided

that the graph G is connected and contains no Steiner bridge:

$$\text{(3.1)}\qquad \begin{array}{ll} \text{(i)} & (V_i, E(V_i)) \text{ is connected for } i = 1, \ldots, p, \\ \text{(ii)} & (V_i, E(V_i)) \text{ contains no Steiner bridge with respect} \\ & \text{to the terminal set } V_i \cap T \ (i = 1, \ldots, p), \text{ and} \\ \text{(iii)} & G(V_1, \ldots, V_p) \text{ is 2-node connected,} \end{array}$$

where $G(V_1, \ldots, V_p)$ is the graph obtained from G by contracting each node set of the partition to a single node. Moreover, the proof shows that each Steiner partition inequality that does not define a facet of STP $(G, \mathcal{N}, \mathbf{1})$ is the nonnegative linear combination of facet-defining Steiner partition inequalities and trivial inequalities. Thus, we can restrict ourselves to solving Problem 3.1 for facet-defining Steiner partition inequalities.

Given a Steiner partition $V_1, \ldots, V_p$ satisfying (3.1) (i) and (iii), we now describe how the edge set $\delta(V_1, \ldots, V_p)$ can be viewed as a Steiner tree in a certain "dual" graph.

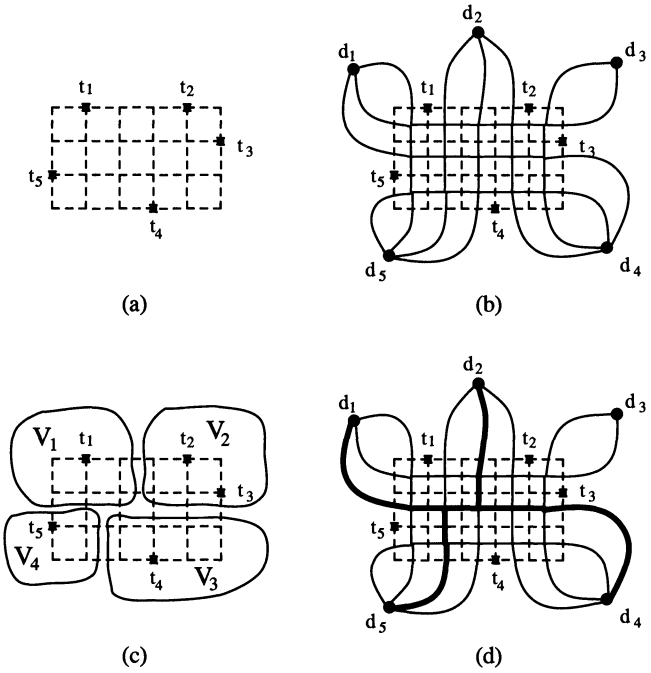


FIG. 1.

For the remainder of this section we assume that the graph G is 2-node connected. We can do this without loss of generality because otherwise the overall problem can be decomposed in an obvious way into subproblems where the corresponding graphs are 2-node connected. Thus, the edge set $O_G(E)$ that encloses the outer face of G is a cycle. We may assume that the terminal set $T = \{t_1, \ldots, t_z\}$ is numbered in a clockwise fashion along this cycle. Let us consider the dual planar graph $G^* = (V^*, E)$ of G. We subdivide the node representing the outer face into z nodes $d_1, \ldots, d_z$ such that every edge of $O_G(E)$ that is passed by walking from t_i to t_{i+1} on $O_G(E)$ in

clockwise order is now incident to d_{i+1} for $i = 1, \ldots, z$. (We identify an index $i > z$ with $((i-1) \text{ modulo } z) + 1$.) Let $G_D = (V_D, E)$ denote the resulting graph and set $T_D := \{d_1, \ldots, d_z\}$. We call T_D the *set of dual terminals.* Observe that instead of working with a bijective mapping, we denote both the edge set of the original graph G and its "dual" G_D by the same symbol E. We make sure that this notational simplification will not lead to confusion. Figure 1(a), showing a 4×6 grid with five terminals, and Figure 1(b), where the edges of G_D are displayed by solid lines, illustrate this construction.

Let us now define the following set of Steiner trees in G_D:

$$\begin{aligned} \mathcal{D} := \{S \subseteq E \mid\ & S \text{ is an edge-minimal Steiner tree in } G_D \\ & \text{for some } J \subseteq T_D,\ |J| \geq 2, \text{ such that} \\ & d_S(j) = 1 \text{ for all } j \in J, \\ & d_S(t) = 0 \text{ for all } t \in T_D \setminus J\}. \end{aligned}$$

Clearly, every Steiner tree S in $\mathcal{D}$ determines the set $J \subseteq T_D$ of its terminals uniquely. For notational ease we will thus often write S_J to denote a Steiner tree S in $\mathcal{D}$ and its associated set J of dual terminals.

LEMMA 3.2. *Let $G = (V, E)$ be a planar graph and T a set of terminals located on the outer face of G. The following statements are then true:*

1. *If $V_1, \ldots, V_p$ is a Steiner partition of V with respect to T satisfying* (3.1) (i) *and* (iii), *then the multicut $\delta(V_1, \ldots, V_p)$ viewed as an edge set of the dual graph G_D is a Steiner tree S_J contained in $\mathcal{D}$ with $|J| = p$.*
2. *If S_J is a Steiner tree in G_D contained in $\mathcal{D}$, then there exists a unique Steiner partition $V_1, \ldots, V_{|J|}$ of V with respect to T satisfying* (3.1) (i) *and* (iii) *such that $S_J = \delta(V_1, \ldots, V_{|J|})$.*

Proof. Let $V_1, \ldots, V_p$ be a Steiner partition satisfying (3.1) (i) and (iii). Since G is planar and all terminals are located on the outer face of G, $G(V_1, \ldots, V_p)$ is outerplanar. This together with property (3.1) (iii) implies that we can assume that the numbering of the partition is clockwise or anticlockwise (without loss of generality clockwise) around the outer face. In addition, we have that

$$[V_i : V_{i+1}] \neq \emptyset \text{ for } i = 1, \ldots, p. \tag{A}$$

For every $i \in \{1, \ldots, p\}$, the graph $(V_i, E(V_i))$ is connected and $V_i \cap T \neq \emptyset$. Hence, there exist $d_{i_1}, d_{i_2} \in T_D$ such that $\delta(V_i)$ defines a path from d_{i_1} to d_{i_2}. Without loss of generality, d_{i_1}, d_{i_2} are chosen such that terminal $t_{i_1} \in V_i$. From (A) and the fact that $V_1, \ldots, V_p$ is a partition it follows that $d_{i_2} = d_{(i+1)_1}$ for $i = 1, \ldots, p$. Thus, $S := \cup_{i=1}^{p} \delta(V_i) = \delta(V_1, \ldots, V_p)$ is a Steiner tree for $J := \cup_{i=1}^{p} \{d_{i_1}\}$. Due to (A) and since $V_1, \ldots, V_p$ is a partition with $V_i \cap T \neq \emptyset$, S is edge-minimal and $d_S(j) = 1$ for all $j \in J$. Property (3.1) (i) and (A) imply that $d_s(t) = 0$ for all $t \in T_D \setminus J$. By construction, $|J| = p$, and thus $|J| \geq 2$. Hence, $S \in \mathcal{D}$.

Conversely, let S_J be a Steiner tree in G_D contained in $\mathcal{D}$; i.e., S_J is an edge-minimal Steiner tree for some $J \subseteq T_D$, $|J| \geq 2$, satisfying $d_S(j) = 1$ for all $j \in J$ and $d_S(t) = 0$ for all $t \in T_D \setminus J$. We number the elements in $J = \{d_{s_1}, \ldots, d_{s_{|J|}}\}$ in clockwise order around the outer face. Every unique path P_i in S from d_{s_i} to $d_{s_{i+1}}$ is a cut in G; i.e., there exists a node set V_i such that $\delta(V_i) = P_i$. Moreover, we can assume that $t_{s_i - 1} \in V_i$, for $i = 1, \ldots, |J|$. Since S is edge-minimal, $V_1, \ldots, V_{|J|}$ is a partition of V. Moreover, $V_1, \ldots, V_{|J|}$ is also a Steiner partition, because $t_{s_i - 1} \in V_i$ for $i = 1, \ldots, |J|$. Since $d_S(d) = 0$ for all $d \in T_D \setminus M$, $(V_i, E(V_i))$ is connected for all $i = 1, \ldots, |J|$, showing (3.1) (i). Furthermore, from $d_S(j) = 1$ for all $j \in J$ it follows

that $[V_i : V_{i+1}] \neq \emptyset$ for $i = 1, \ldots, |J|$. Thus, $G(V_1, \ldots, V_{|J|})$ contains a hamiltonian cycle implying (3.1) (iii). □

Lemma 3.2 shows that the Steiner partitions of V satisfying (3.1) (i) and (iii) are in one-to-one correspondence to the edge-minimal Steiner trees in G_D that are in $\mathcal{D}$. To illustrate this on an example, consider Figures 1(c) and (d): the multicut $\delta(V_1, V_2, V_3, V_4)$ induced by the Steiner partition V_1, V_2, V_3, V_4 of V depicted in Figure 1(c) is a Steiner tree in G_D for the subset $\{d_1, d_2, d_4, d_5\}$ of the dual terminals; see the thick solid lines in Figure 1(d).

To check whether a given vector $y \in \mathbb{R}^E$, $y \geq 0$, satisfies all Steiner partition inequalities $x(\delta(V_1, \ldots, V_p)) \geq p - 1$, we determine the value

$$\alpha := \min_{S_J \in \mathcal{D}} \Big(y(S_J) - |J| \Big). \tag{3.2}$$

If $\alpha \geq -1$, Lemma 3.2 implies that there exists no violated Steiner partition inequality. Otherwise, the corresponding Steiner tree S_J yields the violated Steiner partition inequality $y(S_J) < |J| - 1$.

Observe that the objective function of the minimization problem in (3.2) is not linear. One way to linearize it is to consider the following 2-stage process. First, for every $J \subseteq T_D$ with $|J| \geq 2$, we determine a Steiner tree S_J^* for J in G_D such that the weight $y(S_J^*)$ is minimum, where only those Steiner trees S_J that satisfy $d_{S_J}(j) = 1$ for all $j \in J$ and $d_{S_J}(j) = 0$ for all $j \in T_D \setminus J$ are considered. Then we determine, among all these Steiner trees S_J^*, $J \subseteq T_D$ with $|J| \geq 2$, a Steiner tree $S_{J^*}^*$ such that the value $y(S_{J^*}^*) - |J^*|$ is as small as possible. In other words, (3.2) can be written in the following way:

$$\alpha = \min_{\substack{J \subseteq T_D \\ |J| \geq 2}} \Big(\min_{\substack{S \text{ Steiner tree for } J \\ S \in \mathcal{D}}} y(S) \Big) - |J|. \tag{3.3}$$

However, this does not lead to a polynomial time algorithm. Our approach for the computation of α is based on ideas of [DW71] and [EMV87] who have presented a dynamic programming algorithm for the solution of the following problem.

Suppose we are given a graph $G = (V, E)$ and a set of terminals Z, and we want to compute a minimal (with respect to some weighting $w : E \to \mathbb{R}_+$) Steiner tree for Z. The idea of the algorithm is based on the observation that for every minimal Steiner tree S and every node $v \in V(S)$ that is not a leaf of S, there exists a subset $J \subseteq Z$ such that S can be split into two subtrees, S_1 and S_2, where S_1 is an optimal Steiner tree with respect to $J \cup \{v\}$ and S_2 is an optimal Steiner tree with respect to $(Z \setminus J) \cup \{v\}$. This observation leads to the following recursion formula.

For $J \subseteq Z$ and $v \in V$, let $\gamma(J \cup \{v\})$ denote the value of a minimal Steiner tree in G for $J \cup \{v\}$. Moreover, let $\psi_v(J \cup \{v\})$ be the minimum over all sums of two minimal Steiner trees, each spanning v and a nonempty subset of J such that the two subsets form a partition of J. Then we obtain (see [DW71])

$$\begin{aligned} &\text{(i)} \quad \psi_v(J \cup \{v\}) = \min_{\emptyset \subset I \subset J} \gamma(I \cup \{v\}) + \gamma((J \setminus I) \cup \{v\}), \\ &\text{(ii)} \quad \gamma(J \cup \{v\}) \;\; = \min_{u \in V} w(W(v,u)) + \psi_u(J \cup \{u\}), \end{aligned} \tag{3.4}$$

where $W(u, v)$, $u, v \in V$ denotes a shortest path from u to v in G. Of course, for arbitrary graphs G and terminal sets Z, the running time of the dynamic program based on this recursion is exponential in the number of terminals. However, in the

particular case where G is planar and all terminals lie on the outer face of G, Erickson, Monma, and Veinott (cf. [EMV87]) showed that it suffices to consider only subsets of Z whose elements are located consecutively on the outer face. Since the number of these subsets is quadratic in the number of terminals, a minimal Steiner tree can be computed in polynomial time using this recursion.

Let us return to our problem of determining α. We can clearly use the polynomial time algorithm described above to compute a minimal Steiner tree for every $J \subseteq T_D$ in G_D, because G_D is planar and the dual terminal set T_D (and thus J) lies on the outer face of G_D. We can also consider the additional condition that every Steiner tree S for J has to take $S \in \mathcal{D}$ into account by some slight modifications of the recursion formula. Moreover, by running the recursion appropriately we can simultaneously determine the optimal subset J^* of T_D (and thus solve (3.3)) as follows.

First, from the minimum weight of a Steiner tree for J we subtract the number of its terminals. This can easily be taken into account in the recursion formula, since each terminal is a leaf of the Steiner tree. (See properties of $\mathcal{D}$.) Second, the minimum in (3.3) is taken over all subsets of T_D with at least two elements. The number of these subsets is exponential in the size of the terminals. However, it is possible to decide locally which dual terminal belongs to the optimal solution. Namely, a shortest path $P(v,d), v \in V_D \backslash T_D, d \in T_D$, is a branch of a minimal Steiner tree only if $y(P(v,d)) \leq 1$ holds. This is due to the fact that if such a branch is added to a minimal Steiner tree, the left-hand side of the corresponding Steiner partition inequality increases by the weight of the path, whereas the right-hand side is incremented by one. To sum up, we obtain the recursion

$$
\begin{aligned}
&\text{(i)} \quad y^v_{i,0} := \min\{y(W(v,d_i)) - 1,\ 0\} && \text{for all } v \in V_D \setminus T_D, i = 1,\dots,z,\\
&\text{(ii)} \quad \psi^v_{i,j} := \min_{1 \leq l \leq j} \left(y^v_{i,l-1} + y^v_{i+l,j-l}\right) && \text{for all } v \in V_D \setminus T_D,\\
&\qquad\qquad && i = 1,\dots,z,\ j = 1,\dots,z-1,\\
&\text{(iii)} \quad y^v_{i,j} := \min_{u \in V_D \setminus T_D} \left(y(W(v,u)) + \psi^u_{i,j}\right) && \text{for all } v \in V_D \setminus T_D,\\
&\qquad\qquad && i = 1,\dots,z,\ j = 1,\dots,z-1,
\end{aligned}
\tag{3.5}
$$

where $W(u,v)$, $u,v \in V_D$ denotes a shortest path in G_D from u to v such that $\left(T_D \cap V_D(W(u,v))\right) \setminus \{u,v\} = \emptyset$. This additional restriction is necessary to guarantee that the solution belongs to $\mathcal{D}$. (3.5) (ii) corresponds precisely to the formula in (3.4) (i). (3.5) (iii) is the counterpart of (3.4) (ii), except that one-element terminal sets are treated separately in (3.5) (i); see also the explanations above.

In the following we show that recursion (3.5) works correctly.

For $i = 1,\dots,z$, $j = 0,\dots,z-1$, let $P_{i,j}$ denote the unique path from d_i to d_{i+j} by walking along the outer face of G_D in clockwise order. We define the interval $[d_i, d_{i+j}] := T_D \cap V_D(P_{i,j})$. Consider a Steiner tree S in G_D for some subset $J \subseteq [d_i, d_{i+j}]$, $J \neq \emptyset$. We denote by l_S the index of the "left most" dual terminal and by $l_S + r_S$ the index of the "right most" dual terminal of S, i.e., an element of J; in formulas

$$l_S := i + h^*, \text{ with } h^* := \min\{h \mid h \geq 0,\ d_{i+h} \in V_D(S)\},$$

and

$$r_S := \max\{h \mid h \leq j,\ d_{l_S+h} \in V_D(S)\}.$$

Moreover, for $i = 1,\dots,z$, $j = 1,\dots,z-1$, we introduce the symbol $e_{i,j}$ to denote the edge that is incident to d_i and d_{i+j}. Set $G_{i,j} := (V_D, E \cup \{e_{i,j}\})$. In the planar

representation of $G_{i,j}$ we embed the edge $e_{i,j}$ in the outer face of G_D such that it is homotopic to the path $P_{i,j}$. Figure 2 illustrates this construction. It will turn out to be useful to employ the symbol $e_{i,0}$ in some recursion formula in order to avoid the treatment of additional special cases. We will interpret $e_{i,0}$ as a nonexisting edge and, accordingly, $G_{i,0}$ as the graph G_D.

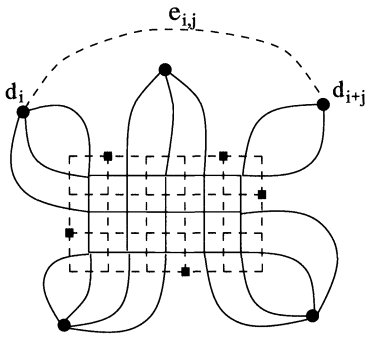


FIG. 2.

LEMMA 3.3. *For $i = 1, \dots, z$, $j = 0, \dots, z-1$, and $v \in V_D \setminus T_D$, define $\mathcal{D}^v_{i,j} := \{S \subseteq E \mid S$ is an edge-minimal Steiner tree for $J \cup \{v\}$ with $J \subseteq [d_i, d_{i+j}]$ such that $d_S(j) = 1$ for $j \in J$ and $d_S(j) = 0$ for $j \in T_D \setminus J$ and such that, if $J \neq \emptyset$, in addition $v \in V_D(O_{G_{l_S,r_S}}(S \cup \{e_{l_S,r_S}\}))\}$. Then, for the values that are computed using recursion* (3.5), *the following property holds:*

$$y^v_{i,j} \leq \min_{\emptyset \subseteq J \subseteq [d_i, d_{i+j}]} \Big(\min_{\substack{S \text{ Steiner tree for } J \cup \{v\} \\ S \in \mathcal{D}^v_{i,j}}} y(S) \Big) - |J|$$

for all $v \in V_D \setminus T_D$, $i = 1, \dots, z, j = 0, \dots, z-1$.

Proof. For $r = 1, \dots, z$, $s = 0, \dots, z-1$ and $v \in V_D \setminus T_D$, let $D^v_{r,s} \subseteq [d_r, d_{r+s}]$ and $S^v_{r,s} \in \mathcal{D}^v_{r,s}$ be a Steiner tree for $D^v_{r,s} \cup \{v\}$ such that

$$y(S^v_{r,s}) - |D^v_{r,s}| = \min_{\emptyset \subseteq J \subseteq [d_r, d_{r+s}]} \Big(\min_{\substack{S \text{ Steiner tree for } J \cup \{v\} \\ S \in \mathcal{D}^v_{r,s}}} y(S) \Big) - |J|.$$

We must show that $y^v_{i,j} \leq y(S^v_{i,j}) - |D^v_{i,j}|$ for all $v \in V_D \setminus T_D$, $i = 1, \dots, z, j = 0, \dots, z-1$. We prove the statement by induction over j. For $j = 0$, Lemma 3.3 is obviously true. Suppose the statement also holds for all $k = 0, \dots, j-1$. Let $v \in V_D \setminus T_D$ be any arbitrary node and $i \in \{1, \dots, z\}$. For ease of notation let $l := l_{S^v_{i,j}}$, $r := r_{S^v_{i,j}}$, and $F := O_{G_{l,r}}(S^v_{i,j} \cup \{e_{l,r}\})$. We distinguish two cases.

(1) $d_{S^v_{i,j}}(v) \geq 2$. Since G_D is planar, since all dual terminals (and thus $D^v_{i,j}$) lie on the outer face of G_D, and since $v \in V_D(F)$, there exists an index $q \in \{1, \dots, j\}$ and two nonempty disjoint subtrees $S_1, S_2 \subseteq S^v_{i,j}$ with $S_1 \cup S_2 = S^v_{i,j}$ such that S_1 is an edge-minimal Steiner tree for $(D^v_{i,j} \cap [d_i, d_{i+q-1}]) \cup \{v\}$ and S_2 is an edge-minimal Steiner tree for $(D^v_{i,j} \cap [d_{i+q}, d_{i+j}]) \cup \{v\}$. (See Figure 3.)

Moreover, $S^v_{i,j} \in \mathcal{D}^v_{i,j}$ implies that $d_{S_1}(d) = 1$ for all $d \in D^v_{i,j} \cap [d_i, d_{i+q-1}]$ and $d_{S_1}(d) = 0$ for all $d \in T_D \setminus (D^v_{i,j} \cap [d_i, d_{i+q-1}])$. The same holds for S_2; i.e., $d_{S_2}(d) = 1$ for all $d \in D^v_{i,j} \cap [d_{i+q}, d_{i+j}]$ and $d_{S_2}(d) = 0$ for all $d \in T_D \setminus (D^v_{i,j} \cap [d_{i+q}, d_{i+j}])$. Let

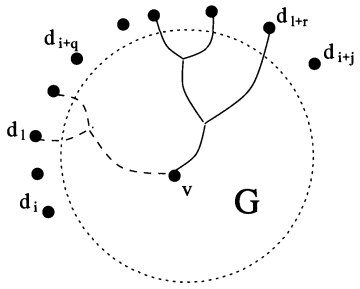


FIG. 3.

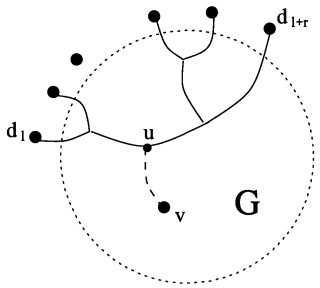


FIG. 4.

$F_1 := O_{G_{l_{S_1},r_{S_1}}}(S_1 \cup \{e_{l_{S_1},r_{S_1}}\})$ and $F_2 := O_{G_{l_{S_2},r_{S_2}}}(S_2 \cup \{e_{l_{S_2},r_{S_2}}\})$. It is clear that $V_D(F) \subseteq V_D(F_1) \cup V_D(F_2)$ and $\{v\} = V_D(F_1) \cap V_D(F_2)$. Therefore, $S_1 \in \mathcal{D}^v_{i,q-1}$ and $S_2 \in \mathcal{D}^v_{i+q,j-q}$. This yields

$$\begin{aligned} y(S^v_{i,j}) - |D^v_{i,j}| &= y(S_1) - |D^v_{i,j} \cap [d_i, d_{i+q-1}]| \\ &\quad + y(S_2) - |D^v_{i,j} \cap [d_{i+q}, d_{i+j}]| \\ &\geq y(S^v_{i,q-1}) - |D^v_{i,q-1}| + y(S^v_{i+q,j-q}) - |D^v_{i+q,j-q}| \\ &\geq y^v_{i,q-1} + y^v_{i+q,j-q} \\ &\geq \psi^v_{i,j} \geq y^v_{i,j}. \end{aligned}$$

(2) $d_{S^v_{i,j}}(v) = 1$. We consider three subcases.

(a) $|D^v_{i,j}| \geq 2$. Then, since $S^v_{i,j}$ is edge-minimal, there exists a node $u \in V_D \setminus T_D$ with $d_{S^v_{i,j}}(u) \geq 2$ such that $S^v_{i,j} = W(v,u) \cup S'$, where $W(v,u) \cap S' = \emptyset$. (See Figure 4.) Obviously, $l_{S'} = l$, $r_{S'} = r$ and, since $d_{S^v_{i,j}}(v) = 1$ and $v \in V_D(F)$, we have that $F = W(v,u) \cup O_{G_{l,r}}(S' \cup \{e_{l,r}\})$. Moreover, it is easy to check that S' is an edge-minimal Steiner tree for $D^v_{i,j} \cup \{u\}$ satisfying all further properties in $\mathcal{D}^u_{i,j}$. This together with (1) (note that $d_{S'}(u) \geq 2$) yields

$$\begin{aligned} y(S^v_{i,j}) - |D^v_{i,j}| &= y(W(v,u)) + y(S') - |D^v_{i,j}| \\ &\geq y(W(v,u)) + \psi^u_{i,j} \\ &\geq y^v_{i,j}. \end{aligned}$$

(b) $D^v_{i,j} = \{d_u\}$ for some $u \in \{i, \ldots, i+j-1\}$. Then we know that $S^v_{i,j} \in \mathcal{D}^v_{i,u-i}$,

and we obtain

$$\begin{aligned} y(S^v_{i,j}) - |D^v_{i,j}| &= y(W(v,d_u)) - 1 \\ &\geq y(S^v_{i,u-i}) - |D^v_{i,u-i}| \\ &\geq y^v_{i,u-i} \geq y^v_{i,u-i} + y^v_{u+1,i+j-u-1} \\ &\geq \psi^v_{i,j} \geq y^v_{i,j}. \end{aligned}$$

(c) $D^v_{i,j} = \emptyset$. Here we have that $y(S^v_{i,j}) - |D^v_{i,j}| = 0 \geq \psi^v_{i,j} \geq y^v_{i,j}$.

This completes the proof. □

Let $\beta := \min_{v \in V_D \setminus T_D} y^v_{1,z-1}$ and let S^* be the corresponding edge set. Obviously, $(V_D(S^*), S^*)$ is connected and Lemma 3.3 implies $\beta \leq \alpha$. If $\beta \geq -1$, then there does not exist a violated Steiner partition inequality. If $\beta < -1$, we get $p^* := |V_D(S^*) \cap T_D| \geq 2$, since $y \geq 0$ holds. Thus, $S^* \in \mathcal{D}$ and $\alpha = \beta$. Due to Lemma 3.2 there exists a Steiner partition $V_1, \ldots, V_{p^*}$ with $\delta(V_1, \ldots, V_{p^*}) = S^*$ and $0 > \beta + 1 = y(\delta(V_1, \ldots, V_{p^*})) - p^* + 1$. Therefore, $V_1, \ldots, V_{p^*}$ defines a violated Steiner partition inequality.

This gives rise to the following algorithm.

ALGORITHM 3.4 (Separation algorithm for the Steiner partition inequalities).

Input:

A planar graph $G = (V, E)$, a set of terminals $T \subseteq V$ that are located on the outer face and a vector $y \in \mathbb{R}^E$, $y \geq 0$.

Output:

One of the following possibilities:

- *a violated Steiner partition inequality,*
- *the message "there does not exist a violated Steiner partition inequality."*

(1) *Construct the graph $G_D = (V_D, E)$ with $T_D = \{d_1, \ldots, d_z\}$.*
(2) *Compute shortest paths $W(u, v)$ for all $u, v \in V_D$ such that no inner node of the corresponding paths is an element of T_D.*
(3) *Determine $y^v_{1,z-1}$ for all $v \in V_D \setminus T_D$ using recursion* (3.5).
(4) *Set $\beta := \min_{v \in V_D \setminus T_D} y^v_{1,z-1}$.*
(5) *If $\beta \geq -1$, print the message "there does not exist a violated Steiner partition inequality," STOP.*
(6) *Determine the edge set S^* corresponding to β.*
(7) *Return the violated inequality $(\chi^{S^*})^T x \geq |V_D(S^*) \cap T_D| - 1$.*
(8) *STOP.*

The running time for the execution of steps (3) and (4) of Algorithm 3.4 is bounded by $O(|V_D|^2|T|^2)$. Also note that (3.1) (ii) can be easily taken into account in step (i) of recursion (3.5). Summing up, we obtain the following theorem.

THEOREM 3.5. *Let $G = (V, E)$ be a planar graph and let $T \subseteq V$ be a set of terminals located on the outer face of G. Then the separation problem for the Steiner partition inequalities can be solved in time $O(|V_D|^2|T|^2 + \gamma)$, where γ is the running time for the computation of the shortest paths between all pairs of nodes.*

Let us close this section with two remarks.

From Lemma 3.2 we know that each Steiner tree in G_D for some subset J of T_D corresponds to a Steiner partition inequality. This observation gives rise to several heuristic algorithms for finding violated Steiner partition inequalities. Namely, instead of calculating an optimal Steiner tree in G_D, we determine a Steiner tree heuristically as well. Many heuristics are known for the solution of the minimum Steiner tree problem. (See, for instance, [HRW92] for a survey.) We have implemented one such algorithm that is based on the ideas described in [TM80]. This heuristic starts with a terminal $d \in T_D$. Then, a terminal $d' \in T_D \setminus \{d\}$ is chosen such that the weight of a

shortest path from d' to d is minimal. Finally, d' and d are connected via a shortest path. This scheme is iterated until all terminals are connected. For our purposes this procedure is slightly modified. First, we have to make sure that no inner node on the corresponding shortest paths is an element of T_D. Second, in order to generate as many inequalities as possible, we compute a Steiner tree starting with all pairs of nodes d_i, d_j, where $d_i, d_j \in T_D$. The advantage of this heuristic is that not only the final Steiner trees define Steiner partition inequalities, but also any of its iteratively computed subtrees defines a Steiner partition inequality (cf. Lemma 3.2). By working in this scheme we obtain plenty of inequalities. For each of them we check whether it is violated. We will see in the last section that this heuristic works very well for our problem instances.

Finally, let us point out that Algorithm 3.4 can also be used to solve certain multicut problems. Suppose there is given a planar graph G, a set of nodes $T \subseteq V$ located on the outer face of G, and nonnegative edge weights $w_e, e \in E$, and we want to determine $\min\{-\lambda, \min\{w(\delta(V_1, \ldots, V_p)) - \lambda p \mid V_1, \ldots, V_p, p \geq 2$ is a Steiner partition of V with respect to T such that $G(V_1, \ldots, V_p)$ is 2-node connected$\}\}$, where λ is the gain for each element of the partition. By applying some modifications to Algorithm 3.4 this problem can be solved in polynomial time as well.

4. Separation of the alternating cycle inequalities and extensions. We first introduce the so-called alternating cycle inequalities. Let $G = (V, E)$ be a graph and $\mathcal{N} = \{T_1, T_2\}$ a net list. We call a cycle F in G an *alternating cycle with respect to* T_1, T_2 if $F \subseteq [T_1 : T_2]$ and $V(F) \cap T_1 \cap T_2 = \emptyset$. (See Figure 5.) Moreover, let $F_1 \subseteq E(T_2)$ and $F_2 \subseteq E(T_1)$ be two sets of diagonals of the alternating cycle F with respect to T_1, T_2. The inequality

$$(\chi^{E\setminus(F\cup F_1)}, \chi^{E\setminus(F\cup F_2)})^T x \geq \tfrac{1}{2}|F| - 1$$

is called an *alternating cycle inequality.*

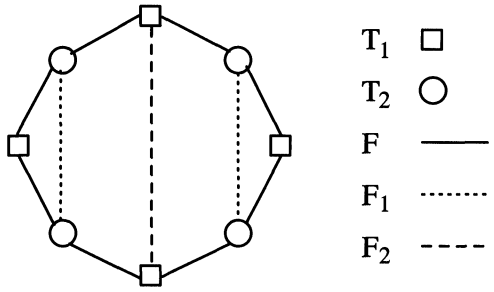


FIG. 5.

The basic form of an alternating cycle inequality, i.e., $F_1 = F_2 = \emptyset$, is valid for STP $(G, \mathcal{N}, \mathbf{1})$, because whenever two terminals of net 1, say, are connected on the cycle, at least one node of the other net is isolated. This means that in order to connect both nets simultaneously, at least $|V(F) \cap T_1| - 1 = |V(F) \cap T_2| - 1 = \frac{1}{2}|F| - 1$ edges not contained in the cycle must be used. In general, the basic form of an alternating cycle inequality is not facet defining. The sets F_1 and F_2 are used to strengthen the basic form; in fact, choosing them appropriately, we can obtain facet-defining inequalities.

The sets of diagonals $F_1 \subseteq E(T_2)$ and $F_2 \subseteq E(T_1)$ are called *maximal cross-free with respect to* F if F_1 and F_2 are cross-free and each diagonal $e_1 \in E(T_1) \setminus F_2$ crosses F_1 and each diagonal $e_2 \in E(T_2) \setminus F_1$ crosses F_2. (See Figure 5.) The following theorem then holds.

THEOREM 4.1. *Let $G = (V, E)$ be a graph that contains the complete graph on node set V as a subgraph and let $\mathcal{N} = \{T_1, T_2\}$ be a disjoint net list with $T_1 \cup T_2 = V$ and $|T_1| = |T_2| = l$, $l \geq 2$. Furthermore, let F be an alternating cycle with respect to T_1, T_2 with $V(F) = V$ and $F_1 \subseteq E(T_2)$, $F_2 \subseteq E(T_1)$. Then, the alternating cycle inequality*

$$(\chi^{E\setminus(F\cup F_1)}, \chi^{E\setminus(F\cup F_2)})^T x \geq l - 1$$

defines a facet of $STP(G, \mathcal{N}, \mathbf{1})$ if and only if F_1 and F_2 are maximal cross-free.

There is a natural way to extend the alternating cycle inequalities as follows. Let $G = (V, E)$ be a graph and $\mathcal{N} = \{T_1, T_2\}$ be a net list. Let $V_1, \ldots, V_k$ be a partition of V with $k \geq 4$ and k even such that the following properties are satisfied:

$$\begin{array}{ll} \text{(i)} & (V_i, E(V_i)) \text{ is connected for } i = 1, \ldots, k, \\ \text{(ii)} & V_{2i+1} \cap T_1 \neq \emptyset,\ V_{2i+1} \cap T_2 = \emptyset \text{ for } i = 0, \ldots, \frac{k}{2} - 1, \\ & V_{2i} \cap T_1 = \emptyset,\ V_{2i} \cap T_2 \neq \emptyset \text{ for } i = 1, \ldots, \frac{k}{2}, \\ \text{(iii)} & [V_i : V_{i+1}] \neq \emptyset \text{ for } i = 1, \ldots, k. \end{array} \tag{4.1}$$

(An index $i > k$ is identified with the index $((i-1) \text{ modulo } k) + 1$.) Condition (iii) guarantees that the contracted graph $G(V_1, \ldots, V_k)$ (i.e., the graph obtained by contracting every element of the partition to a single node) contains at least one hamiltonian cycle. We choose an edge set $F \subseteq \cup_{i=1}^{k} [V_i : V_{i+1}]$ in G that forms a hamiltonian cycle in $G(V_1, \ldots, V_k)$. Note that, due to (ii), F is alternating. Furthermore, let $F_1 \subseteq \bigcup_{\substack{i,j \text{ even} \\ i \neq j}} [V_i : V_j]$, $F_2 \subseteq \bigcup_{\substack{i,j \text{ odd} \\ i \neq j}} [V_i : V_j]$ be two edge sets such that F_1 and F_2, viewed as edge sets in the contracted graph $G(V_1, \ldots, V_k)$, are cross-free with respect to the alternating cycle F. Then we call the following inequality an *extended alternating cycle inequality*:

$$(\chi^{E\setminus(F\cup F_1)}, \chi^{E\setminus(F\cup F_2)})^T x \geq \frac{k}{2} - 1.$$

This inequality is valid with respect to STP $(G, \mathcal{N}, \mathbf{1})$ due to Lemma 2.1. Let us give an example.

Example 4.2. Consider the graph G in Figure 6(a) with $T_1 = \{1, 3, 5, 10\}$ and $T_2 = \{4, 9, 12\}$. It can easily be checked that the partition V_1, V_2, V_3, V_4 satisfies (4.1); the corresponding contracted graph $G(V_1, V_2, V_3, V_4)$ is depicted in Figure 6(b). Obviously, $F := \{\{3,4\}, \{10,11\}, \{9,10\}, \{5,9\}\}$ is a hamiltonian alternating cycle in $G(V_1, V_2, V_3, V_4)$ and $F_1 := \emptyset$, $F_2 := \{\{2,6\}, \{5,6\}\}$ are cross-free sets of diagonals. Thus, the inequality $x_{26}^1 + x_{56}^1 + x_{37}^1 + x_{67}^1 + x_{37}^2 + x_{67}^2 \geq 1$ is an extended alternating cycle inequality.

Observe that when $V_1, \ldots, V_k$ are chosen, we have the freedom to pick F, F_1, and F_2 from among many possible alternatives. We call any triple (F, F_1, F_2) that satisfies the additional requirements defined above a *feasible triple (for $V_1, \ldots, V_k$)*.

We do not know under which conditions the extended alternating cycle inequalities define facets of the Steiner tree packing polyhedron and we do not know how to separate these inequalities in the general case. Our aim here is to outline a separation routine for extended alternating cycle inequalities in the (practically relevant) case where a planar graph G is given and all terminals of T_1 and T_2 are on the outer face.

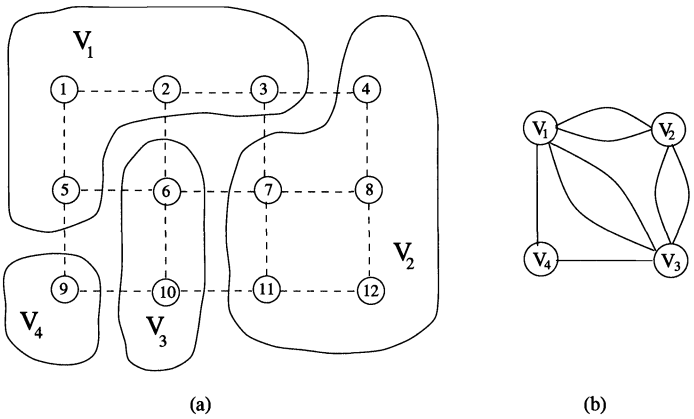


FIG. 6.

We proceed in a similar way as for the Steiner partition inequalities. We show that for each partition $V_1, \ldots, V_k$ satisfying (4.1), the multicut $\delta(V_1, \ldots, V_k)$ corresponds to a certain Steiner tree in a graph that remains to be defined. Here an additional difficulty comes up, since the edges of $\delta(V_1, \ldots, V_k)$ must be evaluated differently. The coefficients depend on the choice of the alternating cycle F in $G(V_1, \ldots, V_k)$ and on the sets F_1 and F_2. Thus, for the corresponding Steiner tree, a vector must be defined that "sifts" the edges that correspond to F, F_1, or F_2, respectively.

Without loss of generality we suppose the planar graph G to be 2-node connected so that the edge set $\mathrm{O}_G(E)$ that encloses the outer face is a cycle. Let $T = T_1 \cup T_2$ and we may assume that $T := \{t_1, \ldots, t_z\}$ is numbered in a clockwise fashion along this cycle. Let us consider the dual graph $G^* = (V^*, E)$ of G. We split the node representing the outer face into z nodes $d_1, \ldots, d_z$ such that every edge of $\mathrm{O}_G(E)$ that is passed by walking from t_i to t_{i+1} on $\mathrm{O}_G(E)$ in clockwise order is incident to d_{i+1} for $i = 1, \ldots, z$. Let $G_D = (V_D, E)$ denote the resulting graph and set $T_D = \{d_1, \ldots, d_z\}$. Figure 7 illustrates this construction for the graph of Example 4.2. Set

$$
\begin{aligned}
M := \quad & \{d_i \in T_D \mid \{t_{i-1}, t_i\} \cap T_k \neq \emptyset \text{ for } k = 1, 2\}, \\
\mathcal{S} := \quad & \{S \subseteq E \mid S \text{ is an edge-minimal Steiner tree for } M \text{ in } G_D \text{ such that} \\
& \qquad d_S(d) = 1 \text{ for all } d \in M \text{ and } d_S(d) = 0 \text{ for all } d \in T_D \setminus M\}.
\end{aligned}
$$

The following relation then holds.

LEMMA 4.3. *Let $G = (V, E)$ be a planar graph and $\mathcal{N} = \{T_1, T_2\}$ where all terminals are located on the outer face. Then the following statements are true:*

1. *If $V_1, \ldots, V_k$ is a partition of V satisfying* (4.1)*, then the corresponding multicut $\delta(V_1, \ldots, V_k)$ viewed as an edge set of the dual graph G_D is a Steiner tree contained in $\mathcal{S}$.*
2. *If S is a Steiner tree in G_D contained in $\mathcal{S}$ and $|M| \geq 4$ holds, then there exists a partition $V_1, \ldots, V_k$ of V satisfying* (4.1) *such that $S = \delta(V_1, \ldots, V_k)$.*

Proof. Let us prove the first statement. Suppose $V_1, \ldots, V_k$ with $k \geq 4$ even is a partition satisfying (4.1). We first observe that $V_1, \ldots, V_k$ is a Steiner partition with respect to T. Moreover, (4.1) (iii) implies (3.1) (iii) and (4.1) (i) is identical to (3.1) (i). Thus, Lemma 3.2 implies that $S := \delta(V_1, \ldots, V_k)$ is an edge-minimal

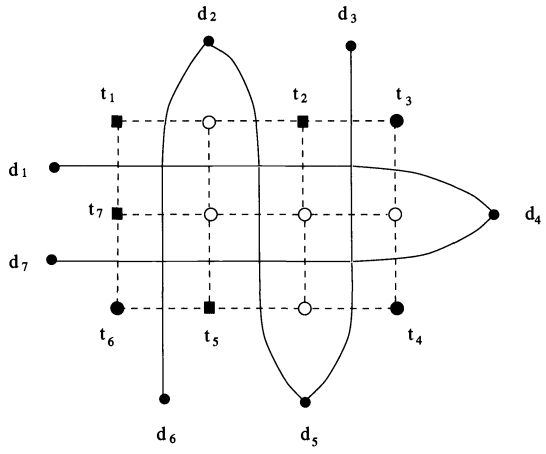


FIG. 7.

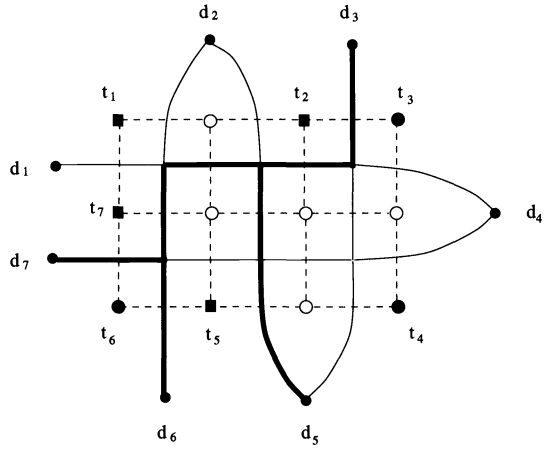


FIG. 8.

Steiner tree in G_D for some $J \subseteq T$ with $|J| = k$, satisfying $d_S(j) = 1$ for $j \in J$ and $d_S(t) = 0$ for all $t \in T_D \setminus J$. Hence, it only remains to be shown that $J = M$. But this immediately follows from properties (4.1) (ii) and (4.1) (iii) together with the fact that k is even.

Conversely, let $S \in \mathcal{S}$ and suppose $|M| \geq 4$. We can apply Lemma 3.2 again, since $\mathcal{S} \subseteq \mathcal{D}$, and thus $S \in \mathcal{D}$. Lemma 3.2 implies that there exists a unique partition $V_1, \ldots, V_{|M|}$ with respect to T satisfying (3.1) (i) and (3.1) (iii). Since G is planar and all terminals lie on the outer face, $G(V_1, \ldots, V_p)$ is also outerplanar. Thus, we can assume that $V_1, \ldots, V_{|M|}$ is numbered in clockwise or anticlockwise (without loss of generality clockwise) order around the outer face and that $V_1 \cap T_1 \neq \emptyset$. The fact that $G(V_1, \ldots, V_{|M|})$ is outerplanar and 2-node connected implies $[V_i : V_{i+1}] \neq \emptyset$ for $i = 1, \ldots, |M|$, proving (4.1) (iii). (3.1) (i) is identical to (4.1) (i). By construction,

$V_1 \cap T_1 \neq \emptyset$. So, we obtain property (4.1) (ii) from the fact that S is a Steiner tree for M with $V_D(S) \cap (T_D \setminus M) = \emptyset$. □

In Figure 8 the Steiner tree $S \in \mathcal{S}$ which corresponds to the partition V_1, V_2, V_3, V_4 shown in Figure 6 (a) is depicted in thick solid lines. From the proof of Lemma 4.3 we see that the cardinality k of a partition $V_1, \ldots, V_k$ satisfying (4.1) equals $|M|$. So, we suppose from now on that $k = |M| \geq 4$; otherwise, there does not exist any extended alternating cycle inequality.

Next, we define a "sifting function" for each $S \in \mathcal{S}$. Let $V_1^S, \ldots, V_k^S$ be the corresponding partition satisfying (4.1) according to Lemma 4.3. We call a vector $a \in \{0,1\}^{\{T_1,T_2\}\times E}$ a *sifting for* S if there exists a feasible triple (F, F_1, F_2) for $V_1^S, \ldots, V_k^S$ such that $a = (\chi^{E\setminus(F\cup F_1)}, \chi^{E\setminus(F\cup F_2)})$. Moreover, set $\eta(S) := \frac{k}{2}$ and let $\mathcal{F}(S)$ denote the set of all siftings for S. Consider now the minimization problem

$$\min_{S\in\mathcal{S}} \min_{a\in\mathcal{F}(S)} y^1(S \cap I_{a^1}) + y^2(S \cap I_{a^2}) - \eta(S) \tag{4.2}$$

where $y \in \mathbb{R}^{\{T_1,T_2\}\times E}$, $y \geq 0$ is the vector to be cut off and $I_b = \{e \in E \mid b_e = 1\}$ for $b \in \{0,1\}^E$. Let μ denote the minimum value of (4.2). From Lemma 4.3 and the definition of a sifting we know that we can solve the separation problem for the extended alternating cycle inequalities via computing (4.2). If $\mu \geq -1$, there obviously exists no violated inequality. If $\mu < -1$, let $\tilde{S} \in \mathcal{S}$ and $\tilde{a} \in \mathcal{F}(S)$ be such that $\mu = y^1(\tilde{S} \cap I_{\tilde{a}^1}) + y^2(\tilde{S} \cap I_{\tilde{a}^2}) - \eta(\tilde{S})$. In this case $(\chi^{\tilde{S}\cap I_{\tilde{a}^1}}, \chi^{\tilde{S}\cap I_{\tilde{a}^2}}) \geq \eta(\tilde{S}) - 1$ is an extended alternating cycle inequality that is violated by y. Thus, it remains to solve problem (4.2).

As in the case of the Steiner partition inequalities we develop a dynamic program in order to compute the optimal Steiner tree $\tilde{S} \in \mathcal{S}$ and the optimal sifting $\tilde{a} \in \mathcal{F}(\tilde{S})$. Consider the following recursion.

RECURSION 4.4. *Let $y^k(u,v)$ denote the value of a shortest path from u to v with respect to the weighting y^k whose inner nodes have empty intersection with T_D. Moreover, let $y(u,v)$ correspond to the value of a shortest path from u to v with respect to the weight function $y^1 + y^2$ whose inner nodes have an empty intersection with T_D. Finally, define $y_{-1}(u,v) := \min\{\hat{y}(W) - \max_{e\in W} \hat{y}_e \mid W$ is a path from u to v whose inner nodes have empty intersection with $T_D\}$ where $\hat{y} = y^1 + y^2$. Then for all $i = 1, \ldots, z$, $j = 0, \ldots, z-1$, $v \in V_D \setminus T_D$ and $k = 1, 2$ (with $\bar{k} = 1$, if $k = 2$, and $\bar{k} = 2$, if $k = 1$), set*

$$\begin{aligned}
&(1)\quad l_1(v,i,0) := y_{-1}(v,d_i) - \tfrac{1}{2} && \text{if } d_i \in M;\\
&\qquad\ l_1(v,i,0) := 0 && \text{if } d_i \in T_D \setminus M;\\
&(2)\quad l_2(v,i,j) := \min_{1\leq r\leq j} l_1(v,i,r-1) + l_1(v,i+r,j-r)\\
&(3)\quad l_1(v,i,j) := \min_{u\in V_D\setminus T_D} y^k(v,u) + l_2(u,i,j) && \text{if } t_{i-1}, t_{i+j} \in T_k;\\
&\qquad\ l_1(v,i,j) := \min_{u\in V_D\setminus T_D} y(v,u) + l_2(u,i,j) && \text{if } t_{i-1} \in T_k,\ t_{i+j} \in T_{\bar{k}}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad && \text{or } t_{i-1} \in T_{\bar{k}},\ t_{i+j} \in T_k.
\end{aligned}$$

In principle, (1), (2), and (3) correspond to the formulas in (3.5) (i), (ii), and (iii), respectively. (1) in Recursion 4.4 takes into account that the resulting Steiner tree S satisfies $d_S(d) = 1$ for all $d \in M$ and $d_S(d) = 0$ for all $d \in T_d \setminus M$; see the definition of $\mathcal{S}$. (2) is precisely the formula (ii) of (3.5) and the subcases in (3) arise because the evaluation of the path from v to u depends on the terminal set to which t_{i-1} and t_{i+j} belong. The following theorem then holds.

THEOREM 4.5. *For the value* $l_{\min} := \min_{v \in V_D \setminus T_D} l_1(v, 1, z-1)$ *computed via Recursion* 4.4, *we have*

$$l_{\min} \leq \min_{S \in \mathcal{S}} \min_{a \in \mathcal{F}(S)} y^1(S \cap I_{a^1}) + y^2(S \cap I_{a^2}) - \eta(S).$$

Proof. We will proceed as in the last chapter and prove a relation between solutions for subproblems in the spirit of Lemma 3.3. Hereto we need some further notation.

First, we have to define subtrees of Steiner trees in $\mathcal{S}$. For all $i = 1, \ldots, z$, $j = 0, \ldots, z-1$, we use the same notation and definitions as in the previous section, i.e., the interval $[d_i, d_{i+j}]$, the edge $e_{i,j}$ embedded in the outer face of G_D, the graph $G_{i,j} := (V_D, E \cup \{e_{i,j}\})$, and, for a Steiner tree in G_D for $J \subseteq [d_i, d_{i+j}]$, $J \neq \emptyset$, the symbols l_S and r_S. For $v \in V_D \setminus T_D$ and $i = 1, \ldots, z$, $j = 0, \ldots, z-1$, we define $\mathcal{S}^v_{i,j}$ as the set of all edge-minimal Steiner trees S in G_D for $([d_i, d_{i+j}] \cap M) \cup \{v\}$ such that

$d_S(d) = 1$ for all $d \in [d_i, d_{i+j}] \cap M$,
$d_S(d) = 0$ for all $d \in T_D \setminus ([d_i, d_{i+j}] \cap M)$,
$v \in V_D(\mathrm{O}_{G_{l_S, r_S}}(S \cup \{e_{l_s, r_S}\}))$ if $[d_i, d_{i+j}] \cap M \neq \emptyset$.

Furthermore, we call a vector $a \in \{0,1\}^{\{T_1, T_2\} \times E}$ a sifting for $S \in \mathcal{S}^v_{i,j}$ if there exists a Steiner tree $\tilde{S} \in \mathcal{S}$ and a sifting $\tilde{a}$ for $\tilde{S}$ such that $S \subseteq \tilde{S}$, $a_e = \tilde{a}_e$ for all $e \in S$ and $a_e = 0$ otherwise.

Let $\mathcal{F}(S)$ denote the set of siftings for $S \in \mathcal{S}^v_{i,j}$ and set $\eta(S) := \frac{1}{2}|[d_i, d_{i+j}] \cap M|$.

What we want to show is that $l_1(v, i, j)$ is a lower bound for $y^1(S \cap I_{a^1}) + y^2(S \cap I_{a^2}) - \eta(S)$ for all Steiner trees $S \in \mathcal{S}^v_{i,j}$ and all siftings $a \in \mathcal{F}(S)$. Unfortunately, this is not true, in general. It turns out that depending on the node v, certain siftings must be excluded.

Let $S \in \mathcal{S}^v_{i,j}$. Define $V^3_S := \{w \in V_D(S) \mid d_S(w) \geq 3\}$. Let $L \subseteq M \cap [d_i, d_{i+j}]$ be the set of nodes d such that for the unique path P in S from v to d, $V_D(P) \cap V^3_S = \emptyset$ holds. If $L = \emptyset$, we set $\mathcal{F}_v(S) := \mathcal{F}(S)$. Otherwise, let P_d, $d \in L$, denote the unique path from v to d. We set $\mathcal{F}_v(S) := \{a \in \mathcal{F}(S) \mid \text{for all } d \in L \text{ there exists an edge } e \in S \cap P_d \text{ with } a^1_e = a^2_e = 0\}$. In other words, we allow only siftings where the corresponding alternating cycle has nonempty intersection with all paths P_d, $d \in L$.

$$l_1(v, i, j) \leq \min_{S \in \mathcal{S}^v_{i,j}} \min_{a \in \mathcal{F}_v(S)} y^1(S \cap I_{a^1}) + y^2(S \cap I_{a^2}) - \eta(S) \tag{4.3}$$

for all $v \in V_D \setminus T_D$, $i = 1, \ldots, z$, $j = 0, \ldots, z-1$.

We prove this by induction on j. (4.3) is obviously true for $j = 0$. Now, suppose (4.3) is also true for all $l = 0, \ldots, j-1$. Consider any arbitrary $v \in V_D \setminus T_D$ and $i \in \{1, \ldots, z\}$. Let $\tilde{S} \in \mathcal{S}^v_{i,j}$ and $\tilde{a} \in \mathcal{F}_v(\tilde{S})$ such that

$$y^1(\tilde{S} \cap I_{\tilde{a}^1}) + y^2(\tilde{S} \cap I_{\tilde{a}^2}) - \eta(\tilde{S}) = \min_{S \in \mathcal{S}^v_{i,j}} \min_{a \in \mathcal{F}_v(S)} y^1(S \cap I_{a^1}) + y^2(S \cap I_{a^2}) - \eta(S).$$

We have to show that $l_1(v, i, j) \leq y^1(\tilde{S} \cap I_{\tilde{a}^1}) + y^2(\tilde{S} \cap I_{\tilde{a}^2}) - \eta(\tilde{S})$. We distinguish two cases.

(1) $d_{\tilde{S}}(v) \geq 2$. Since G_D is planar, all terminals of T_D are located on the outer face of G_D and $v \in V_D(\mathrm{O}_{G_{l_{\tilde{S}}, r_{\tilde{S}}}}(\tilde{S} \cup \{e_{l_{\tilde{S}}, r_{\tilde{S}}}\}))$; there exists an index $r \in \{1, \ldots, j\}$ and two disjoint subtrees S_1, S_2 of $\tilde{S}$ such that

$S_1 \cup S_2 = \tilde{S}$,

$v \in V_D(S_1)$, $v \in V_D(S_2)$,
$S_1 \in \mathcal{S}^v_{i,r-1}$ and $S_2 \in \mathcal{S}^v_{i+r,j-r}$. (See also the proof of Lemma 3.3.)

For $k = 1, 2$, we define values a^k_e and b^k_e as follows:

$$\begin{array}{lll} a^k_e := & \tilde{a}^k_e & \text{if } e \in E \setminus S_2, \\ a^k_e := & 0 & \text{if } e \in S_2, \\ b^k_e := & \tilde{a}^k_e & \text{if } e \in E \setminus S_1, \\ b^k_e := & 0 & \text{if } e \in S_1. \end{array}$$

Next we show that $a \in \mathcal{F}_v(S_1)$ and $b \in \mathcal{F}_v(S_2)$. Since $\tilde{a} \in \mathcal{F}(\tilde{S})$, we know that $a \in \mathcal{F}(S_1)$ and $b \in \mathcal{F}(S_2)$. Let $\tilde{L} \subseteq M \cap [d_i, d_{i+j}]$ denote the set of nodes d such that for the unique path P from v to d holds $V_D(P) \cap V^3_{\tilde{S}} = \emptyset$. Denote by L_1 and L_2 the corresponding node sets of S_1 and S_2. From the fact that $\tilde{L} = L_1 \cup L_2$ we conclude that $a \in \mathcal{F}_v(S_1)$ and $b \in \mathcal{F}_v(S_2)$. Finally, note that $\eta(\tilde{S}) = \eta(S_1) + \eta(S_2)$. Summing up, we obtain that

$$\begin{aligned} y^1(\tilde{S} \cap I_{\tilde{a}^1}) + y^2(\tilde{S} \cap I_{\tilde{a}^2}) - \eta(\tilde{S}) &= y^1(S_1 \cap I_{a^1}) + y^2(S_1 \cap I_{a^2}) - \eta(S_1) \\ &\quad + y^1(S_2 \cap I_{b^1}) + y^2(S_2 \cap I_{b^2}) - \eta(S_2) \\ &\geq l_1(v, i, r-1) + l_1(v, i+r, j-r) \\ &\geq l_2(v, i, j) \geq l_1(v, i, j). \end{aligned}$$

(2) $d_{\tilde{S}}(v) = 1$. If $|V_D(\tilde{S}) \cap [d_i, d_{i+j}]| = 1$, we conclude that there exists an $r \in \{1, \ldots, j\}$ such that $\tilde{S} \in \mathcal{S}^v_{i,r-1}$ and $\emptyset \in \mathcal{S}^v_{i+r,j-r}$, or vice versa, $\emptyset \in \mathcal{S}^v_{i,r-1}$ and $\tilde{S} \in \mathcal{S}^v_{i+r,j-r}$. (Note that $j > 0$.) Since both $r-1$ and $j-r$ are less than or equal to $j-1$, we conclude by the assumption of the induction that (4.3) holds.

Now, suppose $|V_D(\tilde{S}) \cap [d_i, d_{i+j}]| \geq 2$. Then there exists a node $u \in V^3_{\tilde{S}}$ such that $\tilde{S} = W(v,u) \cup S'$, $W(v,u) \cap S' = \emptyset$, with $d_{S'}(u) \geq 2$ and $S' \in \mathcal{S}^u_{i,j}$, where $W(v,u)$ is the unique path in $\tilde{S}$ from v to u. (For a more detailed discussion, see the corresponding case in the proof of Lemma 3.3.) Set $a^k_e := \tilde{a}^k_e$ for all $e \in E \setminus W(v,u)$ and $a^k_e := 0$ for all $e \in W(v,u)$, $k = 1, 2$. Obviously, $\eta(\tilde{S}) = \eta(S')$. Since $\tilde{a} \in \mathcal{F}_v(\tilde{S})$ we obtain that $a \in \mathcal{F}_u(S')$. Moreover, $\tilde{a} \in \mathcal{F}_v(\tilde{S})$ and $v \in V_D(\mathrm{O}_{G_{l_{\tilde{S}}, r_{\tilde{S}}}}(\tilde{S} \cup \{e_{l_{\tilde{S}}, r_{\tilde{S}}}\}))$ imply that for all $e \in W(v,u)$, $\tilde{a}^k_e = 1$ if $t_{i-1} \in T_k$ or $t_{i+j} \in T_k$ for $k = 1, 2$. Thus, taking the correctness of case (1) into account we get the following.

If $t_{i-1}, t_{i+j} \in T_k$ for some $k \in \{1, 2\}$, we obtain that

$$\begin{aligned} y^1(\tilde{S} \cap I_{\tilde{a}^1}) + y^2(\tilde{S} \cap I_{\tilde{a}^2}) - \eta(\tilde{S}) &\geq y^k(W(v,u)) \\ &\quad + y^1(S' \cap I_{a^1}) + y^2(S' \cap I_{a^2}) - \eta(S') \\ &\geq y^k(W(v,u)) + l_2(u, i, j) \\ &\geq l_1(v, i, j). \end{aligned}$$

If $t_{i-1} \in T_1$ and $t_{i+j} \in T_2$ or $t_{i-1} \in T_2$ and $t_{i+j} \in T_1$, we have that

$$\begin{aligned} y^1(\tilde{S} \cap I_{\tilde{a}^1}) + y^2(\tilde{S} \cap I_{\tilde{a}^2}) - \eta(\tilde{S}) &\geq \hat{y}(W(v,u)) \\ &\quad + y^1(S' \cap I_{a^1}) + y^2(S' \cap I_{a^2}) - \eta(S') \\ &\geq \hat{y}(W(v,u)) + l_2(u, i, j) \\ &\geq l_1(v, i, j). \end{aligned}$$

We conclude that (4.3) is true. Relation (4.3) finally implies that

$$\begin{aligned} l_{\min} &= \min_{v \in V_D \setminus T_D} l_1(v, 1, z-1) \\ &\leq \min_{v \in V_D \setminus T_D} \min_{S \in \mathcal{S}^v_{1,z-1}} \min_{a \in \mathcal{F}_v(S)} y^1(S \cap I_{a^1}) + y^2(S \cap I_{a^2}) - \eta(S) \\ &\leq \min_{S \in \mathcal{S}} \min_{a \in \mathcal{F}(S)} y^1(S \cap I_{a^1}) + y^2(S \cap I_{a^2}) - \eta(S). \end{aligned}$$

This completes the proof. □

An edge set S_l that attains the minimum value $l_{\min}$ can easily be obtained by labeling the corresponding edges in Recursion 4.4. Per construction, S_l is an element of $\mathcal{S}$. Note that the Recursion 4.4 formulas implicitly define a vector $a_l \in \{0,1\}^{\{T_1,T_2\} \times E}$ such that $l_{\min} = y^1(S_l \cap I_{a_l^1}) + y^2(S_l \cap I_{a_l^2}) - \eta(S_l)$. If $l_{\min} \geq -1$, we conclude from Theorem 4.5 that there does not exist a violated extended alternating cycle inequality. If $l_{\min} < -1$, a_l is not necessarily a sifting of S_l.

Example 4.6. Consider the example depicted in Figure 9. Given a complete rectangular 3×3 grid graph, the terminals of net 1 are printed as small black rectangles and those of net 2 as small black circles. All other nodes are depicted as white circles. The solid lines represent edges e with value $y_e^1 = 1$, dashed lines edges e having value $y_e^2 = 0.5$, and dotted lines edges e with value $y_e^1 = 0.5$. The edge set in G_D yielding $l_{\min}$ is drawn in thick black lines. $l_{\min}$ results from the following computation: $l_{\min} = l_1(4,4,0) + l_1(4,3,0) + l_1(4,1,1) = (y_{-1}(4,d_4) - \frac{1}{2}) + (y_{-1}(4,d_3) - \frac{1}{2}) + (y^1(4,2) + l_1(2,1,0) + l_1(2,2,0)) = (0.0 - \frac{1}{2}) + (0.0 - \frac{1}{2}) + (0.5 + (0.0 - \frac{1}{2}) + (0.0 - \frac{1}{2})) = -1.5$. The branching nodes of the recursion are the nodes 4 and 2; i.e., edge $\{1,2\}$ is counted twice. The branching nodes of the edge set S_l are the nodes 4 and 1. Therefore, a_l does not define a sifting for S_l. However, a_l can be modified to a sifting $\bar{a}_l$ for S_l. In this case we obtain $y^1(S_l \cap F_{\bar{a}_l^1}) + y^2(S_l \cap F_{\bar{a}_l^2}) - \alpha(S_l) = -1.0$ implying that the corresponding cycle inequality is not violated.

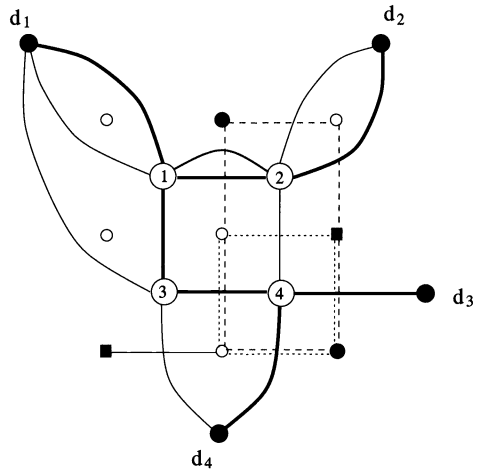


FIG. 9.

We do not see how to avoid these cases. We have no alternative but to check whether a_l actually is an element of $\mathcal{F}(S_l)$. Thus, the proposed algorithm is not an exact separation method. However, it provides a lower bound for the slack of the

most violated extended alternating cycle inequality and, if $l_{\min} \geq -1$, a proof that no extended alternating cycle inequality exists that is violated by y.

Clearly, the recursion itself for computing $l_{\min}$ takes time $O(|V_D|^2(|T_1| + |T_2|)^2)$. The values $y(u,v)$ and $y^k(u,v)$, $k \in \{1,2\}$, can be determined by any shortest path algorithm. However, at first sight it is not obvious how to compute the values $y_{-1}(u,v)$. It turns out that these values can be calculated by calling a shortest path algorithm twice. This will be the topic of the following subsection. Thus, the overall running time of our algorithm is $O(|V_D|^2(|T_1| + |T_2|)^2 + \gamma)$, where γ is the time to compute the shortest paths between all pairs of nodes.

Finally, let us remark that in our cutting plane algorithm for computing a minimum weight Steiner tree packing, we do not only try to determine the most violated extended alternating cycle inequality by using Recursion 4.4. Instead, we also compute Steiner trees for M in G_D heuristically. Again, we use the algorithm proposed by [TM80] with cost function $y^1 + y^2$. Thereafter, we determine the best possible sifting for the resulting Steiner trees.

Up to now, all partitions $V_1, \ldots, V_k$ of V considered in our separation algorithm have cardinality $k = |M|$. In other words, each node in the contracted graph $G(V_1, \ldots, V_k)$ is part of the alternating cycle. It is natural to generalize this in the sense that not all elements of the partition $V_1, \ldots, V_k$ belong to the alternating cycle but are viewed as certain additional nodes in the contracted graph. We have analyzed this generalization from a theoretical point of view and have identified conditions under which the resulting inequalities are facet-defining. (See [GMW92a].) Moreover, the dynamic program described above can be adapted to this generalization. It also provides a lower bound for the slack of the most violated inequality. A detailed description of this algorithm requires many technicalities that we do not want to present here. For a discussion of this separation algorithm we refer to [M92].

5. Determining cheapest paths with cost-free edges. The following combinatorial optimization problem is an interesting variant of the shortest path problem. We are given a graph $G = (V, E)$ with costs $c_e \geq 0$ for all edges $e \in E$, two nodes $s, t \in V$, and a nonnegative integer k. We want to find a cheapest path from s to t where the "cost" of a path is the usual cost minus the sum of the costs of the k (or at most k) most expensive edges of the path. Another way to view the problem is the following. We have k tokens that allow us to use k (or at most k) edges for free. We want to choose a path from s to t and employ the k tokens to use k (or at most k) edges without any costs in such a way that the total sum paid for the use of the remaining edges is as small as possible. Clearly, this problem also has a directed version; we can similarly search for odd or even paths or cycles where k of the edges can be used for free.

We are particularly interested in the case of $[s,t]$-paths where $k = 1$, since the computation of cheapest paths with one cost-free edge is necessary to compute the values (1) $l_1(v,i,0) = y_{-1}(v,d_i) - \frac{1}{2}$ in Recursion 4.4. There is an obvious way to determine a cheapest $[s,t]$-path with one cost-free edge. For every edge $e \in E$, we do the following: we define a new cost function by setting $c_f^e := c_f$, if $f \neq e$, and $c_e^e := 0$, and we compute a shortest $[s,t]$-path in G with cost function c^e. Every shortest of the $|E|$ $[s,t]$-paths determined this way is a cheapest $[s,t]$-path with one cost-free edge. (This process can be clearly generalized to the case $k \geq 1$.) However, a cheapest $[s,t]$-path with one cost-free edge can be computed faster by calling a shortest path algorithm only twice as follows.

ALGORITHM 5.1 (Cheapest paths from s to all other nodes with one cost-free edge).

Input:

A graph $G = (V, E)$, edge costs $c_e \geq 0$, $e \in E$ and a node $s \in V$.

Output:

The costs of cheapest paths from s to all $v \in V$ with one cost-free edge.

Datastructures:

$d(v)$ = *Length of a shortest $[s, v]$-path.*
$m(v)$ = *Cost of a cheapest path from s to v with one cost-free edge.*
N = *List of unlabeled nodes that are incident to some labeled node.*

(1) *Compute $d(v)$ for all $v \in V$ using a shortest path algorithm.*
(2) *Initialize $m(v) = d(v)$ for all $v \in V$.*
(3) *Set $m(s) = 0$ and $N = \emptyset$.*
Label s. (All other nodes are supposed to be unlabeled.)
For all nodes v adjacent to s set
$m(v) = 0$ *and* $N = N \cup \{v\}$.
(4) *As long as there exists an unlabeled node, perform the following steps:*
(5) *Determine a node $v \in N$ with $m(v) = \min\{m(u) \mid u \in N\}$.*
(6) *Label v and set $N = N \setminus \{v\}$.*
(7) *For all nodes u adjacent to v perform the following steps:*
If $\min\{m(v) + c_{vu}, d(v)\} < m(u)$, *set*
$m(u) = \min\{m(v) + c_{vu}, d(v)\}$.
$N = N \cup \{u\}$.
(8) *Return the values $m(v)$ for all $v \in V$.*
(9) *STOP.* The following theorem states the correctness of the algorithm.

THEOREM 5.2. *Let $G = (V, E)$ be a graph with nonnegative edge costs c_e, $e \in E$. Then, Algorithm* 5.1 *determines the cheapest path from s to v with one cost-free edge for all $v \in V$.*

Proof. To avoid confusion we use the following notation throughout this proof: for a path P from u to v, we denote by $c(P) = \sum_{e \in P} c_e$ the "length" of path P and use the term "shortest" if $c(P)$ is minimum among all $[u, v]$-paths. On the other hand, for a path P from u to v with one cost-free edge, we call the value $c(P) - \max_{e \in P} c_e$ the "cost" of path P and speak of a "cheapest" path P if the value $c(P) - \max_{e \in P} c_e$ is minimum among all $[u, v]$-paths.

By induction on the number of labeled nodes we show the following: if a node v is labeled, then $m(v)$ is the cost of a cheapest $[s, v]$-path with one cost-free edge. In order to prove this, we need the property that for all $v \in N$, $m(v)$ is the cost of a cheapest $[s, v]$-path with one cost-free edge whose inner nodes are only labeled nodes. This will be simultaneously shown by the induction.

If s is the only labeled node, the statement is true due to step (3) of Algorithm 5.1. Suppose the statement is true for $i - 1$ labeled nodes and we have chosen an ith node v, say, in step (5). We claim that $m(v)$ is the cost of a cheapest $[s, v]$-path with one cost-free edge. If this is not the case, there exists a path from s to v with one cost-free edge that is cheaper. Suppose P is such a path with cost m_P. Then P must contain an edge that connects an unlabeled node with a labeled one. Let uw (with w unlabeled) be the first of these edges. Obviously, $w \in N$. From the assumption of the induction we know that $m(w)$ is the cost of a cheapest $[s, v]$-path with one

cost-free edge whose inner nodes are only labeled nodes. Thus, $m(w) \leq m_P < m(v)$, a contradiction to the choice of v.

It remains to be shown that for all unlabeled nodes $u \in N$ the value $m(u)$ is the cost of a cheapest $[s,u]$-path with one cost-free edge whose inner nodes are labeled. We assume that v was the node chosen in step (5).

Due to the induction assumption, $m(u)$ is the cost of a cheapest $[s,u]$-path with one cost-free edge whose inner nodes are labeled and different from v. This value is compared in step (7) with the cost of a cheapest $[s,u]$-path with one cost-free edge whose predecessor is v and whose inner nodes are labeled. Suppose there exists an $[s,u]$-path P with one cost-free edge that is cheaper and whose inner nodes are labeled such that $v \in V(P)$ and $wu \in P$, $w \neq v$. Without loss of generality, let P be the cheapest of those paths and m_P the cost of P. If $m_P = d(w)$ (i.e., wu is a maximal edge), we conclude that $m_P = d(w) \geq d(v) \geq m(u)$, a contradiction. Otherwise, $m_P = m(w) + c_{wu}$. Since w was labeled before v, there exists due to the assumption of induction a cheapest path P' from s to w with one cost-free edge whose inner nodes are labeled and different from v. Let $m_{P'}$ be the cost of P'. We obtain that $m_P = m(w) + c_{wu} \geq m_{P'} + c_{wu} \geq m(u)$, a contradiction. This shows Theorem 5.2. □

6. Computational results. In this section we report on the success of our separation algorithms for the solution of practical problem instances. We have developed a branch and cut algorithm to solve a certain class of Steiner tree packing problems arising in the design of electronic circuits. Here, the underlying graph is a complete rectangular grid graph and the set of terminals are located on the outer face. The task is to find a Steiner tree packing with minimal weight, where all edge weights are equal to one. These problems are called *switchbox routing problems* in the VLSI literature. We have tested our algorithm on switchbox routing problems discussed in the literature. We emphasize here the performance of the separation routines and selected four test samples for this purpose.

TABLE 1

Example	h	w	N	Distribution of the nets					Ref.
				2	3	4	5	6	
Difficult switchbox	15	23	24	15	3	4	1	1	[BP83]
Terminal intensive switchbox	16	23	24	8	7	5	4		[L85]
Dense switchbox	17	15	19	3	11	5			[L85]
Pedagogical switchbox	16	15	22	14	4	4			[CH88]

Table 1 summarizes the data of our test problems. Column 1 presents the names of the instances used in the literature. In columns 2 and 3 the height and width of the underlying grid graph is given. Column 4 contains the number of nets. Columns 5 to 9 provide information about the distribution of the nets; more precisely, column 5 gives the number of 2-terminal nets, column 6 gives the number of 3-terminal nets, and so on. Finally, the last column states the reference to the paper the example is taken from.

In [GMW92b] we report on our experiences for solving these problems with a branch and cut algorithm. For more details on these switchbox routing problems and on the general outline of our branch and cut algorithm we refer to that paper.

We focus in this section on our evaluation of the various separation algorithms described in the previous sections. We have, in total, implemented nine exact and heuristic separation routines. We have executed many test runs using just a single separation routine and two, three, or more separation routines in various combinations and orders. It seems impossible to present all the data of these runs here and discuss the relative merits of the choices. We rather want to describe our final selection of separation algorithms and to indicate why we have made some of the choices.

Initially, we started with the trivial LP relaxation consisting of just the upper and lower bounds and the degree constraints for all terminals. This turned out to be a disastrous beginning. It took the separation routines almost forever to add sufficiently many cutting planes so that the graphs G^k induced by the edges $E^k := \{e \in E \mid x_e^k > 0\}$ became connected. We therefore added a preprocessing stage that, for each net, generates certain Steiner partition inequalities by analyzing the positions of the terminals of the net. In particular, our program determines all horizontal and vertical cuts that separate two terminals of a net and a number of further suitably chosen Steiner partition inequalities. In this stage we keep an eye on the spatial distribution of the corresponding cuts and multicuts; i.e., we try to select inequalities in such a way that almost every edge appears with a positive coefficient in one of the initial inequalities and only few edges occur in many inequalities. The reason for this rule is that, by this choice, the LP solver is unable to satisfy many inequalities at once by setting just a few variables to a positive value. Satisfactory rules for determining Steiner cut and partition inequalities of this type were found by running various combinations of choices and comparing the computational results on many practical instances. The introduction of this preprocessing stage was, in retrospect, decisive for the practical success of our approach.

For the separation of the Steiner partition inequalities, we have programmed the exact separation routine described in §3 and two heuristics. These heuristics determine short Steiner trees in the dual graph G_D introduced in §3. The running times of these heuristics are only small fractions of the running time of the exact separation algorithm. Moreover, the heuristics tend to find significantly more violated constraints than the exact routine. Our experiments indicated that a certain combination of the heuristics and the exact method seems to perform best. We first run the two heuristics and stop the cutting plane generation if a certain threshold for the number of cutting planes that we want to generate at most in one iteration is surpassed. We control the heuristics by several parameters so that violated Steiner partition inequalities of different structure and small overlap are generated. The time-consuming exact method is only called if none of the separation heuristics is able to find a violated Steiner partition inequality. Column 2 of Table 2 shows the number of Steiner partition inequalities generated during the runs of our final combination of exact and heuristic separation algorithms for the Steiner partition inequalities on the test instances. The results show that our methods are quite successful cutting plane generators.

TABLE 2

Example	Steiner part. ineq.	Ext. alt. cycle ineq.
Difficult switchbox	5328	924
Terminal intensive switchbox	6050	862
Dense switchbox	3416	436
Pedagogical switchbox	2977	556

Our computational experiments revealed that a similar strategy also yields the best results with respect to separating extended alternating cycle inequalities. Here our final choice was to execute the separation heuristic described in §4 first and to call the dynamic program only if the separation heuristic failed to determine a violated extended alternating cycle inequality. Moreover, based on comparing the running time spent with the probability of success, we decided to call the separation algorithms for the extended alternating cycle inequalities not for all net pairs. Our choice is as follows. We determine "conflicting nets," i.e., those nets that our primal heuristic for finding a Steiner tree packing is unable to route simultaneously, and run the separation routines for extended alternating cycle inequalities only for these pairs of nets. Column 3 of Table 2 shows the number of violated extended alternating cycle inequalities that were generated with these strategies for our test instances. Again, this combination of separation methods was highly successful.

TABLE 3

Example	Steiner cut			Steiner part.			St. part. + al. cycle		
	lb	ub	time	lb	ub	time	lb	ub	time
Difficult switchbox	441	465	1217:11	464	464	1077:05	464	464	983:16
Term. int. switchbox	505	∞	986:12	535	544	1227:54	535	539	1495:32
Dense switchbox	433	∞	2:44	438	∞	580:52	438	∞	347:02
Pedagogical switchbox	318	∞	91:07	331	340	142:35	331	340	158:37

Table 3 emphasizes the performance of our final selection of separation algorithms. The table shows three different strategies. One is to compute the LP-relaxation of the integer program (2.1). Violated Steiner cut inequalities (see (2.1) (i)) can be found in polynomial time by applying min-cut computations. For our test instances, this separation problem reduces to computing shortest paths in the dual graph G_D. (See §3.) This is the method we implemented. Columns 2 through 4 show the lower bound, the upper bound (computed by an LP-based heuristic), and CPU time in minutes (spent on a Sun Sparc SS20-502) until no more violated Steiner cut inequalities can be found. Columns 5 to 7 present the results when Steiner partition inequalities are also separated in the way previously explained. We report on the numbers that we achieve with our final separation strategy (including the separation of Steiner partition inequalities and extended alternating cycle inequalities) in columns 8–10.

Most of the switchbox routing problems that we investigated can be solved to optimality without too much branching. This does not only indicate that the separation algorithms work very well, but also that the Steiner partition inequalities and the alternating cycle inequalities describe the (for our type of problems) relevant part of the Steiner tree packing polyhedron quite well. Moreover, it has turned out that most of the violated inequalities were found by the separation heuristics and that the dynamic programs were called only a few times. Thus, we are hopeful that this approach is also applicable to practical problem instances where only separation heuristics are at hand, i.e., where the underlying graph is not planar or the terminals are not located on a fixed number of faces.

REFERENCES

[AMO93] R. K. AHUJA, T. L. MAGNANTI, AND J. B. ORLIN, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[BP83] M. BURSTEIN AND R. PELAVIN, *Hierarchical wire routing*, IEEE Trans. Computer-Aided-Design, 2 (1983), pp. 223–234.

[CH88] J. P. COHOON AND P. L. HECK, *BEAVER: A computational-geometry-based tool for switchbox routing*, IEEE Trans. Computer-Aided-Design, 7 (1988), pp. 684–697.

[DW71] S. E. DREYFUS AND R. A. WAGNER, *The Steiner problem in graphs*, Networks, 1 (1971), pp. 195–207.

[EMV87] R. E. ERICKSON, C. L. MONMA, AND A. F. VEINOTT, *Send-and-split method for minimum concave-cost network flows*, Math. Oper. Res., 12 (1987), pp. 634–664.

[GM90] M. GRÖTSCHEL AND C. L. MONMA, *Integer polyhedra associated with certain network design problems with connectivity constraints*, SIAM J. Discrete Math., 3 (1990), pp. 502–523.

[GMS92] M. GRÖTSCHEL, C. L. MONMA, AND M. STOER, *Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints*, Oper. Res., 40 (1992), pp. 309–330.

[GMW92a] M. GRÖTSCHEL, A. MARTIN, AND R. WEISMANTEL, *Packing Steiner Trees: Polyhedral Investigations*, preprint SC 92-8, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1992, Mathematical Programming, to appear.

[GMW92b] ———, *Packing Steiner trees: A Cutting Plane Algorithm and Computational Results*, preprint SC 92-9, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1992; Math. Programming, to appear.

[GMW95] ———, *Packing Steiner trees: Further facets*, European J. Combin., 17 (1995), pp. 39–52.

[HRW92] F. K. HWANG, D. S. RICHARDS, AND P. WINTER, *The Steiner Tree Problem*, Annals of Discrete Mathematics 53, North-Holland, Amsterdam, 1992.

[KL84] M. R. KRAMER AND J. VAN LEEUWEN, *The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits*, in Advances in Computing Research, Vol. 2: VLSI Theory, F. P. Preparata, ed., Jai Press, London, 1984, pp. 129–146.

[L85] W. K. LUK, *A greedy switch-box router*, Integration, 3 (1985), pp. 129–149.

[M92] A. MARTIN, *Packen von Steinerbäumen: Polyedrische Studien und Anwendung*, Ph.D. thesis, Technische Universität Berlin, 1992.

[S87] M. SARRAFZADEH, *Channel-routing problem in the knock-knee mode is $\mathcal{NP}$-complete*, IEEE Trans. Computer-Aided-Design, 6 (1987), pp. 503–506.

[TM80] H. TAKAHASHI AND A. MATSUYAMA, *An approximate solution for the Steiner problem in graphs*, Math. Japon., 24 (1980), pp. 573–577.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 258–268, May 1996

008

ON THE POWER OF DEMOCRATIC NETWORKS*

E. N. MAYORAZ †

Abstract. Linear threshold Boolean units (LTUs) are the basic processing components of artificial neural networks of Boolean activations. Quantization of their parameters is a central question in hardware implementation, when numerical technologies are used to store the configuration of the circuit. In the previous studies on the circuit complexity of feedforward neural networks, no differences had been made between a network with "small" integer weights and one composed of majority units (LTUs with weights in $\{-1, 0, +1\}$), since any connection of weight w (w integer) can be simulated by $|w|$ connections of value $\mathrm{sgn}(w)$. This paper will focus on the circuit complexity of democratic networks, i.e., circuits of majority units with at most one connection between each pair of units.

The main results presented are the following: any Boolean function can be computed by a depth-3 nondegenerate democratic network and can be expressed as a linear threshold function of majorities; *AT-LEAST-k* and *AT-MOST-k* are computable by a depth-2, polynomial-sized democratic network; the smallest sizes of depth-2 circuits computing *PARITY* are identical for a democratic network and for a usual network; the *VC*-dimension of the class of the majority functions is $n + 1$, i.e., equal to that of the class of any linear threshold functions.

Key words. artificial neural network, weight quantization, linear threshold function, majority function, circuit complexity

AMS subject classifications. 68Q05, 68R99

1. Introduction.

A Boolean function $f(\boldsymbol{b}) : \mathbb{B}^n \to \mathbb{B}$ is a *linear threshold function* if there exists a weight vector $\boldsymbol{w} \in \mathbb{R}^n$ and a threshold $w_0 \in \mathbb{R}$ such that

$$f(\boldsymbol{b}) = \mathrm{sgn}(w_0 + \boldsymbol{b}^\top \boldsymbol{w}). \tag{1.1}$$

The numerical representation used in this paper for the set of Boolean values $\mathbb{B}$ will be $\{-1, +1\}$, and the sign function $\mathrm{sgn} : \mathbb{R} \to \mathbb{B}$ is defined as $\mathrm{sgn}(x) = +1$ if and only if $x > 0$. The processing unit computing a linear threshold Boolean function (LTU) is the basic component of artificial neural networks. A feedforward (i.e., cycle-free) network $\mathcal{N}$ is characterized by its *depth* $d(\mathcal{N})$, which denotes the length of the longest oriented path in $\mathcal{N}$, and by its *size* $s(\mathcal{N})$, which will be defined, in the present study, as the number of processing units in $\mathcal{N}$. According to the notation used in [2], LT_1 denotes the set of all linear threshold Boolean functions, while $\mathcal{LT}_d$ (respectively, LT_d) represents the set of all Boolean functions that can be computed by a feedforward network composed of LTUs, with a depth d and of any size (respectively, with a size bounded by a polynomial in the number of inputs of $\mathcal{N}$).

Since LT_1 contains the conjunction and the disjunction of arbitrarily many arguments, the set $\mathcal{B}^n$ of every Boolean function of n arguments is clearly included in $\mathcal{LT}_2$. However, when circuits with size bounded polynomially in n are considered, many questions remain. On the one hand, since the number of linear threshold Boolean functions of at most n arguments is in $2^{\Theta(n^2)}$ (see [11, 21]), $\mathcal{B}^n \not\subset LT_d$ for any constant depth d. On the other hand, $LT_1 \subsetneq LT_2$ is the only inclusion known to be proper in the whole hierarchy $LT_1 \subset LT_2 \subset LT_3 \subset \ldots$.

The quantization of the parameters w_i ($i = 0, \ldots, n$) of the LTUs is essential for any hardware implementation using numerical technologies to store the w_i's. A

* Received by the editors February 16, 1993; accepted for publication (in revised form) May 25, 1995. This research was supported by Swiss National Science Foundation grant 20-5637.88.

† Department of Mathematics, Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland. Current address: Dalle Molle Institute of Perceptive Artificial Intelligence (IDIAP), CP 592, CH-1920 Martigny, Switzerland (mayoraz@ idiap.ch).

famous result due to Muroga, Toda, and Takasu [12] (see also [14] for a concise proof) shows that the weights of any linear threshold function of n inputs can be integers bounded from above by

$$\frac{(n+1)^{\frac{n+1}{2}}}{2^n}.$$

It is easy to see that some Boolean functions such as *COMPARISON* are in LT_1 but require weights of exponential size. Thus, the most important restriction of LTUs which has been considered in the literature has *small weights*, i.e., integer weights bounded polynomially in the fan-in [6, 15, 18]. Let $\widehat{LT}_d$ denote the set of Boolean functions computable by a depth-d polynomial-sized circuit composed of LTUs with small weights. The strongest relationship between LT_d and $\widehat{LT}_d$ has been obtained recently by Goldmann and Karpinski [5], who proved that $LT_d \subset \widehat{LT}_{d+1}\ \forall d \geq 1$.

The class of linear threshold Boolean functions with integer parameters w_i bounded by a constant naturally constitutes the next stage in this simplification of the LTUs. The simplest situation, where each w_i is either $+1, 0$, or -1, corresponds to the class of *majority Boolean functions* and is the central topic of this paper. From a circuit complexity point of view, this new subset of linear threshold Boolean functions presents no particular interest since any LTUs can be transformed into a unit computing a majority function by replacing each connection of value w_i by $|w_i|$ connections of value $\mathrm{sgn}(w_i)$. Moreover, the polynomial-sized property of a network is preserved by this transformation when the functions computed by the units of the initial net are in $\widehat{LT}_1$. Therefore, in all the theoretical works on the circuit complexity of feedforward networks, the circuits composed of majority Boolean functions are only mentioned as equivalent to these based on functions in $\widehat{LT}_1$.

However, the aim of an artificial neural network is more to be able to learn a wide family of tasks than to achieve one particular function. Whenever a circuit architecture has to be determined to suit various tasks, it is of high interest to be able to choose, for example, between a circuit with a few bits per connections or another with some more computational units but with at most one connection of one bit between each pair of units. Also, in simulation it would be desirable to know whether a given problem, generally formulated as two sets, one of positive examples and the other of negative examples, can be correctly learned by a network of a specific architecture (number of layers and number of units per layer) and a specific quantization level of the parameters.

More specifically, in some other studies, we are designing training algorithms for democratic networks [7, 8, 9]. Among other approaches, we attempted to develop methods constructing the network layer by layer during the training phase [1]. In this context, it is highly important to know, for example, whether there exists a depth-2 network for any task that has to be loaded.

The present paper investigates the computational power of feedforward networks whose units realize majority functions. It tries to shed some light on the following question:

> Does the computational power of LTUs lie more in the richness of the various affine combinations of the inputs or in the nonlinear function sgn?

The remainder of this paper is divided into five sections. The model of the network involved in the following sections is defined formally in §2. Section 3 presents a couple of simple constructions for circuits computing some basic Boolean functions such as

AND, *OR*, *AT-LEAST-k*, and *PARITY*. The existence of a depth-2 universal circuit, i.e., able to realize any functions of $\mathcal{B}^n$, is discussed in §4. In §5, the *VC*-dimension of the class of majority Boolean functions is shown to be equal to that of all the linear threshold Boolean functions. A general discussion and some suggestions for further research constitute the concluding section.

2. Majority functions and democratic networks. The class MAJ_1 of the *majority Boolean functions* is the set of linear threshold Boolean functions with weights w_i restricted to the set $\{-1, 0, +1\}$.

Remark 2.1. For the sake of simplicity in the further developments, it is convenient to consider a class of functions closed under negation. The negation of a function $f \in MAJ_1$ of weights $\boldsymbol{w}$ is already in MAJ_1 when the number of arguments of f is odd, since the latter is self-dual. The negation of an "even-majority" function can be obtained by an inversion of the weight vector, and by simultaneously changing the convention on the value of sgn(0). For example, this effect can be obtained by allowing the threshold to take values in $\{-\frac{1}{2}, +\frac{1}{2}\}$. This limited flexibility of the threshold gives the choice between an "absolute" and a "nonabsolute" majority and has no effect on odd-majority functions. Finally, observe that with the closure of MAJ_1 under negation we also get its closure under duality.

In practice, an inversion of every out-going connection of a unit computing f corresponds to a modification of f into not-f; if all the in-going connections are also inverted, f is changed into its dual f^d and, finally, if the threshold is inverted at the same time, the new function computed by the unit is f again.

DEFINITION 2.1. *A Boolean function* $f : \mathbb{B}^n \to \mathbb{B}$ *is a* majority Boolean function (i.e., $f \in MAJ_1$) *if there exists a weight vector* $\boldsymbol{w} \in \{-1, 0, +1\}^n$ *and a threshold* $w_0 = \pm\frac{1}{2}$ *satisfying equation* (1.1). *The* w_i *are also called the* coefficients *of* f *and the dot product* $\boldsymbol{w}^\top \boldsymbol{b}$ *is the* potential *of* f *for the input* $\boldsymbol{b}$.

DEFINITION 2.2. *A* democratic network *is a feedforward circuit composed of units computing majority Boolean functions with the additional property that there is at most one connection between each pair of units.* $\mathcal{MAJ}_d$ *(respectively,* MAJ_d*) denotes the set of Boolean functions realizable by a democratic network of depth* d *and of any size (respectively, of size bounded by a polynomial in its number of inputs). A network is said to be* degenerate *if it does contain at least two distinct subcircuits computing the same Boolean function.*

One observes that the closure of MAJ_1 under negation and duality implies this same property on $\mathcal{MAJ}_d$ and on MAJ_d.

3. Representation of basic functions. In the beginning of the last decade, polynomial-sized constant depth circuits composed of LTUs became popular when it was first shown that there is no polynomial-sized constant depth circuit computing *PARITY* with only *AND*, *OR*, and *NOT* processing units [4]. A few years later it was proved that even if we add the *PARITY* function to this previous set of basic units, there is no polynomial-sized constant depth circuit able to compute *MAJORITY* [16]. In contrast, it is interesting to determine the complexity of democratic networks computing these basic functions.

Since the binary conjunction 2-*AND* is a majority function, the conjunction of n arguments n-*AND* is in $MAJ_{\lceil \log_2 n \rceil}$ by decomposition into 2-*AND*s. However, the conjunction of n arguments can be realized by democratic networks of smaller depth.

PROPOSITION 3.1. n-$AND \in MAJ_2$.

Proof. Consider the $n-1$ pairs of inputs $(1,2), (2,3), \ldots, (n-1, n)$. For each of these pairs, introduce two units on the hidden layer, one with coefficients $(+1, +1)$

and the other with coefficients $(-1,-1)$. Each hidden unit has a negative threshold and is connected to the output unit by a link of weight $+1$. The contribution of each pair of hidden units to the output potential is 0 if the two corresponding inputs are identical and -2 otherwise. The total output potential will then be 0 if all the inputs are identical and negative otherwise. Adding one connection from an arbitrary input to the output will produce the desired function. Moreover, the network is clearly nondegenerate, and $d = 2$, $s = 2n - 2$. □

Note that the latter construction uses only fan-in-2 units on the hidden layer. When the depth of the circuit is not critical, *n-AND* can be computed with a number of units bounded by a logarithm in n.

PROPOSITION 3.2. *n-AND can be computed by a nondegenerate democratic network, with* $s \in O(\log n)$ *and* $d \in O(\log^* n)$.

To verify this proposition let us first prove the following lemma.

LEMMA 3.3. *The computation of n-AND can be reduced to the computation of* $\lfloor \log_2(n+1) \rfloor$*-AND, using* $O(\log_2 n)$ *majority units.*

Proof. Consider a partition of the input set I into I_1 and I_2 with $|I_2| = \lceil \frac{n}{2} \rceil - 1$. Add a hidden unit with a negative threshold, connected to each input of I_1 with a weight $+1$ and to each input of I_2 with a weight -1. If all the inputs of I_2 are set to $+1$, the hidden neuron gives an output $+1$ only if all the inputs of I_1 are also set to $+1$. Thus, the computation of *AND* over the n inputs is equivalent to the computation of *AND* over the hidden neuron and the inputs of I_2. Applying this idea recursively to the subset I_2, one can construct one hidden layer composed of $h(n)$ neurons, where $h(n)$ is given by the following recursive equation:

$$h(n) = 1 + h\left(\left\lceil \frac{n}{2} \right\rceil - 1\right), \qquad h(0) = h(1) = 0. \tag{3.1}$$

The exact solution of equation (3.1) is $h(n) = \lfloor \log_2 \frac{2}{3}(n+1) \rfloor$ for every $n > 0$. The number $a(n)$ of arguments of the remaining *AND* is given by the same recursive relation as $h(n)$ but with initial conditions $a(0) = 0$ and $a(1) = 1$. By solving this equation, one obtains $a(n) = \lfloor \log_2(n+1) \rfloor$. □

Proposition 3.2 is established by recursively applying Lemma 3.3.

Proof. The final size $s(n)$ of the network is given by the following recursive equation:

$$s(n) = h(n) + s(a(n)), \qquad s(0) = s(1) = 0. \tag{3.2}$$

It can easily be proved that the solution $s(n)$ of this equation is in $O(\log n)$. The depth of this network is given by the amount of time one has to apply function a to the number n of inputs until a value smaller than 1 is reached, and this is clearly in $O(\log^* n)$. □

Figure 3.1 illustrates this construction by showing the transformation of 10-*AND* into a three-layered democratic network.

At first glance, the fact that in our model of majority function the threshold is limited to $\{\pm\frac{1}{2}\}$ seems to be very restrictive. Indeed, with a threshold varying in the set $\{-n, \ldots, +n\}$, the set of basic functions would contain *n-AND* and *n-OR* and more generally for every k, *AT-LEAST-k* (respectively, *AT-MOST-k*) which takes the value $+1$ if and only if there are at least (respectively, at most) k positive inputs. This restriction on the threshold was maintained for uniformity with the coefficients and to satisfy the constraints given by the hardware implementation. The next proposition shows that for every k, *AT-LEAST-k* and *AT-MOST-k* are computable by small democratic networks.

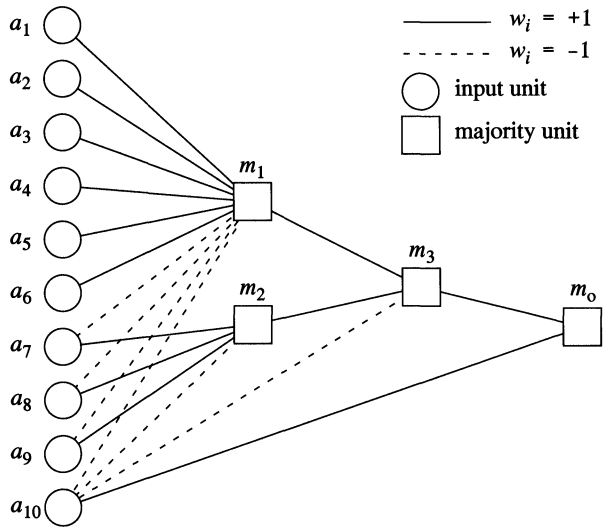


FIG. 3.1. *Multilayer democratic circuit simulating a conjunction* $AND(a_1, \ldots, a_{10}) = AND(m_1, a_7, \ldots, a_{10}) = AND(m_1, m_2, a_{10}) = AND(m_3, a_{10}) = m_o$.

PROPOSITION 3.4. *AT-LEAST-k, AT-MOST-k* $\in MAJ_2$ $\forall k$.

Proof. Considering that MAJ_2 is closed under negation and duality and that *AT-LEAST-*$(\lfloor \frac{n}{2} \rfloor + 1)$ is in MAJ_1, we only have to show that *AT-LEAST-k* is in MAJ_2 $\forall$ $k = \lfloor \frac{n}{2} \rfloor + 2, \ldots, n$.

Regroup the inputs into $m = \lfloor \frac{n}{2} \rfloor$ pairs $(1,2), (3,4), \ldots$. Connect each input directly to the output unit with a positive weight. For each of the m pairs of inputs, add four hidden units of negative threshold and of coefficients $(+1,+1)$, $(+1,-1)$, $(-1,+1)$, and $(-1,-1)$, respectively. When each of the $6m$ connections toward the output unit has the value $+1$, the contribution of each pair of inputs to the output potential is -2 times the number of negative inputs in the pair. The total contribution is then $-2l$, where l is the number of negative inputs among the $2m$ first inputs.

To realize *AT-LEAST-k*, we only need to add on the hidden layer $1+2(n-k)$ units which should have a positive answer whenever there are at least k positive inputs. These $1 + 2(n - k)$ units can be any of the $n + 1$ units with at least $n - 1$ among n coefficients equal to $+1$. As $n + 1$ is bigger than $1 + 2(n - k)$ when $k \geq \lfloor \frac{n}{2} \rfloor + 2$, the network can always be nondegenerate. Thus, the output potential v will always be odd and

$$v \begin{cases} \geq & 1 \quad \text{if} \quad l \leq n - k, \\ \leq & -1 \quad \text{if} \quad l > n - k. \end{cases}$$

In case of an odd number of arguments n, we just need to add a positive connection from the nth input to the output and choose a positive threshold for the output unit. □

DEFINITION 3.1. *A Boolean function f is* symmetric *if and only if* $f(b_1, \ldots, b_n) = f(b_{\sigma(1)}, \ldots, b_{\sigma(n)})$ *for any permutation* σ *of the inputs. A well-known characterization of symmetric functions is the following: f is symmetric if there exist k integers* $t_1, \ldots, t_k$ *such that* $f(b_1, \ldots, b_n) = 1$ *if and only if* $\sum_{i=1}^{n} b_i \in \{t_1, \ldots, t_k\}$ [6, 17].

COROLLARY 3.5. *Any symmetric Boolean function is in* MAJ_3.

Proof. This result is a direct consequence of Proposition 3.4 and of the construction presented in [6, 17].

$$f(\boldsymbol{b}) = \text{maj\_}(AT\text{-}LEAST\text{-}t_1(\boldsymbol{b}),\ AT\text{-}MOST\text{-}t_1(\boldsymbol{b}),\ \ldots,\ AT\text{-}LEAST\text{-}t_k(\boldsymbol{b}),\ AT\text{-}MOST\text{-}t_k(\boldsymbol{b})), \tag{3.3}$$

where maj_ denotes the majority function with all coefficients $+1$ and a negative threshold. □

The *n-PARITY* function is defined as the product of its inputs: $f(b_1, \ldots, b_n) = \prod_{i=1}^{n} b_i$. A well-known depth-2 circuit of LTUs realizes *n-PARITY* with exactly n hidden units: each of them computes *AT-LEAST-k* for $k = 1, \ldots, n$, and the output is a function of MAJ_1 with alternating weight signs $\boldsymbol{w} = (+1, -1, +1, \ldots)$. This construction is the smallest known when no jumping connections over layers are allowed; otherwise, the size s of the depth-2 circuit can be divided by 2 [10] and if depth-3 networks are considered, the size can even be reduced to $O(\sqrt{n})$ [19].

Although the above construction for *PARITY* reduces by a factor 2 the size s of the depth-3 democratic network compared with the general construction for a symmetric function (Corollary 3.5), it cannot be used for the development of a depth-2 circuit computing *n-PARITY*. The following proposition presents a completely different construction solving this problem with no more than n units on the single hidden layer, and without jumping connections. Note that this construction has been discovered independently by Grossman and is mentioned without proof in [13].

PROPOSITION 3.6. *n-PARITY can be computed by a depth-2 nondegenerate democratic network composed of n hidden units.*

Proof. On the hypercube $\mathbb{B}^n$, let us call the point $-\boldsymbol{b}$ the *antipodal* of $\boldsymbol{b}$; *equator* of $\boldsymbol{b}$, noted $\text{Eq}(\boldsymbol{b})$, denotes the set $\{\boldsymbol{e} \in \mathbb{B}^n | \boldsymbol{e}^\top \boldsymbol{b} = 0\}$. For any Boolean function f, the *characteristic subset* of $\mathbb{B}^n$ is defined as $\text{C}(f) = \{\boldsymbol{b} \in \mathbb{B}^n | f(\boldsymbol{b}) = +1\}$. The two following observations are the key elements of the proof:

- The sum of the outputs of two majority units of positive threshold and with weights $\boldsymbol{w}$ and $-\boldsymbol{w}$, respectively, is $+2$ if the input is in $\text{Eq}(\boldsymbol{w})$ and 0 otherwise (obvious).
- For n even, there exist $\frac{n}{2}$ equators covering $\text{C}(n\text{-}PARITY)$ without containing any other points (proved below).

With these remarks, the construction follows easily and it is illustrated in Figure 3.2 for the case $n = 4$.

The network with n hidden units and one output unit is such that each hidden unit i has a weight vector $\boldsymbol{w}^i$, a positive threshold, and a positive connection with the output. The $\boldsymbol{w}^i$'s are defined by

$$\boldsymbol{w}^1 = (\underbrace{+1, \ldots, +1}_{\lceil \frac{n}{2} \rceil}, \underbrace{-1, \ldots, -1}_{\lfloor \frac{n}{2} \rfloor}),$$

and $\boldsymbol{w}^2, \ldots, \boldsymbol{w}^n$ are the cyclic permutations of the n components of $\boldsymbol{w}^1$; i.e., $w^{i+1}_{j \oplus 1} = w^i_j$, $i, j \in \{1, \ldots, n\}$, where $j \oplus 1$ denotes the cyclic increment of j ($n \oplus 1 = 1$).

As a consequence of this definition of the $\boldsymbol{w}^i$ we note that $\forall \boldsymbol{b} \in \mathbb{B}^n, \forall i \in \{1, \ldots, n\}$,

$$\boldsymbol{b}^\top \boldsymbol{w}^{i \oplus 1} - \boldsymbol{b}^\top \boldsymbol{w}^i \in \{-4, 0, +4\}. \tag{3.4}$$

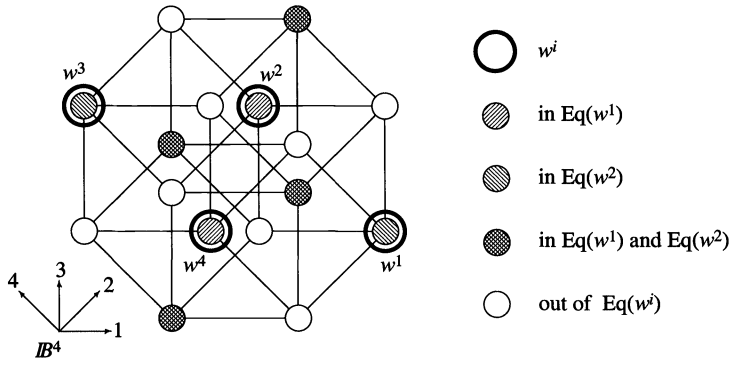


FIG. 3.2. *Construction of four-PARITY using four majority units. Four majority units, whose weight vectors w^i are equal to the vertices of the hypercube indicated with a thick circle, are used to cover all the vertices of positive parity without covering any other vertex.*

Moreover, if b^+ (respectively, b^-) denotes the number of positive (respectively, negative) components in the point $\boldsymbol{b}$,

$$\boldsymbol{b}^\top \boldsymbol{w}^i \bmod 4 = \begin{cases} 0 & \text{if } b^+, b^- \text{ are even} & \text{(i.e., } n \text{ is even)}, \\ 1 & \text{if } b^+ \text{ is odd and } b^- \text{ is even} & \text{(i.e., } n \text{ is odd)}, \\ 2 & \text{if } b^+, b^- \text{ are odd} & \text{(i.e., } n \text{ is even)}, \\ 3 & \text{if } b^+ \text{ is even and } b^- \text{ is odd} & \text{(i.e., } n \text{ is odd)}. \end{cases} \tag{3.5}$$

- When n is even, $\boldsymbol{w}^i$ and $\boldsymbol{w}^{i\pm\frac{n}{2}}$ are antipodal and thus the contribution of the units i and $i \pm \frac{n}{2}$ to the output potential is 0 for every point of $\mathbb{B}^n$ except those in Eq($\boldsymbol{w}^i$) for which it is $+2$. By equation (3.5), if a point $\boldsymbol{b}$ is in Eq($\boldsymbol{w}^i$), b^+ and b^- are both even and thus $\boldsymbol{b} \in$ C(n-*PARITY*). To complete the proof it remains to show that any point of C(n-*PARITY*) is in Eq($\boldsymbol{w}^i$) for at least one i. Let $\boldsymbol{b}$ be an arbitrary point in C(n-*PARITY*); i.e., b^- is even and by equation (3.5), $\boldsymbol{b}^\top \boldsymbol{w}^i \bmod 4 = 0$, and say $\boldsymbol{b}^\top \boldsymbol{w}^1 = 4\alpha = -\boldsymbol{b}^\top \boldsymbol{w}^{\frac{n}{2}+1}$. By property (3.4), the sequence $4\alpha = \boldsymbol{b}^\top \boldsymbol{w}^1, \ldots, \boldsymbol{b}^\top \boldsymbol{w}^{\frac{n}{2}+1} = -4\alpha$ has to be 0 for at least one $i \in \{1, \ldots, \frac{n}{2}\}$, and thus $\boldsymbol{b} \in$ Eq($\boldsymbol{w}^i$).
- When n is odd (say $n = 2m + 1$), one observes that $\boldsymbol{b}^\top \boldsymbol{w}^i = -\boldsymbol{b}^\top \boldsymbol{w}^{i\pm m} \pm 2$ $\forall i \in \{1, \ldots, n\}$, $\forall \boldsymbol{b} \in \mathbb{B}^n$. This implies that these two dot products are of the same sign only if they are both $+1$ or both -1. As n is odd, there is at least one couple $(\boldsymbol{b}^\top \boldsymbol{w}^i, \boldsymbol{b}^\top \boldsymbol{w}^{i\pm m})$ with both elements of the same sign and by property (3.4), the existence of couples of type $(+1, +1)$ and of type $(-1, -1)$ are mutually exclusive. The sign of the output is thus completely determined by the type of these particular couples appearing in the sequence of the potentials of all the hidden units. Property (3.5) concludes the proof, since couples of type $(+1, +1)$ correspond to the case of b^- even, i.e., $\boldsymbol{b} \in$ C(n-*PARITY*). □

4. Universal democratic networks. Although *AND*s and *OR*s are not very adequate to simulate most of the Boolean functions within a compact-sized circuit, they remain interesting since every function can be represented in a depth-2 circuit, using the CNF or DNF forms (conjunctive or disjunctive normal forms). This section will address the question of what is the shortest democratic network able to compute any Boolean function.

Using the CNF or the DNF form, an obvious corollary of Proposition 3.1 is that $\mathcal{B}^n \in \mathcal{MAJ}_4$. The following proposition shows how $\mathcal{B}^n \in \mathcal{MAJ}_3$ by the simultaneous use of both conjunctive and disjunctive forms.

PROPOSITION 4.1. $\mathcal{MAJ}_3 = \mathcal{B}^n$.

Proof. For a given function f, consider a CNF decomposition $AND(d_1, \ldots, d_k)$ and a DNF decomposition $OR(c_1, \ldots, c_l)$, where d_i and c_i are disjunctions and conjunctions, respectively, over the set of inputs and their negations. Let us assume that $k \geq l$; if it is not the case, one can replace f by not-f and exchange the roles of the d_i and the c_i. Then we claim that

$$f = \text{maj}(c_1, \ldots, c_l, d_1, \ldots, d_{l-1}).$$

This majority is composed of $2l-1$ terms. If $f(\boldsymbol{b}) = +1$, then $d_i(\boldsymbol{b}) = +1 \; \forall i = 1, \ldots, k$ and $c_i(\boldsymbol{b}) = +1$ for at least one $i = 1, \ldots, l$, so at least l terms among the $2l-1$ will be $+1$. On the other hand, if $f(\boldsymbol{b}) = -1$, $c_i(\boldsymbol{b}) = -1 \; \forall i = 1, \ldots, l$ and thus at least l of the $2l-1$ hidden units answer -1 for $\boldsymbol{b}$. □

The size of the network obtained by this construction is in $O(\min\{k, l\})$, but, of course, for almost all interesting Boolean functions, this size of the most compact normal form can be quite large, i.e., exponential in n (e.g., *PARITY*). Moreover, there is very little hope to be able to save one more layer of a three-layered democratic network simulating an arbitrary Boolean function, when the construction is based on CNFs or DNFs of the functions, as suggested in Proposition 4.1.

An alternative way for constructing Boolean functions is based on the well-known fact that every Boolean function can be expressed as a polynomial over the field of rational numbers, in the variables $b_1, \ldots, b_n$, when the numerical representations -1 and $+1$ are used for the Boolean values true and false, respectively. For every Boolean function $f : \mathbb{B}^n \to \mathbb{B}$, there is a unique vector of coefficients $\boldsymbol{c} \in \mathbb{R}^{2^n}$, such that

$$f(\boldsymbol{b}) = \sum_{\alpha \in \{0,1\}^n} c_\alpha \prod_{i=1}^{n} b_i^{\alpha_i}. \tag{4.1}$$

A term of this polynomial, indexed by α, is simply a parity function over the subset of variables whose characteristic vector is α. Thus, using expression (4.1), an arbitrary Boolean function can be expressed as a linear threshold function of parities defined over some of the n inputs. Since the coefficients c_α of the polynomial are rational, the linear threshold function can be simulated by a majority function, assuming a duplication of the parity functions. Using Proposition 3.6, this polynomial form provides a depth-3 degenerate circuit of majority units.

In the last part of this section, we are going to show how any Boolean function can be simulated by a democratic network of depth 2. This issue is of high interest when we consider incremental training algorithms that build a depth-2 democratic network by adding hidden units iteratively.

In [2], the polynomial representation (4.1) has been used to simulate any Boolean function by a network with LTUs, and the author shows how one can get rid of the second layer in order to get a depth-2 network of LTUs computing an arbitrary Boolean function. (See Theorem 2.1 in [2].) However, the construction proposed uses *AT-LEAST-k* and *AT-MOST-k* functions in the first layer, and so it cannot be exploited for the construction of a depth-2 universal democratic network, unless n constant inputs are artificially added to the n original inputs of the network.

PROPOSITION 4.2. *Any Boolean function can be computed by a depth-2 democratic network.*

Proof. Since nondegeneracy is not required in this result, it is sufficient to show that any Boolean function can be computed by a depth-2 circuit, with majority units on the hidden layer, and a single LTU as output unit. The latter can then be simulated by a majority unit and an appropriate duplication of the hidden units. For this purpose, we are going to start from the depth-3 democratic network mentioned above and based on the polynomial representation of equation (4.1), and by using Remark 4.1, we will show how one can get rid of the second layer computing the parity functions.

Remark 4.1. An intermediate unit can be suppressed from a network composed of LTUs if the absolute value of its potential is a nonzero constant α for every possible input of the network. After dropping such a unit of coefficients $\boldsymbol{w}$ and of output connection value o, for each of its in-connections of value w_i, a connection of value $\frac{ow_i}{\alpha}$ should be introduced from the ith predecessor of the unit to its successor.

To complete the proof, we will show that there is a depth-2 democratic network computing the *n-PARITY* in such a way that the absolute value of the output potential is α, where α is a function of n. This network is obtained by placing on the hidden layer the 2^n units with positive threshold and coefficients $\boldsymbol{w} \in \mathbb{B}^{2^n}$. The output connection o of a given hidden unit is fixed by the following rule:

$$\text{If } n \text{ is even,} \quad o = \begin{cases} +1 & \text{if} \quad \mathrm{Eq}(\boldsymbol{w}) \subset \mathrm{C}(n\text{-}PARITY), \\ -1 & \text{if} \quad \mathrm{Eq}(\boldsymbol{w}) \subset \mathbb{B}^n - \mathrm{C}(n\text{-}PARITY); \end{cases}$$

$$\text{if } n \text{ is odd,} \quad o = \begin{cases} +1 & \text{if} \quad \exists\, \boldsymbol{x} \in \mathrm{C}(n\text{-}PARITY) \quad \text{s.t.} \quad \boldsymbol{x}^\top \boldsymbol{w} = +1, \\ -1 & \text{if} \quad \exists\, \boldsymbol{x} \in \mathrm{C}(n\text{-}PARITY) \quad \text{s.t.} \quad \boldsymbol{x}^\top \boldsymbol{w} = -1. \end{cases}$$

This particular choice for the value o ensures that the total contribution to the potential of the output unit is strictly positive if the input is in C(*n-PARITY*) and strictly negative otherwise. Moreover, the property assumed in Remark 4.1 is a consequence of the symmetry due to consideration on the hidden layer of all the complete majorities (i.e., without 0 weights) over the n inputs. □

5. The *VC*-dimension of MAJ_1. A characterization of the computational power of a class of functions is given by the Vapnik–Chervonenkis dimension [20]. In order to formalize this notion, we first introduce some preliminary definitions. A *dichotomy* of p points $\omega_1, \ldots, \omega_p$ in some space Ω is a partition of these points into two disjoint classes. It can be considered as a function $d : \{\omega_1, \ldots, \omega_n\} \to \mathbb{B}$ and it will be denoted by a Boolean vector $\boldsymbol{d} \in \mathbb{B}^p$. Let F be a set of Boolean-valued functions defined on Ω; a subset $\Gamma \subseteq \Omega$ is said to be *shattered* by F if each of the $2^{|\Gamma|}$ possible dichotomies of Γ corresponds to at least one function of F restricted to Γ. The *Vapnik–Chervonenkis dimension* of F, noted $VC\text{-dim}(F)$, is the size of the largest shattered subset $\Gamma \in \Omega$.

When $\Omega = \mathbb{R}^n$, all dichotomies of $n+1$ points are linearly separable if and only if the $n+1$ points are not contained in an $n-1$-dimensional hyperplane of $\mathbb{R}^n$ [3]. Since it is easy to find $n+1$ points in $\mathbb{B}^n$ not contained in an $n-1$-dimensional subspace of $\mathbb{R}^n$, the *VC*-dimension of the set of all linearly separable Boolean functions of n arguments is $n+1$. The following proposition shows that the *VC*-dimension does not change when the set of Boolean functions is restricted from LT_1 to MAJ_1.

PROPOSITION 5.1. $VC\text{-dim}(MAJ_1 \cap \mathcal{B}^n) = n+1$.

Proof. Since $MAJ_1 \subset LT_1$, the *VC*-dimension of $MAJ_1 \cap \mathcal{B}^n$ is bounded from above by $n+1$. To prove the proposition, we will show that the following set of $n+1$

points of $\mathbb{B}^n$ is shattered by MAJ_1:

$$\underbrace{(-1,-1,\ldots,-1)}_{\boldsymbol{b}^0},\ \underbrace{(+1,-1,\ldots,-1)}_{\boldsymbol{b}^1},\ \underbrace{(+1,+1,\ldots,-1)}_{\boldsymbol{b}^2},\ \ldots,\ \underbrace{(+1,+1,\ldots,+1)}_{\boldsymbol{b}^n}.$$

Let an arbitrary dichotomy of these $n+1$ points be given by $\boldsymbol{d} = (d_0, d_1, \ldots, d_n) \in \mathbb{B}^{n+1}$. Since MAJ_1 is closed under negation, we can assume without loss of generality that $d_0 = +1$.

Consider the following definition of the threshold and of the weights:

$$w_0 = \frac{1}{2}, \quad w_i = \frac{d_i - d_{i-1}}{2} \qquad \forall i = 1, \ldots, n. \tag{5.1}$$

With these choices, equation (1.1) gives $f(\boldsymbol{b}^0) = +1 = d_0$, since $w_0 + \boldsymbol{w}^\top \boldsymbol{b}^0 = w_0 + \frac{1-d_n}{2} = 1 - \frac{d_n}{2}$, which is either $\frac{1}{2}$ or $\frac{3}{2}$. Let us call α the quantity $1 - \frac{d_n}{2}$. Finally, observe that for all $i = 1, \ldots, n$, $w_0 + \boldsymbol{w}^\top \boldsymbol{b}^i = w_0 + \boldsymbol{w}^\top \boldsymbol{b}^0 + \boldsymbol{w}^\top (\boldsymbol{b}^i - \boldsymbol{b}^0) = \alpha + \frac{1}{2}\sum_{j=1}^{n}(d_j - d_{j-1})(b_j^i - b_j^0) = \alpha + \sum_{j=1}^{i}(d_j - d_{j-1}) = \alpha + d_i - 1$, which is α (i.e., positive) if $d_i = 1$, and $\alpha - 2$ (i.e., negative) if $d_i = -1$. Thus, the choice of the parameters proposed in (5.1) solves the dichotomy $\boldsymbol{d}$. □

6. Discussion. Throughout this paper, we established various results suggesting that many Boolean functions can be represented efficiently by multilayered networks composed of majority units, even if multiple connections between two units are not allowed. This suggests that a usual network of fixed architecture has no intrinsic limitations when its parameters are limited to +1, 0, and −1.

The constructions developed in the first part of §3 for the computation of *AND* and *AT-MOST-k* are quite simple and their results are not surprising. On the other hand, the number n of hidden units used for the computation of *n-PARITY* with a depth-2 nondegenerate democratic network without jumping connections is probably a tight bound of the minimum, since this value is also the best-known one when general LTUs compose the network.

Many open questions are related to universal democratic networks. Is any Boolean function f computable by a depth-2 nondegenerate democratic network? This question has been solved positively by computer for $n \leq 4$ but is still open for larger numbers of arguments. The polynomial (4.1) is unique and gives the exact value +1 or −1 of the function for any input, and so the sgn function is not necessary. Another open question we attempted to solve without success is the following:

> Can the coefficients c_α of this polynomial be restricted to $-1, 0, +1$ when only the sign of the polynomial is required to match with the output of function f?

This issue goes beyond the neural network field, since it will provide a general way of expressing any Boolean function into a majority of distinct parities. We also checked this question by computer, and it was found to be true for all functions with up to five arguments, but the general question remains open.

Acknowledgment. This work was initiated while the author was working at the Swiss Federal Institute of Technology. It was terminated during a postdoctoral visit at RUTCOR—Rutgers University's Center for Operations Research—and at DIMACS—Center for Discrete Mathematics and Theoretical Computer Science. Thanks are expressed to Frederic Aviolat for his valuable participation in the result of §5 and for his research on the question quoted in the discussion section. I am also grateful to

an anonymous referee for his pertinent remarks and especially for his suggestion of a simpler form of the proof of Proposition 5.1.

REFERENCES

[1] F. AVIOLAT AND E. MAYORAZ, *A constructive training algorithm for feedforward neural networks with ternary weights*, in Proc. European Symposium on Artificial Neural Networks, F. Blayo and M. Verleysen, eds., Brussels, 1994, pp. 123–128.

[2] J. BRUCK, *Harmonic analysis of polynomial threshold functions*, SIAM J. Discrete Math., 3 (1990), pp. 168–177.

[3] T. M. COVER, *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*, IEEE Trans. Electronic Computers, 14 (1965), pp. 326–334.

[4] M. FURST, J. B. SAXE, AND M. SIPSER, *Circuits and the polynomial-time hierarchy*, in Proc. 22nd IEEE Symposium on Foundations of Computer Science, 1981, pp. 260–270.

[5] M. GOLDMANN AND M. KARPINSKI, *Simulating threshold circuits by majority circuits*, in Proc. 25th Annual ACM Symposium on Theory of Computing, San Diego, CA, 1993, pp. 551–560.

[6] A. HAJNAL, W. MAASS, P. PUDLÁK, M. SZEGEDY, AND G. TURÁN, *Threshold circuits of bounded depth*, in Proc. 28th IEEE Symposium on Foundations of Computer Science, Los Angeles, CA, 1987, pp. 99–110.

[7] E. MAYORAZ, *Maximizing the stability of a majority perceptron using tabu search*, in Proc. International Joint Conference on Neural Networks, Baltimore, MD, 1992, pp. II254–II259.

[8] ———, *Feedforward Boolean Networks with Discrete Weights: Computational Power and Training*, Ph.D. thesis, Department of Mathematics, Swiss Federal Institute of Technology, Switzerland, 1993.

[9] E. MAYORAZ AND V. ROBERT, *Maximizing the robustness of a linear threshold classifier with discrete weights*, Network, 5 (1994), pp. 299–315.

[10] R. C. MINNICK, *Linear-input logic*, IEEE Trans. Electronic Computers, 10 (1961).

[11] S. MUROGA, *Threshold Logic and Its Applications*, John Wiley & Sons, New York, 1971.

[12] S. MUROGA, I. TODA, AND S. TAKASU, *Theory of majority decision elements*, J. Franklin Inst., 271 (1961), pp. 376–418.

[13] D. NABUTOVSKY, T. GROSSMAN, AND E. DOMANY, *Learning by CHIR without storing internal representations*, Complex Systems, 4 (1990), pp. 519–541.

[14] I. PARBERRY, *Circuit Complexity and Neural Networks*, Tech. report CRPDC-91-9, Department of Computer Sciences, University of North Texas, Denton, TX, 1991.

[15] I. PARBERRY AND G. SCHNITGER, *Parallel computation with threshold functions*, J. Comput. System Sci., 36 (1988), pp. 278–302.

[16] A. A. RAZBOROV, *Lower bounds on the size of bounded depth circuits over a complete basis with logical addition*, Math. Notes, 41 (1987), pp. 333–338.

[17] K.-Y. SIU AND J. BRUCK, *Neural computation of arithmetic functions*, Proc. IEEE, 78 (1990), pp. 1669–1675.

[18] ———, *On the power of threshold circuits with small weights*, SIAM J. Discrete Math., 4 (1991), pp. 423–435.

[19] K.-Y. SIU, V. P. ROYCHOWDHURY, AND T. KAILATH, *Depth-size tradeoffs for neural computation*, IEEE Trans. Comput., Special Issue on Neural Networks (1991), pp. 1402–1412.

[20] V. N. VAPNIK, *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, New York, 1982.

[21] Y. A. ZUEV, *Asymptotics of the logarithm of the number of threshold functions of the algebra of logic*, Soviet Math. Dokl., 39 (1989), pp. 512–513.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 269–273, May 1996

009

BOUNDING FUNCTIONS AND RIGID GRAPHS*

MICHAEL O. ALBERTSON† AND RUTH HAAS†

Abstract. A function f *bounds graphs from above* if there exists an infinite family of graphs $\mathcal{G}$, such that if $G \in \mathcal{G}$ then $f(|V_G|) = |E_G|$ and for all nonempty subgraphs H of G we have that $f(|V_H|) \geq |E_H|$. This paper considers the question: Which functions bound graphs?

Key words. graphs, extremal graph theory

AMS subject classification. 05C99

1. Introduction.

Let V_G (E_G) denote the vertex (edge) set of a simple graph G. A function f is said to *bound graphs from above* if there exists an infinite family of graphs $\mathcal{G}$, such that if $G \in \mathcal{G}$, then $f(|V_G|) = |E_G|$ and for all nonempty subgraphs H of G we have that $f(|V_H|) \geq |E_H|$. If we insist on inequality for all proper subgraphs we say that f *strictly* bounds.

In this context we say that a family of graphs is *bounded* (strictly bounded). We will also speak of a single graph being bounded when its membership in a particular family is implicit. For example, the function $f(n) = n$ strictly bounds the cycles C_n, since any proper subgraph of C_n with k vertices contains fewer than k edges. The function $f(n) = \binom{n}{2}$ bounds (but not strictly) since the graphs in $\mathcal{G}$ must be complete.

Some graph properties are characterized by bounding functions. For instance, a graph is a Hamilton cycle if and only if it is strictly bounded by $f(n) = n$. Not surprisingly, a tree is characterized by $f(n) = n - 1$. More generally, we have the following result.

THEOREM 1.1. *A graph is the edge disjoint union of k spanning trees if and only if it is bounded by the function $f(n) = k(n-1)$.*

Proof. Suppose G can be decomposed into k disjoint spanning trees. Since any subgraph, say H, of G will contain a forest from each of these k trees, $|E_H| \leq k(|V_H| - 1)$.

On the other hand, suppose that the G is not decomposable into k disjoint spanning trees. If $|E_G| \neq k|V_G| - k$ then clearly G is not bounded by $k(n-1)$. So we must only consider the case where $|E_G| = k|V_G| - k$. We invoke a result of Tutte.]

THEOREM 1.2 (see [3]). *A graph is the union of k disjoint spanning trees if and only if for every partition of the vertices of G into m nonempty parts, there are at least $k(m-1)$ edges between parts.*

Hence, there exists a partition of the vertices of G into m parts such that the number of edges between parts is strictly less than $km - k$. Thus the total number of edges from within the parts is at least $k(m - |V_G|) + 1$. By the pigeon hole principle at least one part, say H, must have more than $k|V_H| - k$ edges. □

Originally, we were intrigued by the role of bounding functions in rigidity theory. Suppose we construct the graph G in the plane using inflexible bars for the edges and rotatable joints for the vertices. This immersion is rigid if the only motions of this structure are trivial (rotations and translations). The graph G is said to be *rigid* in the plane if some immersion of it is rigid. A graph is *minimally rigid* if the removal of any edge makes it nonrigid. See [1], [2], and [4] for further information on rigidity.

* Received by the editors August 17, 1992; accepted for publication (in revised form) June 30, 1995.

† Math Department, Smith College, Northampton, MA 01063 (albertson@smith.smith.edu and rhaas@smith.smith.edu).

There is a combinatorial characterization of planar rigidity which in our language is stated as follows.

THEOREM 1.3 (Laman, see [2, p. 243]). *A graph is minimally rigid in the plane if it is bounded above by $f(n) = 2n - 3$ for subgraphs on two or more vertices.*

In higher dimensions a minimally rigid graph must have a bounded number of edges, but this condition is not sufficient. Specifically, if a graph is minimally rigid in 3 space then it is bounded above by $f(n) = 3n - 6$.

There are other properties of graphs for which belonging to a family bounded by a function is only necessary and not sufficient. For example, maximal planar graphs must be bounded by $f(n) = 3n - 6$. Graphs that are decomposable into k disjoint Hamiltonian cycles must be strictly bounded by $f(n) = kn$, but again this condition is not sufficient. For instance, there are graphs on 6 vertices with 12 edges that are strictly bounded by $f(n) = 2n$ but will not have two disjoint Hamiltonian cycles.

It would be interesting to determine which graph properties are characterized by, or just imply, a bounding function. In this paper we address a more fundamental issue, namely which functions bound graphs. Section 2 constructs families of graphs bounded by every reasonable linear function. Section 3 provides constructions for many quadratic functions. Section 4 considers functions between linear and quadratic with only modest success. Section 5 provides an extended definition of bounding and examines the interrelationships between the definitions.

2. Linear functions. In this section we address the question of which linear functions bound graphs. Specifically, we obtain the following two theorems.

THEOREM 2.1. *Given rational $c \geq 1$ and integer $d \geq 0$, the function $f(n) = cn+d$ strictly bounds graphs from above.*

THEOREM 2.2. *Given c a positive multiple of $\frac{1}{2}$ and integer $d \geq -c$, the function $f(n) = cn + d$ bounds graphs from above.*

The condition that $d \geq c$ is in some sense the best possible. If $d < -c$ then $f(1) < 0$, and the inequality would fail for subgraphs of one vertex.

Our constructions depend on combining families of graphs while preserving the fact that they are bounded. The simplest such combination is an edge union. If G and G^* are edge disjoint graphs on the vertex set V with edge sets E and E^*, then we define the graph $G+G^*$ to be the graph whose edge set is $E \cup E^*$. If $\mathcal{G}$ and $\mathcal{G}^*$ are families of graphs such that whenever $G \in \mathcal{G}$ and $G^* \in \mathcal{G}^*$ have the same number of vertices, there are vertex labelings that make G and G^* edge disjoint; then we define $\mathcal{G} + \mathcal{G}^*$ to be the family of graphs whose individuals are $G + G^*$.

THEOREM 2.3. *If $\mathcal{G}$ and $\mathcal{G}^*$ are families of pairwise edge disjoint graphs bounded above by f and f^*, then $\mathcal{G} + \mathcal{G}^*$ is bounded above by $f + f^*$. Moreover, if f or f^* bounds strictly then so does $f + f^*$.*

Proof. Let H be a subgraph of $G + G^*$ on some vertex set $V_H \subset V_G$. Each edge of E_H comes either from E_G or E_{G^*} but not both. Thus $|E_H| = |E_{H \cap G}| + |E_{H \cap G^*}| \leq f(|V_H|) + f^*(|V_{H^*}|) = (f + f^*)(|V_H|)$. If at least one of f or f^* bounds strictly then the inequality is strict. □

From this theorem we can immediately construct families of graphs with n vertices (for n sufficiently large) that are strictly bounded above by $f(n) = cn$ for any integer constant c. These are graphs whose edge sets are decomposable into c Hamiltonian cycles. A more general result can also be obtained using perfect matchings.

COROLLARY 2.4. *Any family of graphs, each of whose members consists of c perfect matchings (and is thus c-regular), is bounded above by $f(n) = \frac{c}{2}n$.*

Let $\mathcal{G}$ be a family of graphs (strictly) bounded by $f(n)$. If $G \in \mathcal{G}$ has n vertices and $f(n) \leq \binom{n}{2} - d$, then we can construct a graph G^* with n vertices and d edges

that is edge disjoint from G. If there is an infinite number of G's such that this can be done, then by Theorem 2.3 $f(n)+d$ (strictly) bounds the family $\mathcal{G}+\mathcal{G}^*$. For example, if we take $f(n) = cn$ for positive integer c, then $cn+d$ bounds for all positive integers c and d.

To prove Theorem 2.1 it suffices to show that for rational $c \geq 1$, $f(n) = cn$ strictly bounds. We construct a family of strictly f bounded graphs where $f(n) = (1 + \frac{p}{q})n$ where p and q are integers and $p < \frac{q}{2}$. These graphs will have vertices of degree 2 and 3 only. Since every graph with minimum degree at least $\frac{n}{2}$ has a Hamiltonian cycle, the complement of such a graph on n vertices will contain at least $\frac{n}{4} - 2$ disjoint Hamiltonian cycles.

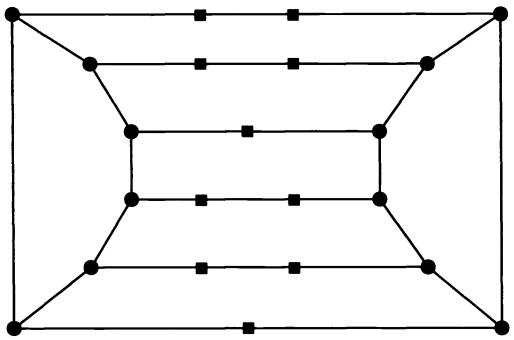


FIG. 1. *The ladder for* $p = 3$, $q = 11$, *and* $t = 1$.

Fix such a p and q. For each integer t we construct G_t, a graph with $n = 2tq$ vertices and $(1 + \frac{p}{q})n$ edges. The construction begins with the Cartesian product of an edge and the cycle C_{2tp}. Alternatively one can imagine two disjoint copies of C_{2tp} where corresponding vertices in the two cycles are adjacent. This graph is called a ladder and the edges between the two cycles are called rungs. The remaining $t(2q-4p)$ vertices will be partitioned into $2tp$ parts, each part containing either $\lfloor (q-2p)/p \rfloor$ or $\lceil (q-2p)/p \rceil$ vertices. Each rung is subdivided by the vertices of one part. The resulting graph G_t contains $2tq$ vertices, $t(2q-4p)$ of degree 2, and $4tp$ of degree 3. Thus, it contains $(1 + \frac{p}{q})2tq$ edges. Figure 1 illustrates this construction with $p = 3, q = 11$, and $t = 1$. To see that G_t is strictly bounded we look at the average degree of its subgraphs. If a graph has average degree greater than 2 then its average degree can be increased by deleting all vertices of degree 1. The reader may verify that any proper subgraph of G_t that contains no vertices of degree 1 must have lower average degree. Consequently, G_t is strictly bounded. □

To prove Theorem 2.2 we use Theorem 2.3 to construct a family of graphs bounded by $cn - d$ for positive integer $d \leq c$ and c a positive multiple of $\frac{1}{2}$. Graphs in this family are those that are decomposable into d Hamiltonian paths and $2(c-d)$ perfect matchings.

3. Quadratic functions. DEFINITION 3.1. *Given a graph G on n vertices we compose k copies of G as follows. Make k vertex disjoint copies of G and from each vertex in a particular copy of G put an edge to all vertices in all other copies of G. This gives a graph with a total of $m = kn$ vertices. This new graph will be denoted by the symbol $K_k(G)$.*

THEOREM 3.2. *Suppose $\mathcal{G}$ is a family of graphs bounded by the function $f(n)$ where $f'' \leq 1$. Then the graph $K_k(G)$ is bounded by the function $F(m) := kf(\frac{m}{k}) + (\frac{k-1}{k})\frac{m^2}{2}$. Further, if $f(n)$ strictly bounds then so does $F(m)$.*

Proof. It is straightforward to check that $K_k(G)$ contains $F(m)$ edges. We show the case where $f(n)$ and consequently $F(m)$ bounds strictly. To show that the family is strictly bounded we look at all subgraphs on V vertices ($V < m$). Suppose the subgraph consists of v_i vertices from the ith copy of G, $i = 1, \ldots, k$ and $v_i > 0$ for all i. The edges of the subgraph come from within the copies of G and between the copies of G. The number of edges that come from between is given by $(\frac{1}{2})\Sigma v_i(V - v_i)$, and the number of edges that come from within the copies of G is less than $\Sigma f(v_i)$. Combining these we get that the total number of edges in the subgraph is less than

$$\left(\frac{1}{2}\right)\Sigma v_i(V - v_i) + \Sigma f(v_i).$$

We will show that this function achieves its maximum value when all v_i are equal. The value at the maximum is then

$$\left(\frac{1}{2}\right)k\frac{V}{k}\frac{(k-1)V}{k} + kf\left(\frac{V}{k}\right) = F(V).$$

Consider two copies of G in $K_k(G)$, say G_i and G_j. The number of edges from these copies to all the others is equal regardless of how the vertices are divided between G_i and G_j. Suppose G_i has $W - t$ vertices and G_j has $W + t$ vertices. (W need not be an integer.) The number of edges in G_i and G_j and between these two parts is then less than

$$g(t) = (W - t)(W + t) + f(W - t) + f(W + t).$$

We use the first degree Taylor polynomial approximation of f near W to get that

$$g(t) = W^2 - t^2 + 2f(W) + f'(W)(-t + t) + \frac{(-t)^2}{2}f''(c_1) + \frac{t^2}{2}f''(c_2)$$

for some $W - t < c_1 < W$ and $W < c_2 < W + t$. By assumption, $f''(c) \leq 1$ for any value of c. Thus, $g(t) \leq W^2 + 2f(W)$.

Hence, the number of edges that can come from within and between just two parts is bounded by the case that these two parts have an equal number of vertices. Since this argument can be made for any pair of copies of G the maximum number of edges occurs when all the parts have an equal number of vertices. □

4. $\frac{n}{2}\log_2 n + n$ bounds. Let Q_r denote the standard r-cube. Q_r has 2^r vertices and $r2^{r-1}$ edges. So a cube with n vertices has $\frac{n}{2}\log_2 n$ edges.

THEOREM 4.1. *The cubes form a family that is bounded by* $f(n) = \frac{n}{2}\log_2 n$.

Proof. The proof will be by induction on the dimension of the cube. The base case is elementary. Suppose H is a subgraph of Q_r with m vertices. Let H_1 be the restriction of H to those vertices whose first coordinate in Q_r is 1, while H_2 is the restriction of H to those vertices whose first coordinate in Q_r is 0. Both H_1 and H_2 are subgraphs of Q_{r-1}. Let m_1 denote $|V_{H_1}|$. Then $|V_{H_2}| = m - m_1$. We may assume that $m_1 \leq m - m_1$. The edges in H are either in H_1, in H_2, or between these two. Since H_1 and H_2 are bounded by the inductive hypothesis, we obtain

$$|E_H| \leq \frac{m_1}{2}\log_2 m_1 + \frac{m - m_1}{2}\log_2 (m - m_1) + m_1. \tag{1}$$

It is straightforward to see that the second derivative of the right-hand side of the preceding equation is positive. Thus a maximum value must occur at one of the endpoints. From this it is immediate that $|E_H| \leq \frac{m}{2}\log_2 m$. □

By Theorem 2.3, for large enough n we can add a Hamiltonian cycle to Q_r. We then obtain the following.

COROLLARY 4.2. $f(n) = \frac{n}{2}\log_2 n + n$ *bounds strictly.*

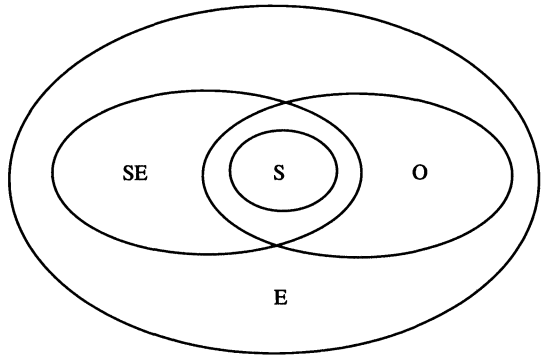


FIG. 2. *Relationship between various notions of bounding.*

5. Extensions. In our original example of rigidity, the bounding condition only needed to hold for subgraphs on two or more vertices. This suggests that we should use a broader definition for functions that bound from above.

Extended definition. A function f *bounds graphs from above* if there exists an infinite family of graphs $\mathcal{G}$ and an integer $n_0 > 0$ such that if $G \in \mathcal{G}$ then $f(|V(G)|) = |E(G)|$, and for all subgraphs H of G with at least n_0 vertices we have that $f(|V(H)|) \geq |E(H)|$. As before, we say f strictly bounds if the inequality is strict for all proper subgraphs with at least $n_0 > 0$ vertices.

Most of our results still hold under this extended definition. We will get a larger class of functions that bound and that strictly bound this way. For example, the function $f(n) = an - (2a - 1)$, with $a \geq 3$ an integer, strictly bounds graphs from above as witnessed by the family $C_{n-2}^{a-2} \vee K_2$.

Let S be the set of functions that strictly bound under the original definition, O the set of functions that bound under the original definition, E the functions that bound under the extended definition, and SE the set of functions that strictly bound under the extended definition. Figure 2 shows the relationships between these.

We show that the sets are arranged as indicated by the figure. It is clear that any function that bounds strictly also bounds and that any function that bounds under the original definition will bound under the extended definition. The function $f(n) = n - 1$ is in O and not in SE, while $2n - 4$ is in SE but not in O. Functions of the form $cn - c$ for $c \geq 2$ are in both SE (for subgraphs on two or more vertices) and O, and not in S. We have no examples to show that $SE \bigcup O \neq E$, but we suspect this to be so. The functions $n - d$ for $d \geq 2$ are examples of increasing functions for which there are graphs with n vertices and $n - d$ edges, but yet these are not bounding functions.

REFERENCES

[1] J. GRAVER AND B. SERVATIUS, *Combinatorial Rigidity*, Graduate Studies in Mathematics 2, American Mathematical Society, Providence, RI, 1993.

[2] A. RECSKI, *Matroid Theory and Its Applications in Electrical Network Theory and in Statics*, Springer-Verlag, Berlin, 1989.

[3] W. T. TUTTE, *On the problem of decomposing a graph into n-connected factors*, J. London Math. Soc., 36 (1961), pp. 221–230.

[4] W. WHITELEY, *Matroids and rigid structures*, in Matroid Applications, N. White, ed., Encyclopedia of Mathematics and its Applications 40, Cambridge University Press, Cambridge, England, 1992, pp. 1–53.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 274–289, May 1996

010

A LINEAR ALGORITHM FOR MAXIMUM WEIGHT CLIQUES IN PROPER CIRCULAR ARC GRAPHS*

BINAY BHATTACHARYA†, PAVOL HELL†, AND JING HUANG‡

Abstract. We present an $O(n)$ algorithm to find a maximum clique in a proper circular arc graph. We assume that the input graph is represented by a sorted simple family of circular arcs or by an equivalent representation. In Deng, Hell, and Huang [*SIAM J. Comput.*, 25 (1996), pp. 390–403], we gave an $O(m+n)$ algorithm to find such a representation for a proper circular arc graph given by its adjacency lists. As an application we also give an $O(n)$ algorithm for q-coloring proper circular arc graphs for a fixed q. (Such an algorithm was first given by Teng and Tucker.) Finally we indicate how our algorithm can be modified to find a maximum weight clique in a weighted graph, also in time $O(n)$.

Key words. proper circular arc graph, maximum clique, maximum weight clique, coloring, algorithms, complexity

AMS subject classifications. 05C85, 68Q25

1. Introduction.

An undirected graph G is a *circular arc graph* if there is a one-to-one correspondence between the vertex set $V(G)$ and a family $\mathcal{F}$ of arcs on a circle such that two vertices are adjacent if and only if the corresponding two circular arcs intersect. The family $\mathcal{F}$ is called a *representation* of G. A *proper* circular arc graph is a circular arc graph which admits a representation by an inclusion-free family $\mathcal{F}$. We shall, in this paper, be mostly concerned with proper circular arc graphs, and, unless explicitly stated otherwise, shall only consider inclusion-free representations. A *sorted* representation is a representation $\mathcal{F} = A_1, A_2, \ldots, A_n$ in which the specified circular order of the arcs is such that the clockwise endpoints (and since $\mathcal{F}$ is inclusion-free, also the counterclockwise endpoints) appear in their clockwise circular order. A *simple* representation $\mathcal{F}$ is a representation in which no two arcs share an endpoint or cover the entire circle. It is well known that a proper circular arc graph always admits a simple representation [5]. In fact, according to [10], an arbitrary representation can be made simple in time $O(n)$. (We use n to denote the number of vertices and m the number of edges.) However, obtaining a sorted representation, in general, requires time proportional to $n \log n$ because it entails sorting the endpoints. On the other hand, for an arbitrary graph G (given by its adjacency lists), we can, in time $O(m+n)$, decide whether or not G is a proper circular arc graph, and, if it is, obtain a sorted simple representation of it [2, 3]. Our algorithms apply to sorted simple representations.

Proper circular arc graphs have been widely studied, cf. [4, 5, 6, 9, 18], and have many applications, including those in traffic control [15], cyclic scheduling, and compiler design [17].

A *clique* of a graph (or an oriented graph) is just a complete subgraph. A clique of maximum size is called a *maximum clique.* Maximum cliques in circular arc graphs can be found in time $O(m \log\log n + n \log n)$ [13]. Recently this was improved to an algorithm of complexity $O(m + n \log n)$ in [1]. For proper circular arc graphs, Manacher and Mankus [10] gave an $O(n \log\log n)$ algorithm to find a maximum clique,

* Received by the editors December 7, 1992; accepted for publication (in revised form) June 30, 1995.

† School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada.

‡ Department of Mathematics and Computer Science, Odense University, Odense DK-5230, Denmark.

assuming that a sorted simple representation is given. In this paper we give an $O(n)$ time algorithm, also assuming that a sorted simple representation is given. (While revising this paper we learned that Manacher and Mankus also recently improved their algorithm to time $O(n)$ [11].) Moreover, our algorithm can be adapted to find the maximum weight clique in a weighted proper circular arc graph also in time $O(n)$. (The improved algorithm [11] yields $O(n^2)$ for the weighted case. The algorithm of [13], of time complexity $O(m \log\log n + n \log n)$, mentioned above, also applies in the weighted case.)

The problem of q-coloring proper circular arc graphs arose in the cyclic scheduling and register allocation applications. It was first studied by Orlin, Bonuccelli, and Bovet [12]. The best current algorithm for finding the chromatic number of a proper circular arc graph is due to Shih and Hsu and has a time bound of $O(n^{\frac{3}{2}})$ [14]. On the other hand, for a fixed q, Teng and Tucker [16] gave an $O(n)$ algorithm for q-coloring proper circular arc graphs given by a sorted simple representation. We shall show how to use our maximum clique algorithm to derive a q-coloring algorithm which is also $O(n)$ (for any fixed q).

We first introduce a purely combinatorial description of representations. It is possible to formulate our algorithms in terms of circular arcs, but we found it more convenient to use the following alternate description. Let D be an oriented graph, i.e., a digraph without directed two-cycles. We write $x \rightarrow y$ and say that x *dominates* y (or that y is *dominated by* x) when xy is an arc of D; we write $x \sim y$ and say that x and y are *adjacent* if $x \rightarrow y$ or $y \rightarrow x$. We write $A \rightarrow B$ and say that A dominates B (or that B is dominated by A) if each vertex of A dominates each vertex of B for arbitrary sets A, B of vertices of D. A *round enumeration* of D is a circular ordering of its vertices $v_1, v_2, \ldots, v_n$, such that for each i there exist nonnegative integers l_i and r_i with $v_i v_j$ being an arc if and only if $i+1 \leq j \leq i+r_i$ and $v_k v_i$ being an arc if and only if $i-l_i \leq k \leq i-1$ (all additions and subtractions are modulo n). We define $R(u) = w$ just if $u = v_i$ and $w = v_{i+r_i}$ and $L(u) = w$ just if $u = v_i$ and $w = v_{i-l_i}$. An oriented graph D is *round* if it admits a round enumeration. A round enumeration of D *with complete FO-information* is a round enumeration $v_1, v_2, \ldots, v_n$ of D, together with all values $R(v_i)$ for $i = 1, 2, \ldots, n$ (the "Farthest Out-neighbors" of all vertices).

It is helpful to visualize a round enumeration of an oriented graph D by drawing the vertices $v_1, v_2, \ldots, v_n$ clockwise around the circle, as illustrated in Fig. 1b. The FO-information associated with this example consists of the values $R(v_1) = v_5, R(v_2) = v_5$, $R(v_5) = v_6 \ldots R(v_7) = v_2$. If $a = v_i$ and $b = v_j$ are two vertices in a round enumeration $v_1, v_2, \ldots, v_n$ of D, then we define the *interval* $[a, b]$ as the set $\{v_i, v_{i+1}, v_{i+2}, \ldots, v_{j-1}, v_j\}$, with the subscripts calculated modulo n. Thus in our illustrative representations, the interval $[a, b]$ extends from a to b clockwise. The intervals $(a, b), (a, b]$, and $[a, b)$ are defined analogously. If $a \rightarrow b$ then $b \in (a, R(a)]$, and therefore $a \rightarrow d$ for each $d \in (a, b] \subseteq (a, R(a)]$. Furthermore, each $d \in (a, b]$ is dominated by every $c \in [a, d)$ by a similar argument. Thus, we conclude that if $a \rightarrow b$ and $[c, d] \subseteq [a, b]$, then also $c \rightarrow d$. This means that each interval $[a, R(a)]$ is a transitive tournament in D, and in particular a clique of D. Consider, for instance, the tournament induced by $[v_1, R(v_1)] = \{v_1, v_2, v_3, v_4, v_5\}$ in Fig. 1b. Because of the above property, we can simplify our illustrations by not depicting arcs cd such that $[c, d]$ is properly included in some interval $[a, b]$. In Fig. 1b, for instance, the interval $[v_1, R(v_1)] = \{v_1, v_2, v_3, v_4, v_5\}$ properly contains the intervals $[v_1, v_4]$ and $[v_2, v_5]$ and so the arcs $v_1 v_4, v_2 v_5$ (and also $v_1 v_2, v_1 v_3$, etc.) are not pictured in Fig. 1c. Note that this convention may result even in some arcs $aR(a)$ not being pictured, such as the arc $v_2 v_5$ in Fig. 1c. Thus in these symbolic depictions (Figs. 1c, 2b, and 3) the

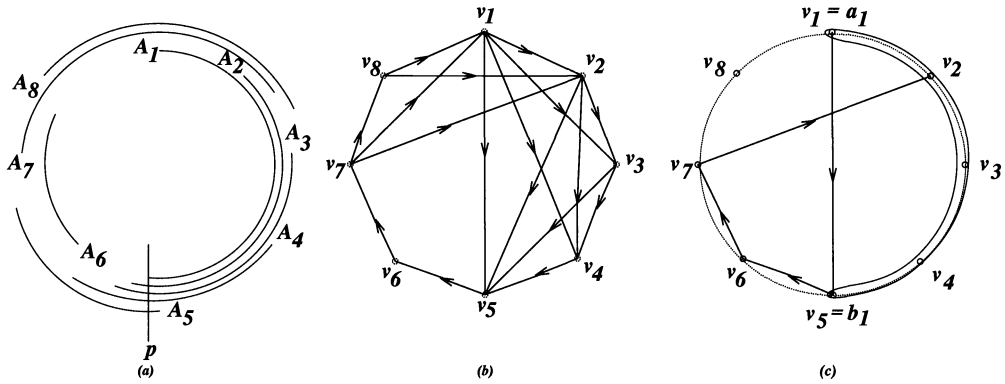


FIG. 1. (a) *an inclusion-free family of circular arcs,* (b) *the corresponding round enumeration, and* (c) *a simplified depiction of the round enumeration.*

presence of each arc ab implies the existence of all arcs cd with $[c,d] \subset [a,b]$.

Note that if $v_1, v_2, \ldots, v_n$ is a round enumeration of D, then $v_2, \ldots, v_n$ is a round enumeration of $D - v_1$. Hence we may omit vertices from the round enumeration maintaining a round enumeration of the corresponding induced subgraph.

We first establish a correspondence between round enumerations of D with complete FO-information and sorted simple representations of the underlying graph of D.

LEMMA 1.1. *A graph is a proper circular arc graph if and only if it can be oriented so that the resulting oriented graph is round.*

Proof. Suppose that G is given by a sorted simple representation $\mathcal{F} = A_1, A_2, \ldots, A_n$. Let v_i be the vertex of G corresponding to the arc A_i. Let D be the orientation of G in which $v_i \rightarrow v_j$ just if A_i contains the counterclockwise endpoint of A_j. This is an oriented graph because no two arcs of $\mathcal{F}$ cover the entire circle; i.e., we cannot have both arcs $v_i \rightarrow v_j$ and $v_j \rightarrow v_i$. It is easy to see that D is round, and that the circular order $v_1, v_2, \ldots, v_n$ is a round enumeration of D. Moreover, $R(v_i) = v_j$, provided A_j is the last arc, in the clockwise circular direction, intersecting A_i. Conversely, suppose that G is given by a round enumeration $v_1, v_2, \ldots, v_n$ with complete FO-information (thus with all values r_i known). Construct a circle from the real interval $[0, n]$ by identifying 0 and n. Associate with each v_i the circular arc from i to $i + r_i + 1 - \frac{r_i+1}{n+1}$. This produces a sorted simple representation of G. □

In Fig. 1a,b we illustrate the above correspondence between a representation by circular arcs and a round enumeration.

This proof yields easy $O(n)$ time transformations between a sorted simple representation of G and a round enumeration with complete FO-information of a suitable orientation of G. Thus, from now on, we shall be searching for a maximum clique (q-coloring, maximum weight clique) in a round oriented graph D with complete FO-information.

The oriented graph D is *connected* if its underlying graph G is connected; the *degree* of a vertex v in D is just the degree of v in G; the *maximum degree* of D, denoted by $\Delta(D)$, is just the maximum degree of G. For our purposes, D may be assumed to be connected and have no vertices of degree $n - 1$. Indeed, both the maximum clique and coloring problems can be solved for each component separately. Furthermore, any maximum clique must contain all vertices of degree $n - 1$, and any coloring must assign each vertex of degree $n - 1$ a color not used by any other

vertex. Thus it is sufficient to solve both problems for the graph obtained by removing all vertices of degree $n-1$. Moreover, having our graph D represented by a round enumeration with complete FO-information, we can easily identify the components and the vertices of degree $n-1$ in time $O(n)$. Therefore we shall assume from now on that D is a connected graph with $\Delta(D) \leq n-2$. This assumption implies that for every vertex u there is at least one nonneighbor of u between $R(u)$ and $L(u)$. Thus, in our illustrations, for any vertex u, we first encounter (moving clockwise from u) all out-neighbors of u (the last one being $R(u)$), then all nonneighbors of u (of which there is at least one), and, finally, just before returning to u, all in-neighbors of u (the first one being $L(u)$); cf. Fig. 1b.

2. The maximum clique algorithm. We first give some geometric intuition; the concepts suggested below will all be formally defined later, in the context of round enumerations.

Orlin, Bonuccelli, and Bovet [12] introduced the notion of an overlap clique. Given a family of circular arcs and a point p of the circle, the *overlap clique* of p is the set of all circular arcs from the given family which contains the point p. For instance, for the family in Fig. 1a and the point of the circle corresponding to 3 o'clock, we obtain the overlap clique $\{A_1, A_2, A_3\}$.

Although every overlap clique is a clique, it is not the case that a maximum clique must be an overlap clique. (But note Corollary 2.3.) In Fig. 2a there is a family of circular arcs for which the largest overlap clique has nine elements (nine arcs contain the point p_1 and no point is contained in ten arcs), but there is a clique with eleven elements: Indeed, consider the three points p_1, p_2, p_3 depicted in Fig. 2a and all circular arcs that contain at least two of these three points. There are 11 such circular arcs, $A_1, A_2, A_3, A_4, A_5, A_7, A_8, A_9, A_{13}, A_{14}, A_{15}$, and it is clear that any two intersect.

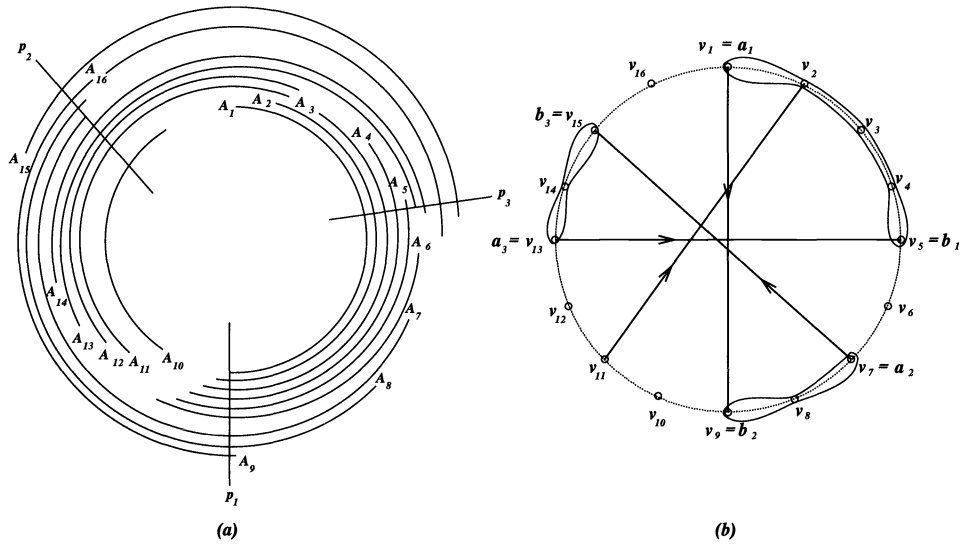


FIG. 2. (a) *a clique generated by three points* p_1, p_2, p_3 *on the circle, and* (b) *the corresponding* 3*-overlap clique.*

This suggests a generalization of overlap cliques: For any family $\mathcal{F}$ of circular arcs (whether inclusion-free or not), any odd integer $\mu = 2\mu' + 1$, and any set of μ

points of the circle, the set of all arcs of $\mathcal{F}$ which contain at least $\mu' + 1$ of the chosen points is a clique. We can show (cf. [8]) that for any family of circular arcs (whether inclusion-free or not) some maximum clique must be of this form. In this paper, we will prove this only for proper circular arc graphs and will do it in the language of round enumerations.

In order to motivate the appropriate definitions in terms of round enumerations, we first note that the circular arcs in a usual overlap clique correspond to consecutive vertices in the corresponding round enumeration, which span some interval $[x, y]$. Corresponding to $\{A_1, A_2, A_3\}$ in Fig. 1a, we have in Fig. 1b the clique $[v_1, v_3]$. It is clear that if p is chosen to be the clockwise endpoint of some arc x (e.g., of A_1 in Fig. 1a), then, in the corresponding round enumeration, an overlap clique spans the interval $[x, R(x)]$ ($[v_1, v_5]$ in Fig. 1b).

In Fig. 2b, the vertices corresponding to the eleven circular arcs $A_1, A_2, A_3, A_4, A_5, A_7, A_8, A_9, A_{13}, A_{14}, A_{15}$ form a clique which consists of the three intervals $[v_1, v_5]$, $[v_7, v_9]$, and $[v_{13}, v_{15}]$. Here the interval $[v_1, v_5]$ consists of the vertices corresponding to the circular arcs containing p_1 and p_3, the interval $[v_7, v_9]$ consists of the vertices corresponding to the circular arcs containing p_1 and p_2, and the interval $[v_{13}, v_{15}]$ consists of the vertices corresponding to the circular arcs containing p_2 and p_3. Note that $R(v_1) = v_9, R(v_7) = v_{15}$, and $R(v_{13}) = v_5$—the beginning of the ith interval is associated with the end of the $(i+1)$st interval. This suggests, in a general situation, to take vertices $a_1, b_1, a_2, b_2, \ldots, a_\mu, b_\mu$, in circular order, such that each $R(a_i) = b_{i+\mu'}$. It is then easy to see that $[a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ is a clique.

We now give the formal definitions. (These are independent of the above geometric intuition, which should be viewed only as a rough motivation.) Assume that D is a round oriented graph with a given round enumeration. Let $\mu = 2\mu' + 1 \geq 3$, and let $a_1, b_1, a_2, b_2, \ldots, a_\mu, b_\mu$ be vertices of D listed in clockwise circular order, such that for each $i = 1, 2, \ldots, \mu$,

$$R(a_i) = b_{i+\mu'} \quad \text{and} \quad |[a_i, b_i]| > |(b_{i+\mu'}, a_{i+\mu'+1})|$$

where the subscript additions are modulo μ. Then we say that $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ is a *μ-overlap clique* (*generated by* $a_1, a_2, \ldots, a_\mu$). We also refer to the vertices $a_1, a_2, \ldots, a_\mu$ as the *generators* of C.

The condition $|[a_i, b_i]| > |(b_{i+\mu'}, a_{i+\mu'+1})|$ is assumed only for technical convenience (cf. the proof of Claim 6). If, for instance, in our example in Fig. 2b we had $|[a_3, b_3]| \leq |(b_1, a_2)|$, then it would be better to replace $[a_3, b_3]$ by (b_1, a_2), thereby obtaining the 1-overlap clique $[a_1, R(a_1)]$.

Note that for each $i = 1, 2, \ldots, \mu$ there is at least one vertex between $b_{i+\mu'}$ and $a_{i+\mu'+1}$, because otherwise the vertex a_i has degree $n-1$, contrary to our assumption (cf. Fig. 2b). By our cardinality restriction this means that each interval $[a_i, b_i]$ has at least two vertices, i.e., that $a_i \neq b_i$.

For convenience we also define 1- and (-1)-overlap cliques: A *1-overlap clique* (*generated by* a_1) is any interval $[a_1, b_1]$ with $b_1 = R(a_1)$ and a *(-1)-overlap clique* is just the empty set $\emptyset$.

LEMMA 2.1. *There exists a maximum clique which is a μ-overlap clique for some odd integer μ.*

Proof. Let C be a set of vertices which induces a maximum clique of D. If $C \neq \emptyset$, then there exists an integer μ such that C may be written as $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ where $a_1, b_1, a_2, b_2, \ldots, a_\mu, b_\mu$ appear in this clockwise circular order in the round enumeration of D. Let C and μ be chosen so that μ is as small

as possible among all maximum cliques of G. Then, by the minimality of μ, every (b_i, a_{i+1}) contains at least one vertex. This implies that b_i and a_{i+1} are distinct and it will be seen from Claim 6 below that a_i and b_i are also distinct.

Suppose that $\mu = 1$, i.e., that $C = [a_1, b_1]$. Consider the adjacent vertices a_1, b_1. If $b_1 \rightarrow a_1$, then $[b_1, a_1]$ is also a clique, contradicting $\Delta(D) \leq n-2$. Thus $a_1 \rightarrow b_1$, and hence $R(a_1) \in [b_1, a_1)$. Since $[a_1, b_1]$ is a maximum clique, $R(a_1) = b_1$ and C is a 1-overlap clique. Thus suppose for the rest of this proof that $\mu > 1$.

Claim 1. If $u \notin [a_i, b_i]$ is adjacent to all vertices of $[a_i, b_i]$ then $u \rightarrow [a_i, b_i]$ or $[a_i, b_i] \rightarrow u$.

Suppose there are two vertices $x, y \in [a_i, b_i]$ such that $u \rightarrow x$ and $y \rightarrow u$. This means that $u \rightarrow (u, x]$ and $[y, u) \rightarrow u$. Together with the assumption that u is adjacent to each vertex of $[a_i, b_i]$, we contradict the fact that $\Delta(D) \leq n-2$.

Claim 2. If $i \neq j$ then either $[a_i, b_i] \rightarrow [a_j, b_j]$ or $[a_j, b_j] \rightarrow [a_i, b_i]$.

If $a_i \rightarrow b_j$ then $x \rightarrow y$ for each $x \in [a_i, b_i]$ and $y \in [a_j, b_j]$. On the other hand, if $b_j \rightarrow a_i$ then two applications of Claim 1 yield $b_j \rightarrow b_i$ and $a_j \rightarrow b_i$. Thus in this case $x \rightarrow y$ for each $x \in [a_j, b_j]$ and $y \in [a_i, b_i]$.

Claim 3. If $[a_i, b_i] \rightarrow [a_j, b_j]$ then $[a_{i+1}, b_{i+1}] \rightarrow [a_{j+1}, b_{j+1}]$.

Suppose that $[a_i, b_i] \rightarrow [a_j, b_j]$ and $[a_{j+1}, b_{j+1}] \rightarrow [a_{i+1}, b_{i+1}]$. Let $u \in (b_i, a_{i+1})$. (It was noted above that $(b_i, a_{i+1}) \neq \emptyset$.) Then u is adjacent to all vertices of $[a_{i+1}, b_{i+1}] \cup \ldots \cup [a_j, b_j]$ because $a_i \rightarrow b_j$ and to all vertices of $[a_{j+1}, b_{j+1}] \cup \ldots \cup [a_i, b_i]$ because $a_{j+1} \rightarrow b_{i+1}$. This contradicts the maximality of our clique.

Claim 4. μ is odd (say, $\mu = 2\mu' + 1$).

If μ is even, then $[a_i, b_i] \rightarrow [a_{i+\frac{\mu}{2}}, b_{i+\frac{\mu}{2}}]$ implies $[a_{i+\frac{\mu}{2}}, b_{i+\frac{\mu}{2}}] \rightarrow [a_i, b_i]$ by Claim 3, contrary to Claim 2.

Claim 5. $R(a_i) = b_{i+\mu'}$ for each $i = 1, 2, \ldots, \mu$.

Since a_i and $b_{i+\mu'}$ are in C, they are adjacent. If some $b_{i+\mu'} \rightarrow a_i$, then Claim 2 implies that $[a_{i+\mu'}, b_{i+\mu'}] \rightarrow [a_i, b_i]$ and Claim 3 implies that $[a_i, b_i] \rightarrow [a_{i+\mu'+1}, b_{i+\mu'+1}]$. However, this is impossible, as $a_i \rightarrow b_{i+\mu'+1}$ implies $a_i \rightarrow b_{i+\mu'}$. Hence $a_i \rightarrow b_{i+\mu'}$ for each $i = 1, 2, \ldots, \mu$. In particular, $a_{i+\mu'+1} \rightarrow b_{i+2\mu'+1} = b_i$. So $R(a_i) \in [b_{i+\mu'}, a_{i+\mu'+1})$. On the other hand, if $R(a_i) \neq b_{i+\mu'}$, then $R(a_i)$ is adjacent to every vertex of $[a_i, b_i] \cup [a_{i+1}, b_{i+1}] \cup \ldots \cup [a_{i+\mu'}, b_{i+\mu'}]$, and because $a_{i+\mu'} \rightarrow b_{i+2\mu'} = b_{i-1}$, $R(a_i)$ is also adjacent to every vertex of $[a_{i+\mu'+1}, b_{i+\mu'+1}] \cup \ldots \cup [a_{i-1}, b_{i-1}]$. Thus $R(a_i)$ is adjacent to every vertex of C, contradicting its maximality. Therefore $R(a_i) = b_{i+\mu'}$.

Claim 6. $|[a_i, b_i]| > |(b_{i+\mu'}, a_{i+\mu'+1})|$ for each $i = 1, 2, \ldots, \mu$.

If $|[a_i, b_i]| \leq |(b_{i+\mu'}, a_{i+\mu'+1})|$ for some i, then let $C' = [a_1, b_1] \cup \ldots \cup [a_{i-2}, b_{i-2}] \cup [a_{i-1}, b_{i-1}] \cup [a_{i+1}, b_{i+1}] \cup \ldots \cup [a_{i+\mu'-1}, b_{i+\mu'-1}] \cup [a_{i+\mu'}, b_{i+\mu'+1}] \cup [a_{i+\mu'+2}, b_{i+\mu'+2}] \cup \ldots [a_\mu, b_\mu]$. In effect, C' is obtained from C by replacing $[a_i, b_i]$ with $(b_{i+\mu'}, a_{i+\mu'+1})$. We see easily that C' is also a clique: We only need to verify that each vertex $u \in (b_{i+\mu'}, a_{i+\mu'+1})$ is adjacent to all other vertices of C'. Since $a_{i+1} \rightarrow b_{i+\mu'+1}$ and hence $a_{i+1} \rightarrow u$, we conclude u is adjacent to $[a_{i+1}, b_{i+1}] \cup \ldots \cup [a_{i+\mu'-1}, b_{i+\mu'-1}] \cup [a_{i+\mu'}, u)$; since $a_{i+\mu'} \rightarrow b_{i-1}$ and hence $u \rightarrow b_{i-1}$, we also conclude that u is adjacent to $(u, b_{i+\mu'+1}] \cup \ldots \cup [a_{i-1}, b_{i-1}]$. Thus C' is a clique with fewer intervals than C and with $|C'| \geq |C|$, contradicting the choice of C. □

For a fixed μ, we call a μ-overlap clique of maximum size a *largest μ-overlap clique.*

LEMMA 2.2. *Let $\mu \geq 3$ be an odd integer. Let $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ be a μ-overlap clique of D and suppose that $x \in (b_{i-1}, b_i]$ for some $i = 1, 2, \ldots, \mu$.*

If $|[x, R(x)]| \geq |[a_i, R(a_i)]|$, then the vertices $a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_\mu$ generate a μ-overlap clique C', with $|C'| \geq |C|$. Moreover, $|C'| = |C|$ if and only if $|[x, R(x)]| = |[a_i, R(a_i)]|$.

Proof. To prove that C' is a μ-overlap clique we need to show

1. $R(x) \in (a_{i+\mu'}, a_{i+\mu'+1})$,
2. $|[x, b_i]| > |(R(x), a_{i+\mu'+1})|$, and
3. $|[a_{i+\mu'}, R(x)]| > |(b_{i-1}, x)|$.

Since $a_{i-1} \rightarrow b_{i+\mu'-1}$, we have $x \rightarrow b_{i+\mu'-1}$. Since $a_{i+\mu'+1} \rightarrow b_i$, we also have $a_{i+\mu'+1} \rightarrow x$. Thus $R(x) \in [b_{i+\mu'-1}, a_{i+\mu'+1})$. We claim that $R(x) \notin [b_{i+\mu'-1}, a_{i+\mu'})$. Suppose, on the contrary, that $R(x) \in [b_{i+\mu'-1}, a_{i+\mu'})$. Then $x \in (b_{i-1}, a_i)$ because $a_i \rightarrow b_{i+\mu'}$. Since $|[x, R(x)]| \geq |[a_i, R(a_i)]|$, we have $|[x, a_i]| \geq |[R(x), R(a_i)]|$. But, on the other hand, we have $|[x, a_i]| \leq |(b_{i-1}, a_i)| \leq |[a_{i+\mu'}, b_{i+\mu'}]| \leq |[R(x), R(a_i)]|$, a contradiction. This proves 1.

Now we prove 2 and 3 together.

If $x \in (b_{i-1}, a_i]$, then $R(x) \notin (b_{i+\mu'}, a_{i+\mu'+1})$ as otherwise we would have $a_i \rightarrow R(x)$ contradicting the fact that $R(a_i) = b_{i+\mu'}$. Thus $R(x) \in [a_{i+\mu'}, b_{i+\mu'}]$. Then $|[x, a_i)| \geq |(R(x), b_{i+\mu'}]|$ because $|[x, R(x)]| \geq |[a_i, R(a_i)]|$, and hence

$$|[x, b_i]| = |[x, a_i)| + |[a_i, b_i]| > |(R(x), b_{i+\mu'}]| + |(b_{i+\mu'}, a_{i+\mu'+1})| = |(R(x), a_{i+\mu'+1})|$$

and

$$|[a_{i+\mu'}, R(x)]| = |[a_{i+\mu'}, b_{i+\mu'}]| - |(R(x), b_{i+\mu'}]| > |(b_{i-1}, a_i)| - |[x, a_i)| = |(b_{i-1}, x)|.$$

On the other hand, if $x \in (a_i, b_i)$, then $R(x) \in (b_{i+\mu'}, a_{i+\mu'+1})$ and $|(b_{i+\mu'}, R(x)]| \geq |[a_i, x)|$ because $|[x, R(x)]| \geq |[a_i, R(a_i)]|$. Thus

$$|[x, b_i]| = |[a_i, b_i]| - |[a_i, x)| > |(b_{i+\mu'}, a_{i+\mu'+1})| - |(b_{i+\mu'}, R(x)]| = |(R(x), a_{i+\mu'+1})|$$

and

$$|[a_{i+\mu'}, R(x)]| = |[a_{i+\mu'}, b_{i+\mu'}]| + |(b_{i+\mu'}, R(x)]| > |(b_{i-1}, a_i)| + |[a_i, x)| = |(b_{i-1}, x)|.$$

It is now easy to conclude that $|C'| \geq |C|$ because $|[x, R(x)]| \geq |[a_i, R(a_i)]|$ means $|[x, a_i]| \geq |[R(x), R(a_i)]|$ (or $|[a_i, x]| \leq |[R(a_i), R(x)]|$); similarly we can conclude that $|C'| = |C|$ if and only if $|[x, R(x)]| = |[a_i, R(a_i)]|$. □

COROLLARY 2.1. *Suppose that* $[a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ *is a largest* μ*-overlap clique. Then for every* $i = 1, 2, \ldots, \mu$ *and each* $x \in (b_{i-1}, b_i]$ *we have* $|[x, R(x)]| \leq |[a_i, R(a_i)]|$. □

COROLLARY 2.2. *Let* $\mu \geq 3$ *be an odd integer. If* $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ *is a largest* μ*-overlap clique, then for any* $x \in (b_{i-1}, b_i]$ *with* $|[x, R(x)]| = |[a_i, R(a_i)]|$, *the vertices* $a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_\mu$ *generate a* μ*-overlap clique* C', *which also is a largest* μ*-overlap clique.* □

Let $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ be a largest μ-overlap clique. Then C is called *normalized* (with respect to a_i) if, for every $j \neq i$ and each $x \in (b_{j-1}, a_j)$, $|[x, R(x)]| < |[a_j, R(a_j)]|$. Trivially, any 1-overlap clique $[a, R(a)]$ is normalized (with respect to a).

To illustrate these concepts in Fig. 2b, for $x = v_{11} \in (b_2, a_3]$ we have $|[x, R(x)]| = |[v_{11}, v_2]| < |[a_3, R(a_3)]| = |[v_{13}, v_5]|$; in fact, C is normalized with respect to v_1. However, if we added the arc $v_{11}v_3$ then $|[x, R(x)]| = |[a_3, R(a_3)]|$, and C would no longer be normalized with respect to v_1: The clique $C' = [v_1, v_3] \cup [v_7, v_9] \cup [v_{11}, v_{15}]$ would have the same cardinality as C and C' would be normalized with respect to v_1.

LEMMA 2.3. *If* x *generates a largest* 1*-overlap clique, then* x *is a generator of some maximum clique which is a* μ*-overlap clique normalized with respect to* x.

Proof. Let $[a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ be a μ-overlap clique which is also a maximum clique. Then $x \in (b_{i-1}, b_i]$ for some i. By Corollary 2.1 $|[x, R(x)]| \leq |[a_i, R(a_i)]|$. Since x generates a largest 1-overlap clique, $|[x, R(x)]| \geq |[a_i, R(a_i)]|$. Hence $|[x, R(x)]| = |[a_i, R(a_i)]|$, and by Corollary 2.2, $a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_\mu$ generate a μ-overlap clique C which is also a maximum clique.

Now to obtain a maximum clique which is a μ-overlap clique normalized with respect to x, we proceed as follows, making use of Corollary 2.2: Replace the generator $a_{i+\mu'+1}$ by the closest vertex $a'_{i+\mu'+1}$ to $R(x)$ in $(R(x), a_{i+\mu'+1}]$ with $|[a'_{i+\mu'+1}, R(a'_{i+\mu'+1})]| = |[a_{i+\mu'+1}, R(a_{i+\mu'+1})]|$, and then repeat this procedure for the generators in the order $a_{i+1}, a_{i+\mu'+2}, a_{i+2}, \ldots, a_{i-1}, a_{i+\mu'}$. □

We note the following interesting consequence of our lemma, which will simplify our task of finding a maximum clique.

COROLLARY 2.3. *If K is a largest 1-overlap clique and if $|K| \leq \frac{n}{2}$ then K is a maximum clique.*

Proof. Suppose that $K = [x, R(x)]$ is a largest 1-overlap clique which is not a maximum clique. Then, by Lemma 2.3, there is a μ-overlap clique C, with $\mu \geq 3$ and generators $x = a_1, a_2, \ldots, a_\mu$. We claim that $a_{\mu'+2}$ generates a clique of size greater than $K = |[a_1, R(a_1)]|$, contrary to the choice of K. Note that we have $|(R(a_1), a_1)| \geq |[a_1, R(a_1)]|$ since $|K| \leq \frac{n}{2}$. Hence $|[a_{\mu'+2}, R(a_{\mu'+2})]| = |[a_{\mu'+2}, a_1)| + |[a_1, R(a_{\mu'+2})]| > |[a_{\mu'+2}, a_1)| + |(R(a_1), a_{\mu'+2})| = |(R(a_1), a_1)| \geq |[a_1, R(a_1)]|$. (Since $R(a_1) = b_{\mu'+1}$ and $R(a_{\mu'+2}) = b_1$, the inequality $|[a_1, R(a_{\mu'+2})]| > |(R(a_1), a_{\mu'+2})|$ follows from the definition of a μ-overlap clique.) □

According to this corollary, we only need to search for μ-overlap cliques with $\mu > 1$ when a largest 1-overlap clique contains more than half of the vertices.

We now present our algorithm for finding a maximum clique in a round oriented graph of maximum degree smaller than $n - 1$, given by a round enumeration with complete FO-information. To simplify the notation, we write $x + k = y$ if $x = v_i$ and $y = v_{i+k}$, with the additions calculated modulo n. (Thus each $v_i + k = v_{i+k}$ and, in general, $x + k$ is the kth vertex after x in the clockwise direction.)

The algorithm first finds a vertex x_0 which generates a largest 1-overlap clique. If the clique contains more than half of the vertices, then the algorithm proceeds to repeat a basic iteration step I. The iteration step $I(x)$, using the starting vertex x (initially $x = x_0$), normally produces a vertex s which will be the starting vertex of the next iteration; if $I(x)$ does not produce an s we say that $I(x)$ *fails*. Moreover, $I(x)$ may produce another vertex g, which will be one of the generators of our final maximum clique; we note that $I(x)$ may produce a g even if it fails.

Specifically, the step $I(x)$ is defined as follows:

$g \leftarrow$ undefined, and $I(x)$ fails and $I(x)$ fails $g \leftarrow$ undefined
do until $\{x + k \sim R(x) + k + 1\}$ or $\{x + k = x_0$ and $k > 0\}$ or $\{R(x) + k + 1 = x_0\}$
$k \leftarrow k + 1$
if $\{x + k = x_0$ and $k > 0\}$ then $s \leftarrow$ undefined and $g \leftarrow$ undefined, and $I(x)$ fails
else if $\{R(x) + k + 1 = x_0\}$ then $g \leftarrow x$ and $s \leftarrow$ undefined, and $I(x)$ fails
else if $x + k \rightarrow R(x) + k + 1$ then $s \leftarrow x + k$ and $g \leftarrow$ undefined
else $s \leftarrow R(x) + k + 1$ and $g \leftarrow x$

Thus, we start from x seeking the least nonnegative integer k such that $x + k$ is adjacent to $R(x) + k + 1$. (In fact, we could have started with $k = 1$ because x is never adjacent to $R(x) + 1$.) In this search we only consider values of k up to the point when either $x + k$ or $R(x) + k + 1$ reaches the fixed vertex x_0. If neither of these happens, then $I(x)$ succeeds and terminates with $s = x + k$ and g undefined if $x + k \rightarrow R(x) + k + 1$ or terminates with $s = R(x) + k + 1$ and $g = x$ if $R(x) + k + 1 \rightarrow x + k$.

If $I(x)$ fails because $x+k$ became x_0 then it doesn't produce either g or s, and if it fails because $R(x)+k+1$ became x_0 then it produces $g=x$ but no s.

We state the entire algorithm as follows.

ALGORITHM MAX-CLIQ
find a vertex x_0 which generates a largest 1-overlap clique
if $||[x_0, R(x_0)]|| \leq \frac{n}{2}$ then halt
$x \leftarrow x_0$
do until $I(x)$ fails
perform $I(x)$
$x \leftarrow s$

The correctness of our algorithm relies on the following lemma.

LEMMA 2.4. *Let $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ be a maximum clique of D which is a μ-overlap clique normalized with respect to $x_0 = a_1$ and assume that $|C| > \frac{n}{2}$.*

(Let i denote any subscript $1, 2, \ldots, \mu$ and recall that we reduce the subscripts modulo μ; also recall that $\mu = 2\mu' + 1$.)

1. *$I(a_i)$ will always produce the value $g = a_i$.*
2. *If $i \neq \mu'+1$ then $I(a_i)$ will also produce a value $s \in (b_{\mu'+i}, a_{\mu'+i+1}]$ with $R(s) \in (a_i, b_i]$.*
3. *$I(a_{\mu'+1})$ will either produce a value $s \in (b_\mu, a_1)$ with $R(s) \in (a_{\mu'+1}, b_{\mu'+1})$, or fail.*
4. *If $x \in (b_{i-1}, a_i)$ and $R(x) \in (a_{\mu'+i}, b_{\mu'+i})$ then $I(x)$ will not produce a value of g.*
5. *If $x \in (b_{i-1}, a_i)$ and $R(x) \in (a_{\mu'+i}, b_{\mu'+i})$ and $i \neq 1$ then $I(x)$ will produce the value $s \in (x, a_i]$ with $R(s) \in (a_{\mu'+i}, b_{\mu'+i}]$.*
6. *If $x \in (b_\mu, a_1)$ and $R(x) \in (a_{\mu'+1}, b_{\mu'+1})$ then $I(x)$ will produce the value $s \in (x, a_1)$ with $R(s) \in (a_{\mu'+1}, b_{\mu'+1})$, or fail.*

Proof. Note that according to the assumption of the lemma, we have the inequality $||[a_i, b_i]|| > |(b_{\mu'+i}, a_{\mu'+i+1})|$ (from the definition of a μ-overlap clique with $\mu \geq 3$) even when $\mu = 1$.

Consider an iteration $I(a_i)$. We first observe that the value k found by the iteration is such that $a_i+k \in [a_i, b_i]$ and $R(a_i)+k+1 = b_{\mu'+i}+k+1 \in (b_{\mu'+i}, a_{\mu'+i+1}]$. Indeed, by the time k became large enough for $R(a_i)+k+1 = a_{\mu'+i+1}$ to hold, we must still have $a_i+k \in [a_i, b_i]$ (because $||[a_i, b_i]|| \geq |(b_{\mu'+i}, a_{\mu'+i+1}]|$) and hence $R(a_i)+k+1$ and a_i+k are adjacent, because $[R(a_i)+k+1, a_i+k] = [a_{\mu'+i+1}, a_i+k] \subseteq [a_{\mu'+i+1}, b_i] = [a_{\mu'+i+1}, R(a_{\mu'+i+1})]$. Since $a_i+k \in [a_i, b_i]$ we cannot have $a_i+k = x_0$. Moreover, we cannot have $a_i+k \rightarrow R(a_i)+k+1$ because in this case a_i+k would generate a larger 1-overlap clique than a_i, contradicting Corollary 2.1. Thus the iteration $I(a_i)$ will produce $g = a_i$, proving statement 1. When $i \neq \mu'+1$, then $a_{\mu'+i+1} \neq x_0 = a_1$ and $I(a_i)$ also produces a value $s = R(a_i)+k+1 \in (b_{\mu'+i}a_{\mu'+i+1}]$. We have $R(s) \in (a_i, b_i]$, again by Corollary 2.1. This proves statement 2. Statement 3 is proved analogously, the only difference being that in this case $R(a_{\mu'+1})+k+1$ may become equal to x_0 and hence $I(a_{\mu'+1})$ might fail.

To prove statement 4, assume that $x \in (b_{i-1}, a_i)$ and $R(x) \in (a_{\mu'+i}, b_{\mu'+i})$. The iteration $I(x)$ will either fail with $i = 1$ and $x+k = x_0 = a_1$ or find a value k for which $x+k \in [x, a_i]$ and $R(x)+k+1 \in (R(x), b_{\mu'+i}]$. Indeed, if $i \neq 1$ then $I(x)$ does not fail, because by the time $x+k = a_i$ we must still have $R(x)+k+1 \in (R(x), b_{\mu'+i}]$ (as $||[x, a_i]|| < ||[R(x), b_{\mu'+i}]||$ by the definition of a normalized clique). Furthermore we see that $R(x)+k+1 \rightarrow x+k$ is impossible, because otherwise $[a_1, b_1] \cup \ldots \cup [a_{\mu'+1}, R(x)] \cup$

$[u, b_{\mu'+1}] \cup [a_{\mu'+2}, b_{\mu'+2}] \cup \ldots \cup [a_\mu, b_\mu] \cup [x, v]$ is a clique larger than C, contrary to our assumption. Hence $I(x)$ will not produce a value g, proving statement 4. Moreover, when $i \neq 1$, then $I(x)$ will also produce $s = x + k \in [x, a_i]$, and by Corollary 2.1, we have $R(s) \in (R(x), b_{\mu'+i}]$. This proves statement 5. Finally, when $i = 1$, as we noted above, $I(x)$ may fail with $x + k = x_0 = a_1$ (and hence $R(s) = R(a_1) = b_{\mu'+1}$). Otherwise $I(x)$ will produce $s = x+k \in (x, a_1)$ with $R(s) \in (a_{\mu'+1}, b_{\mu'+1})$, completing the proof of statement 6. □

We now explain how to use the above lemma to prove the correctness of our algorithm. Firstly, according to Lemma 2.3, there exists a maximum clique $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ of D which is a μ-overlap clique normalized with respect to $x_0 = a_1$. Recall that $R(a_i) = b_{i+\mu'}$ for each $i = 1, 2, \ldots, \mu = 2\mu' + 1$. We claim that the sequence $g_1, g_2, \ldots, g_d$ of values g output by our algorithm is precisely the sequence $a_1, a_{\mu'+2}, a_2, a_{\mu'+3}, a_3, \ldots, a_{\mu'+1}$.

Assume first that $\mu > 1$. Our algorithm first performs the iteration $I(a_1)$. By Lemma 2.4, statement 1, $I(a_1)$ will produce the value $g_1 = a_1$. According to statement 2, $I(a_1)$ will also produce a value $s \in (b_{\mu'+1}, a_{\mu'+2}]$ with $R(x) \in (a_1, b_1]$. The algorithm will let $x \leftarrow s$ and next perform the iteration $I(x)$. Suppose first that $x \in (b_{\mu'+1}, a_{\mu'+2})$. Then statements 4 and 5 imply that no g is produced and the value of s is in the interval $(x, a_{\mu'+2}]$ and $R(s) \in (R(x), b_1]$. Since this s is used as the next x, and since each iteration moves x clockwise, we will eventually (after at most $|(b_{\mu'+1}, a_{\mu'+2}]|$ iterations) perform $I(a_{\mu'+2})$. This iteration will produce the next value of g, namely $g_2 = a_{\mu'+2}$, according to statement 1. Moreover, statement 2 applies, and we obtain an $s \in (b_1, a_2]$ with $R(s) \in (a_{\mu'+2}, b_{\mu'+2}]$. As above, the algorithm will not produce another value of g until we perform the iteration $I(a_2)$. Continuing this way, we see that our algorithm will find $g_3 = a_2, g_4 = a_{\mu'+3}, g_5 = a_3$, and so on, until we invoke the iteration $I(a_{\mu'+1})$. (In the following discussion we also include the case $\mu = 1$, when the very first iteration $I(a_1)$ is $I(a_{\mu'+1})$.) For the iteration $I(a_{\mu'+1})$ we argue in a similar fashion, using statements 3 and 6 in place of statements 2 and 5. First, $I(a_{\mu'+1})$ will produce $g_\mu = a_{\mu'+1}$. Moreover, there will be at most $|(b_\mu, a_1]|$ further iterations, which do not produce any further values g, and then we must fail.

To illustrate the algorithm and its proof of correctness, consider first the 3-overlap clique in Fig. 2b. Suppose we found (in one pass, from the complete FO-information) the vertex v_1 as the generator of a largest 1-overlap clique. The algorithm starts by performing $I(v_1)$. Thus we search for the least nonnegative integer k such that $v_1 + k$ and $R(v_1) + k + 1$ are adjacent. In our case we find that $v_1 + 1 = v_2$ is already adjacent to $R(v_1) + 2 = v_{11}$. So the least nonnegative integer k equals 1. Now we need to check whether $v_{11} \rightarrow v_2$ or $v_2 \rightarrow v_{11}$. Since $v_{11} \rightarrow v_2$, the iteration $I(v_1)$ produces $g_1 = v_1$ as the first generator, and the vertex $s = v_{11}$ as the starting vertex of next iteration. Following the same procedure, the next iteration $I(v_{11})$ finds that v_{12} is not adjacent to v_4 but v_{13} is adjacent to v_5; thus $k = 2$. Since $v_{13} \rightarrow v_5$ the iteration $I(v_{11})$ does not produce a value g, and will yield $s = v_{13}$. The next iteration $I(v_{13})$ will produce the generator $g_2 = v_{13}$ and $s = v_7$. Finally the iteration $I(v_7)$ fails because $R(v_7) + 2 = v_{15} + 2 = v_1$ is our fixed starting point (called x_0 in the algorithm). However, even though $I(v_7)$ fails, it still produces the generator $g_3 = v_7$. Thus our algorithm correctly identifies the 3-overlap clique $[v_1, v_5] \cup [v_7, v_9] \cup [v_{13}, v_{15}]$ (generated by v_1, v_{13}, v_7) as a maximum clique. To illustrate a situation in which $I(x)$ fails because $x + k$ becomes x_0, consider the example in Fig. 1b: Starting with $I(v_1)$ we produce $g_1 = v_1$ and the value $s = v_7$. Now $I(v_7)$ fails because $v_7 + 1 = v_8$ is not adjacent to $R(v_7) + 2 = v_2 + 2 = v_4$, and $v_7 + 2 = v_1$. In this situation neither g nor s is produced and the algorithm only produced one generator, corresponding to the

fact that $[v_1, R(v_1)]$ is a maximum clique.

We next consider the complexity of our algorithm. First, we can find the vertex x_0 in time $O(n)$ from the complete FO-information. For the remainder of the analysis we may assume that $|[x_0, R(x_0)]| > \frac{n}{2}$; i.e., $|[x_0, R(x_0)]| > |(R(x_0), x_0)|$. Suppose that the μ-clique generated by the algorithm is $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$, where $a_1 = x_0$. Note that the time for $I(x)$ is dominated by the time spent searching for k. We claim that the total time spent on searching for the k's over all the iterations is proportional to $|(b_1, a_2] \cup (b_2, a_3] \cup \ldots \cup (b_\mu, a_1]|$. For instance, during $I(x_0)$ we search $(R(x_0), s_1] = (b_{\mu'+1}, s_1]$ (where s_1 is the s produced by $I(x_0)$), then during $I(s_1)$ we search $(s_1, s_2]$ (where s_2 is the s produced by $I(s_1)$), and so on, until we generate $g_2 = a_{\mu'+1}$; by this time we have searched the whole interval $(b_{\mu'+1}, a_{\mu'+2}]$ at a cost of $|(b_{\mu'+1}, a_{\mu'+2}]|$. The next sequence of iterations will search the interval $(b_{\mu'+2}, a_{\mu'+3}]$ at a cost of $|(b_{\mu'+2}, a_{\mu'+3}]|$ and so on. Note that this is so even when $I(s_i)$ eventually fails—this will only happen when we are searching the interval $(b_\mu, a_1]$, and in this case as well, the time for searching is proportional to $|(b_\mu, a_1]|$. When $I(x_0)$ fails, the total time for searching for k is $|(b_1, a_1)| < |[a_1, b_1]|$ (where $b_1 = R(a_1) = R(x_0)$). In general we have, for each i, that $|[a_i, b_i]| > |(b_{\mu'+i}, a_{\mu'+i+1})|$. Therefore the time is always at most proportional to $|(b_1, a_2] \cup (b_2, a_3] \cup \ldots \cup (b_\mu, a_1]| \leq |C|$.

THEOREM 2.1. *There is an $O(n)$ algorithm which finds a maximum clique in an oriented graph given by a round enumeration with complete FO-information.* □

COROLLARY 2.4. *There is an $O(n)$ algorithm which finds a maximum clique in a proper circular arc graph given by a sorted simple family of circular arcs.* □

By combining the above theorem with the $O(m + n)$ algorithm of [2, 3], which finds a round enumeration with complete FO-information of a proper circular arc graph, we conclude the following fact.

COROLLARY 2.5. *There is an $O(m+n)$ algorithm which finds a maximum clique in a proper circular arc graph.* □

3. An application to q-coloring. We now apply the above algorithm to derive another $O(n)$ q-coloring algorithm for proper circular arc graphs for any fixed q. (An $O(n)$ algorithm was previously given in [16].) We again assume that D is a connected oriented graph of $\Delta(D) < n - 1$, given by a round enumeration $v_1, v_2, \ldots, v_n$ with complete FO-information. Let q be a fixed integer. We first find, using the algorithm of the previous section, a maximum clique C which is a μ-overlap clique for some odd integer μ. Let $\omega = |C|$.

If $\omega > q$, then D is not q-colorable. If $\omega = q$ and n is divisible by q then an easy q-coloring was proposed by Orlin, Bonuccelli, and Bovet [12]: D is q-colorable in the order $v_1, v_2, \ldots, v_n$ by consecutive "stretches" each consisting of colors $1, 2, \ldots, q$. The fact that q divides n means that these stretches fit together to color all n vertices (as $1, 2, \ldots, q, 1, 2, \ldots, q, \ldots, 1, 2, \ldots, q$). The fact that $\omega = q$ assures that this is a proper coloring: Indeed, any two vertices x, y of the same color have "circular distance" of at least $q + 1$ (that is, both $x + k = y$ and $y + k = x$ require $k \geq q$); if they were adjacent then $[x, y]$ or $[y, x]$ would be a clique of size $q + 1$.

A similar idea takes care of the case when $\omega \leq q - 1$. This allows us to use both stretches $1, 2, \ldots, q$ and $1, 2, \ldots, (q-1)$. If n is big enough, then we can easily fit these stretches of size $q-1$ and q together to cover the circular arrangement of all n vertices. Assume for instance that $n > (q - 1)^2$ and $n = P(q - 1) + S$ with $0 \leq S < q - 1$. Since $P \geq q - 1 > S$, we can color the first Sq vertices by stretches of size q, and color the remaining $(P - S)(q - 1)$ vertices by stretches of size $q - 1$. On the other hand, if $n \leq (q - 1)^2$, then any algorithm (for example, the one in [12]) can be used

to find a q-coloring in time $O(n)$, since $n = O(1)$.

We suppose from now on that $\omega = q$. Consider first the situation when μ is greater than one: let $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$. If a q-coloring exists, it must assign a different color to each vertex of C. Thus we may, without loss of generality, arbitrarily assign colors $1, 2, \ldots, q$ to the vertices of C. When can this coloring be completed to a q-coloring of the entire D? Note first that the vertices of each interval (b_i, a_{i+1}) are adjacent to each other and to all vertices of C with the possible exception of $[a_{i-\mu'}, b_{i-\mu'}]$. This is due to the fact that (b_i, a_{i+1}) is contained in both cliques $[a_{i-\mu'+1}, R(a_{i-\mu'+1})]$ and $[a_i, R(a_i)]$, which together cover all vertices of C except for $[a_{i-\mu'}, b_{i-\mu'}]$. Therefore, our coloring can be extended if and only if each vertex of (b_i, a_{i+1}) admits a different nonneighbor in $[a_{i-\mu'}, b_{i-\mu'}]$ for each $i = 1, 2, \ldots, \mu$. This happens if and only if there is, for each i, a matching of (b_i, a_{i+1}) into $[a_{i-\mu'}, b_{i-\mu'}]$ in the complement of the underlying graph of D. In fact, we now show that these matchings may be assumed to have a very special form: a *round matching* associates with each vertex x of (b_i, a_{i+1}) a nonneighbor $m(x)$ in $[a_{i-\mu'}, b_{i-\mu'}]$ in such a way that circular order is preserved; namely, for any other vertex $x' \in (b_i, a_{i+1})$, exactly one of $x', m(x')$ belongs to $(x, m(x))$. We claim that a matching (as above) exists if and only if a round matching exists. (Of course, it will be easier to test for the existence of round matching, cf. below.) Indeed, suppose that each vertex x of (b_i, a_{i+1}) is matched with a different nonneighbor $m(x)$ in $[a_{i-\mu'}, b_{i-\mu'}]$. A round matching can be obtained as follows: write $(b_i, a_{i+1}) = \{y_1, y_2, \ldots, y_s\}$ and $\{m(x) : x \in (b_i, a_{i+1})\} = \{z_1, z_2, \ldots, z_s\}$ where $y_1, y_2, \ldots, y_s, z_1, z_2, \ldots, z_s$ appear in the clockwise circular order. We prove that y_i, z_i are not adjacent. Suppose, without loss of generality, that y_i dominates z_i; then $[y_i, z_i]$ is a clique, and hence it was impossible to match $\{z_1, z_2, \ldots, z_i\}$ with $\{y_1, y_2, \ldots, y_{i-1}\}$, a contradiction. This proves the claim, and hence it suffices to look for round matchings. Round matchings can be found by an obvious greedy algorithm. Hence we can determine, in time $O(n)$, whether or not the coloring of C is extendable to a q-coloring of D.

Thus it only remains to explain what to do when $\omega = q$, C is a 1-overlap clique, and n is not divisible by q, say, $n = Qq + R$, with $0 < R < q$. It is known that D admits a q-coloring if and only if it admits a q-coloring with R "larger" classes of size $Q + 1$ and $q - R$ "smaller" classes of size Q [12]. The clique C must have one vertex in each of the color classes. Let Y denote the set of vertices of C which belong to the larger color classes. Then $D - Y$ has precisely Q vertices of each of the q colors. Thus D is q-colorable if and only if there exists a set Y of R vertices of C such that D has a q-coloring in which $D - Y$ has precisely Q vertices of each color. Since q is fixed, there are at most a constant number

$$\binom{q}{R} \leq \binom{q}{\lfloor \frac{q}{2} \rfloor}$$

of possible sets Y and we can afford to try all of them. Thus it will suffice to test if, for a fixed set Y of R vertices of C, there exists a q-coloring of D in which $D - Y$ has precisely Q vertices of each of the q colors. This is what we shall do below, by a greedy approach. To guide our choice of coloring we shall be referring to "blocks" of a vertex. For a fixed vertex v of D, the zeroth block has R consecutive vertices of D (in the clockwise order), starting with v; the first block consists of the next q vertices; the second block of the next q vertices after that and so on. (Thus the ith block of v is $[v + (i-1)q + R, v + iq + R - 1]$.)

Let $y_1, y_2, \ldots, y_R$ be the vertices of Y, and color y_i by color i. This uses colors $1, 2, \ldots, R$ in C. We color the remaining vertices of C by the remaining $q - R$ colors

in any order. It remains to color the vertices of $D - C$. The first step will be to color those vertices that will obtain colors $1, 2, \ldots, R$. We first color by 1 the first vertex v not dominated by y_1 (first in the linear order on $D - C$, induced by the clockwise order of D). It is easy to see that v is in the first block of y_1. Suppose we have just colored a vertex u in the jth block of y_i by color i. If $i < R$, and if s was the last vertex we colored by $i+1$ (possibly $s = y_{i+1}$), we find the first vertex w after u which lies in the jth block of y_{i+1} and is not dominated by s, and we color it by $i + 1$. If $i = R$ and $j < Q$ and if t was the last vertex colored by 1, then we find the first vertex z after u which lies in the $(j + 1)$st block of y_1 and is not dominated by t, and we color it by 1. It can be shown, cf. [7], that these vertices w, z always exist. However, it is not necessarily the case that this will produce a proper coloring. It is not difficult to see that if two vertices of color i are adjacent then they are y_i and the last vertex colored i by our procedure. This is easy to check as part of our procedure. If this happens (y_i is adjacent to the last vertex colored i) for some i, we try a different set Y. If each color class C_i is independent, then we easily extend the coloring to the remaining vertices by using the colors $R+1, \ldots, q$ as follows: Let D' be the subgraph of D consisting of all uncolored vertices, listed in the same circular order. Then D' has $Q(q - R)$ vertices, and it can be shown [7] that it has no 1-overlap clique of size $q - R + 1$. Thus it can be colored by stretches of colors $R + 1, \ldots, q$, as above.

It is not difficult to see that all procedures in our algorithm can be performed in time $O(n)$. Thus we have, for any fixed q, an $O(n)$ algorithm to test for q-colorability of an oriented graph given by its round enumeration with complete FO-information or, equivalently, of a proper circular arc graph given by a sorted simple family of circular arcs. As for maximum cliques, this implies, in conjunction with [2, 3], an $O(m + n)$ q-coloring algorithm for abstract proper circular arc graphs.

4. The maximum weight clique algorithm. In this section, we show how our maximum clique algorithm can be extended to weighted circular arc graphs. Thus assume that D is a round oriented graph with complete FO-information, and that each vertex v has an associated weight $w(v) > 0$. (Note that vertices of nonpositive weights may be removed from the graph without affecting the optimum solution.) The weight of a set S of vertices of D, denoted $w(S)$, is the sum of the weights of the vertices in S; if D' is a subgraph of D with vertex set S', then we let $w(D') = w(S')$. We shall seek a *maximum weight clique* of D, i.e., a clique C in D with the greatest value of $w(C)$. As in the unweighted case, we may assume that $\Delta(D) \leq n - 2$.

We extend the definitions by replacing all cardinalities by the corresponding weights. Thus a μ-overlap clique generated by $a_1, a_2, \ldots, a_\mu$ ($\mu = 2\mu' + 1 \geq 3$) is a clique $C = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_\mu, b_\mu]$ such that $a_1, b_1, a_2, b_2, \ldots, a_\mu, b_\mu$ are vertices of D in clockwise circular order and for each $i = 1, 2, \ldots, \mu$,

$$R(a_i) = b_{i+\mu'} \quad \text{and} \quad w([a_i, b_i]) > w((b_{i+\mu'}, a_{i+\mu'+1})).$$

It is still the case that there is at least one vertex in each interval $(b_{i+\mu'}, a_{i+\mu'+1})$; however, we note that this no longer implies that $a_i \neq b_i$ for each i. For instance in Fig. 3 we have $a_1 = b_1$ in the depicted maximum weight clique.

By proofs similar to those of §2, we easily show that some maximum weight clique must be a μ-overlap clique in the above sense, and that any generator x of a maximum weight 1-overlap clique is also a generator of some maximum weight clique which is a μ-overlap clique normalized with respect to x. (In the definition of a normalized clique we again replace the cardinalities by the corresponding weights.) It also remains true

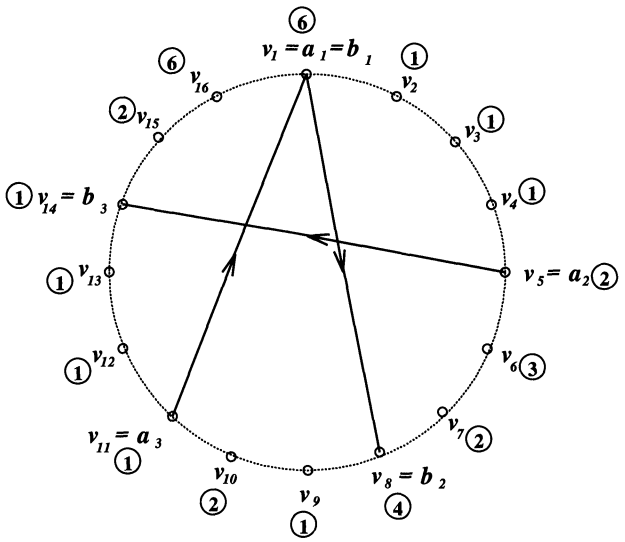


FIG. 3. *A maximum weight clique generated by* a_1, a_2, a_3 *(the circled numbers denote weights).*

that if the weight of a maximum weight 1-overlap clique K does not exceed half of the total weight of D, then K is a maximum weight clique.

We now explain how to adapt the algorithm: We again begin by finding a vertex x_0 which generates a maximum weight 1-overlap clique K and halt if $w(K) \leq \frac{w(D)}{2}$. Otherwise we proceed, as in the unweighted case, with basic iteration steps $I(x)$ using the starting vertex x (initially $x = x_0$).

Before we define the iterations in the weighted case, let us look more closely at the motivation for the iterations in the unweighted case. The iteration $I(x)$ seeks a suitable k such that $x + k \sim R(x) + k + 1$. We view the current x as a candidate for the next generator and if $x + k \rightarrow R(x) + k + 1$ then we conclude that $x + k$ is a better candidate. In fact, using $x+k$ in place of x will have the effect of replacing, in the clique being formed, the interval $[x, x+k)$ by the interval $(R(x), R(x)+k+1]$. Since we always have $|[x, x+k)| < |(R(x), R(x)+k+1]|$, this will result in a larger clique. On the other hand, if $R(x)+k+1 \rightarrow x+k$, then we accept x as a generator and go on to look for the next generator, starting with $R(x) + k + 1$ as the first candidate. This is a reasonable choice, in part because we always have $|[x, x + k]| > |(R(x), R(x) + k + 1)|$. In the weighted case we lack the "weight equivalents" of these two cardinality inequalities, and we must do something to assure their validity. Let

A denote the event $x + k \rightarrow R(x) + k'$ and $w([x, x + k)) < w((R(x), R(x) + k'])$,
B denote the event $R(x) + k' \rightarrow x + k$ and $w([x, x + k]) > w((R(x), R(x) + k'))$,
C denote the event $x + k = x_0$ and $k > 0$, and
D denote the event $R(x) + k' = x_0$.

The iteration $I(x)$ is defined as follows.

$k \leftarrow 0$ and $k' \leftarrow 1$
do until A or B or C or D
if $w([x, x + k]) \leq w((R(x), R(x) + k'])$ then $k \leftarrow k + 1$
else $k' \leftarrow k' + 1$
if C occurs then $s \leftarrow$ undefined and $g \leftarrow$ undefined, and $I(x)$ fails

else if D occurs then $g \leftarrow x$ and $s \leftarrow$ undefined, and $I(x)$ fails
else if A occurs then $s \leftarrow x + k$ and $g \leftarrow$ undefined
else $s \leftarrow R(x) + k + 1$ and $g \leftarrow x$

Finally, we present the entire algorithm.

ALGORITHM MAX-WEIGHT-CLIQ

find a vertex x_0 which generates a largest 1-overlap weighted clique
if $w([x_0, R(x_0)]) \leq \frac{w(D)}{2}$ then halt
$x \leftarrow x_0$
do until $I(x)$ fails
perform $I(x)$
$x \leftarrow s$

The correctness of the algorithm follows by arguments similar to those in the unweighted case. In particular, the six statements of Lemma 2.4 still hold, although their proofs need to be slightly modified. For instance, in the proof of statement 1, we now first prove that the values k and k' found by $I(a_i)$ are such that $a_i + k \in [a_i, b_i]$ and $R(a_i)+k' = b_{\mu'+i}+k' \in (b_{\mu'+i}, a_{\mu'+i+1}]$. This is certainly the case for the starting values of $k = 0, k' = 1$. If $a_i+k \in [a_i, b_i)$ and $R(a_i)+k' \in (b_{\mu'+i}, a_{\mu'+i+1})$ and if either k or k' is incremented by $I(a_i)$, then we still have $a_i + k \in [a_i, b_i]$ and $R(a_i) + k' \in (b_{\mu'+i}, a_{\mu'+i+1}]$. If $R(a_i) + k' = a_{\mu'+i+1}$ and $a_i + k \in [a_i, b_i]$, then $R(a_i) + k' \rightarrow a_i + k$, as $[R(a_i) + k', a_i + k] = [a_{\mu'+i+1}, a_i + k] \subseteq [a_{\mu'+i+1}, b_i] = [a_{\mu'+i+1}, R(a_{\mu'+i+1})]$. If $w([a_i, a_i + k]) > w((R(a_i), R(a_i) + k'))$, then the event B occurs and the current values of k and k' are the final values produced by $I(a_i)$; note that this includes the case when we also had $a_i + k = b_i$. On the other hand, if $w([a_i, a_i + k]) \leq w((b_{\mu'+i}, R(a_i) + k')) \leq w((b_{\mu'+i}, R(a_i) + k'])$ then $I(a_i)$ directs us to increment k, and we still have $a_i+k \in [a_i, b_i]$ and $R(a_i)+k' \in (b_{\mu'+i}, a_{\mu'+i+1}]$. Finally, if $a_i+k = b_i$ and $R(a_i)+k' \in (b_{\mu'+i}, a_{\mu'+i+1})$, then we cannot have $R(a_i)+k' \rightarrow a_i+k$, as otherwise the 1-overlap clique generated by $R(a_i) + k'$ would have weight greater than that generated by $a_{\mu'+i+1}$; this would contradict the weighted analogue of Corollary 2.1. Thus the event B does not occur. It is also clear that neither C nor D are occurring as $a_i + k = b_i$. Moreover, it is easy to see that the event A cannot occur either, because the 1-overlap clique generated by $b_i = a_i + k$ would contradict the weighted analogue of Corollary 2.1. Since $w([a_i, a_i + k]) = w([a_i, b_i]) > w((b_{\mu'+i}, a_{\mu'+i+1})) \geq w((b_{\mu'+i}, R(a_i)+k'])$, the iteration $I(a_i)$ increments k' and we still have $a_i+k \in [a_i, b_i]$ and $R(a_i) + k' \in (b_{\mu'+i}, a_{\mu'+i+1}]$. The rest of the proof of statement 1 proceeds in a manner analogous to the proof in the unweighted case, taking into account the modifications in the statement of the algorithm and replacing cardinalities by the corresponding weights.

The complexity of this algorithm is analyzed in a way similar to the unweighted algorithm. However, because of the modifications of the algorithm, we may visit all vertices of $[x, x + k] \cup (R(x), R(x) + k']$ when searching for k and k' in the iteration $I(x)$. Hence we may have to search through all the vertices of the graph; nevertheless, the time is still proportional to n.

Thus we have an $O(n)$ algorithm to find a maximum weight clique in a weighted oriented graph given by a round enumeration with complete FO-information, or equivalently, in a proper circular arc graph given by a sorted simple family of circular arc graphs. Therefore, using [2, 3], we obtain an $O(m+n)$ algorithm to find a maximum weight clique in a weighted proper circular arc graph.

To illustrate the algorithm consider the weighted oriented graph in Fig. 3. Using the complete FO-information it is still possible to find in one pass a largest 1-overlap clique, in this case the one generated by v_1 (which has weight 20). The algorithm starts by performing $I(v_1)$, which searches for k and k' such that A or B or C or D occurs. Since $v_1 + 0 = v_1$ is not adjacent to $R(v_1) + 1 = v_8 + 1 = v_9$ and $w([v_1, v_1]) = 6 > 1 = w((v_8, v_9])$, we increment k' and consider v_1 and $v_8 + 2 = v_{10}$. Since these vertices are not adjacent either and $w([v_1, v_1]) = 6 > 3 = w((v_8, v_{10}])$, we again increment k' and consider v_1 and v_{11}. Now $v_{11} \rightarrow v_1$ and $w([v_1, v_1]) = 6 > 3 = w((v_8, v_{11}))$; that is, B occurs. Thus $I(v_1)$ finds $k = 0$ and $k' = 3$ and produces the value $g = v_1$ and $s = v_{11}$. Next the algorithm performs $I(v_{11})$ which finds $k = 3$, $k' = 4$ and produces $g = v_{11}$, $s = v_5$. Finally, the iteration $I(v_5)$ finds $k = 3$ and $k' = 3$, and $I(v_5)$ fails, producing $g = v_5$ but no value of s. Hence the algorithm correctly finds the three generators v_1, v_{11}, v_5 of the maximum weight clique $[v_1, v_1] \cup [v_5, v_8] \cup [v_{11}, v_{14}]$ (of weight 21).

REFERENCES

[1] B. Bhattacharya and D. M. Kaller, *An $O(m+n \log n)$ Algorithm for the Maximum Clique Problem in Circular Arc Graphs*, Simon Fraser University report CSS/LCCR TR 94-26.

[2] X. Deng, P. Hell, and J. Huang, *Recognition and representation of proper circular arc graphs*, in Integer Programming and Combinatorial Optimization, Proc. 2nd IPCO conference, E. Balas, G. Cornuéjols, and R. Kannan, eds., Pittsburgh, PA, 1992, pp. 114–121.

[3] ——— *Linear time representation algorithms for proper circular arc graphs and proper interval graphs*, SIAM J. Comput., 25 (1996), pp. 390–403.

[4] F. Gavril, *Algorithms on circular-arc graphs*, Networks, 4 (1974), pp. 357–369.

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

[6] P. Hell, J. Bang-Jensen, and J. Huang, *Local tournaments and proper circular arc graphs*, in Algorithms, Lecture Notes in Computer Science 450, T. Asano, T. Ibaraki, H. Imai, and T. Nishizeki, eds., Springer-Verlag, Berlin, New York, 1990, pp. 101–109.

[7] P. Hell and J. Huang, *Linear Algorithms for c-Coloring and for Finding Maximum Cliques in Proper Circular Arc Graphs*, Dept. of Math. and Compt. Sci., Odense University, Odense, Denmark, 1993, Preprint 9.

[8] ———, *Two remarks on circular arc graphs*, Graphs Combin., to appear.

[9] J. Huang, *Tournament-Like Oriented Graphs*, Ph.D. thesis, Simon Fraser University, Burnaby, BC, 1992.

[10] G. Manacher and T. Mankus, *Finding normal forms, cliques, hamiltonian paths, and hamiltonian circuits in a set of proper circular arcs in optimum time and space*, in Proc. 26th Allerton Conference on Communication, Control, and Computing, Allerton, IL, 1988, pp. 832–841.

[11] ———, *Finding a maximum clique in a set of proper circular arcs in time $O(n)$*, manuscript.

[12] J. B. Orlin, M. A. Bonuccelli, and D. P. Bovet, *An $O(n^2)$ algorithm for coloring proper circular arc graphs*, SIAM J. Alg. Disc. Meth., 2 (1981), pp. 88–93.

[13] W. K. Shih and W. L. Hsu, *An $O(n \log n + m \log \log n)$ maximum weight clique algorithm for circular arc graphs*, Information Processing Letters, 31 (1989), pp. 129–134.

[14] ———, *An $O(n^{1.5})$ algorithm to colour proper circular arcs*, Discrete Appl. Math., 25 (1989), pp. 321–323.

[15] K. E. Stouffers, *Scheduling of traffic lights—a new approach*, Transportation Res., 2 (1968), pp. 199–234.

[16] A. Teng and A. Tucker, *An $O(qn)$ algorithm to q-color a proper family of circular arcs*, Discrete Math., 55 (1985), pp. 233–243.

[17] A. Tucker, *Colouring a family of circular arcs*, SIAM J. Appl. Math., 29 (1975), pp. 493–502.

[18] ———, *Matrix characterization of circular arc graphs*, Pacific J. Math, 39 (1971), pp. 535–545.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 290–300, May 1996

011

THE ULTIMATE CATEGORICAL INDEPENDENCE RATIO OF A GRAPH*

JASON I. BROWN†, RICHARD J. NOWAKOWSKI†, AND DOUGLAS RALL‡

Abstract. Let $\beta(G)$ denote the independence number of a graph G. We introduce $A(G) = \lim_{k\to\infty} \beta(G^k)/|V(G)|^k$, where the categorical graph product is used. This limit, surprisingly, lies in the range $(0, 1/2] \cup \{1\}$. We can show that this limit can take any such rational number, but is there any G for which $A(G)$ is irrational? A useful technique for bounding $A(G)$ is to consider special spanning subgraphs. These bounds allow us to efficiently compute $A(G)$ for many G. We give a condition which if true for G shows that $A(G) > \beta(G)/|V(G)|$. This brings up the question: for which G does $A(G) = \beta(G)/|V(G)|$? This happens if G is a Cayley graph of an Abelian group or if G is a connected graph that has an automorphism which has a single orbit.

Key words. independence number, ultimate categorical independence ratio, decomposition, self-universal

AMS subject classifications. 05C70, 05C99

1. Introduction.

Graph products have been used to find the *essential* value of a graph parameter (such as independence number or chromatic number) of a graph G by "multiplying" G by itself k times and examining the growth of the parameter on G^k. For example, the *Shannon capacity* [12, 15, 16] of a graph G is defined by

$$\Theta(G) = \lim_{k\to\infty} \beta(\otimes_{i=1}^{k} G)^{1/k},$$

where $\otimes$ is the strong product of graphs and $\beta(H)$ denotes the independence number of graph H (i.e., the maximum cardinality of an independent set of vertices of H). The Shannon capacity of a graph arose from a problem of transmission of words over a noisy line but has a number of other applications (see [12]).

Another such concept is the *ultimate chromatic number* of a graph G; that is,

$$\chi_u(G) = \lim_{k\to\infty} (\chi(\bullet_{i=1}^{k} G))^{1/k}.$$

(Here $\bullet$ denotes the lexicographic product and $\chi(H)$ the chromatic number of H.) This was introduced by Hilton, Rado, and Scott [11] (see also [5]) and is related to the problem of assigning radio frequencies to vehicles operating in zones (see Gilbert [6] and Roberts [13, 14]). The determination of both the Shannon capacity for some graphs and the ultimate chromatic number for all graphs can be solved using linear programming techniques. (See [15] for the former and [9] for the latter.)

In contrast to the Shannon capacity, one can investigate the parameter of the independence number by looking at the ratio of this parameter to the total number of vertices in the graph; if $|V(G)| = n$, the *ultimate independence ratio* of G is

$$I(G) = \lim_{k\to\infty} \beta(\Box_{i=1}^{k} G)/n^k,$$

where $\Box$ is the Cartesian product. This was introduced in [10]. We consider an analogous (but significantly different) concept.

* Received by the editors November 9, 1994; accepted for publication (in revised form) June 30, 1995.

† Department of Mathematics, Statistics, and Computing Science, Dalhousie University, Halifax, Nova Scotia B3H 4H6 Canada. The research of these authors was partially supported by an NSERC grant.

‡ Department of Mathematics, Furman University, Greenville, SC 29613.

Let $G = (V(G), E(G))$ be a graph. We will assume that graphs are finite and simple. We write $a \sim b$ if a is adjacent to but not equal to b, and $a \perp b$ if a is neither adjacent nor equal to b. The *categorical* product $G \times H$ of G and H has the vertex set $V(G) \times V(H)$ and $(a, x) \sim (b, y)$ if both $a \sim b$ and $x \sim y$. As the categorical product is the only product we shall consider henceforth, for the rest of the paper we use $\times_{i=1}^{k} G$ and G^k interchangeably. The parameter we consider is the *ultimate categorical independence ratio* of a graph G which is defined as

$$A(G) = \lim_{k \to \infty} \beta(\times_{i=1}^{k} G)/n^k,$$

where $|V(G)| = n$. We show that this limit exists in the next section.

The next two sections deal with upper and lower bounds, respectively. The fourth section investigates disjoint unions of graphs, and we find classes of graphs for which the sequence $\langle \beta(G^k)/|V(G^k)| \rangle$ is not constant; i.e., $A(G) > \beta(G)/|V(G)|$. In the fifth section we look at graphs for which $A(G) = \beta(G)/|V(G)|$ and in the last section we pose several problems.

We follow standard graph theoretic terminology (cf. [1, 2, 8]), but we make explicit note of a few definitions. We abbreviate the disjoint union of m copies of graph G by mG. Let $S \subseteq V(G)$ and $v \in V(G)$. Then $\langle S \rangle$ denotes the induced subgraph on the vertices of S. The *neighborhood* of $v \in V(G)$ is $N(v) = \{y \mid y \sim v\}$ and $N(S) = \cup_{v \in S} N(v)$; $N[v] = N(v) \cup \{v\}$ is the *closed* neighborhood of v and $N[S] = \cup_{v \in S} N[v]$. The set S is called *independent* if $\langle S \rangle$ contains no edges and, as mentioned above, $\beta(G)$ denotes the maximum cardinality of an independent set of G. The *chromatic* number of G is denoted by $\chi(G)$. We remark that $G \times H$ is connected if and only if both G and H are connected and at least one is 3-chromatic [17]. It is easy to see as well that $G \times H \cong H \times G$, $G \times (H \cup K) = (G \times H) \cup (G \times K)$, and $\chi(G \times H) \leq \min\{\chi(G), \chi(H)\}$. Whether $\chi(G \times H) = \min\{\chi(G), \chi(H)\}$ is Hedetniemi's conjecture, an open problem that has attracted (and frustrated!) a number of mathematicians (see, for example, [4, 3]).

2. Upper bounds for $A(G)$. The first fact needed is that the ultimate categorical independence ratio really exists for any graph. We will make use of the following elementary but important fact.

LEMMA 2.1. *Let G and H be graphs. Then $\beta(G \times H) \geq \max\{\beta(G)|V(H)|, |V(G)| \beta(H)\}$.*

Proof. If I is an independent set of H then $\cup_{a \in I} G \times \{a\}$ is an independent set of $G \times H$ and thus $\beta(G \times H) \geq \beta(H)|V(G)|$. The second part follows similarly since the product is commutative. □

From this lemma it follows that $\beta(G^k)/n^k \geq n\beta(G^{k-1})/n^k = \beta(G^{k-1})/n^{k-1}$. Therefore the sequence $\beta(G^k)/n^k$ is nondecreasing and is bounded above by 1 and so the ultimate categorical independence ratio exists. We contrast this with the ultimate independence ratio of Hell, Yu, and Zhou [10], where $I(G)$ is the limit of the *nonincreasing* sequence $\beta(\square_{i=1}^{k} G)/n^k)$.

The following observation is extremely useful and forms the basic idea for this and the next section.

LEMMA 2.2. *If G is a spanning subgraph of H then $A(G) \geq A(H)$.*

Proof. For all k, G^k is a spanning subgraph of H^k. Thus $\beta(G^k) \geq \beta(H^k)$ and therefore $\beta(G^k)/n^k \geq \beta(H^k)/n^k$. □

The previous result, along with the next, is quite helpful in bounding the ultimate categorical independence ratio of a graph.

THEOREM 2.3. *If G is a regular graph of degree $r > 0$ then $A(G) \leq 1/2$.*

Proof. Let $I \subseteq V(G)$ be an independent set with $|I| = \beta(G)$ and let $C = V(G) - I$. Each vertex in G has degree r so summing the degrees of the vertices in C we have $(n-\beta(G))r$. Some edges have been counted twice, but the edges between I and C have only been counted once. Therefore, since I is an independent set, $r\beta(G) \leq (n-\beta(G))r$; i.e., $\beta(G)/n \leq 1/2$. Now G^k is a regular graph of degree r^k so $\beta(G^k)/n^k \leq 1/2$ and consequently $A(G) \leq 1/2$. □

Theorem 2.3 and Lemma 2.2 give the following corollary.

COROLLARY 2.4. *If H has a regular spanning subgraph of degree at least one then*

$$\beta(H)/n \leq A(H) \leq \frac{1}{2}.$$

This result shows that $A(C_{2n}) = A(P_{2n}) = 1/2$, the latter because P_{2n} has a perfect matching as a spanning subgraph. Moreover, if G has a Hamiltonian cycle then $A(G) \leq 1/2$. However, if G has a Hamiltonian path then it is possible for $A(G)$ to be greater than $1/2$. For example, $A(P_{2n+1}) \geq (n+1)/(2n+1) > 1/2$. Indeed, this raises the question: *What is $A(P_{2n+1})$?* We answer this question later.

General upper bounds for the ultimate categorical independence ratio are few and far between. One technique that appears fruitful involves partitioning the vertex set into subgraphs. In the following, the phrase *K decomposes into $L_1, \ldots, L_l$* means that $L_1 \cup \cdots \cup L_l$ is a spanning subgraph of K. (We denote this as well by $K \Rightarrow L_1 \cup \cdots \cup L_l$.)

LEMMA 2.5. *If G is a graph of order n and $G \times H \Rightarrow nH$ then $\beta(G \times H) = \beta(H)n$. Moreover, if also $H \cong G$ then $A(G) = \beta(G)/n$.*

Proof. From Lemma 2.1 we have that $\beta(G \times H) \geq \beta(H)|V(G)| \geq \beta(H)n$ for any graphs G and H.

Suppose that $G \times H$ can be partitioned so that the subgraphs in each partition are all isomorphic to H. Thus any independent set of $G \times H$ intersects each part in no more than $\beta(H)$ many vertices. Therefore $\beta(G \times H) \leq \beta(H)n$. Consequently, $\beta(G \times H) = \beta(H)n$.

If a graph G is of order n and G^2 decomposes into nG, then inductively $G^k \Rightarrow n^{k-1}G$ and $\beta(G^k)/n^k \leq (n^{k-1}\beta(G))/n^k = \beta(G)/n$. Thus $A(G) = \beta(G)/n$. □

This result shows that $A(K_n) = \frac{1}{n}$. This follows since K_n^2 can be decomposed into nK_n: if the vertices of K_n are $\mathbb{Z}_n$, and X_i is the subgraph of K_n^2 induced by $\{(j, j+i) : j \in \mathbb{Z}_n\}$, then K_n^2 decomposes into $X_0, \ldots, X_{n-1}$; i.e., $K_n^2 \Rightarrow nK_n$.

Decompositions can be utilized to prove the following general upper bound.

THEOREM 2.6. *Suppose $G \Rightarrow H \cup K_{m_1} \cup \cdots \cup K_{m_p}$ where $|V(H)| = n$ and $\chi(H) \leq m_1 \leq \cdots \leq m_p$. Then $A(G) \leq A(H)$.*

Proof. Suppose $H \cup K_{m_1} \cup \cdots \cup K_{m_p}$ is a spanning subgraph of G, where $\chi(H) \leq m_1 \leq \cdots \leq m_p$. Then G^k decomposes into

$$\bigcup_{i=0}^{k} \binom{k}{i} H^i \times (K_{m_1} \cup \cdots \cup K_{m_p})^{k-i}.$$

We observe that if $\chi(H) \leq r$ then $H \times K_r$ has a decomposition into r copies of H, as follows. Let $\chi(H) = j$ and the color classes of H be $C_1, C_2, \ldots, C_j$. Let $V(K_r) = \{a_1, a_2, \ldots, a_r\}$. The ith copy of H would be $(C_1 \times \{a_i\}) \cup (C_2 \times \{a_{i+1}\}) \cup \cdots \cup (C_j \times \{a_{i+j-1}\})$ for $i = 1, 2, \ldots, r$ where the subscripts are taken modulo r.

Since $\chi(H^i) \leq \chi(H)$ for all i, using the previous remark we have (for $i < k$)

$$H^i \times (K_{m_1} \cup \cdots \cup K_{m_p})^{k-i}$$

$$
\begin{aligned}
&= H^i \times (K_{m_1} \cup \cdots \cup K_{m_p}) \times (K_{m_1} \cup \cdots \cup K_{m_p})^{k-i-1} \\
&= (H^i \times K_{m_1} \cup H^i \times K_{m_2} \cup \cdots \cup H^i \times K_{m_p}) \times (K_{m_1} \cup \cdots \cup K_{m_p})^{k-i-1} \\
&\Rightarrow (m_1 + \cdots + m_p) H^i \times (K_{m_1} \cup \cdots \cup K_{m_p})^{k-i-1},
\end{aligned}
$$

and so by induction,

$$
H^i \times (K_{m_1} \cup \cdots \cup K_{m_p})^{k-i} \Rightarrow (m_1 + \cdots + m_p)^{k-i} H^i
$$

for any $i \in \{0, \ldots, k\}$. (We interpret the 0th power of a graph to be K_1.)

Thus G^k has a decomposition into

$$
\left(\bigcup_{i>0} \binom{k}{i} (m_1 + \cdots + m_p)^{k-i} H^i \right) \quad \cup \quad (K_{m_1} \cup \cdots \cup K_{m_p})^k.
$$

Now as $\beta((K_{m_1} \cup \cdots \cup K_{m_p})^k) \leq (m_1 + \cdots + m_p)^k$ we have

$$
\beta(G^k) \leq \left(\sum_{i>0} \binom{k}{i} (m_1 + \cdots + m_p)^{k-i} \beta(H^i) \right) + (m_1 + \cdots + m_p)^k.
$$

Since $\beta(H^i) \leq A(H) n^i$,

$$
\begin{aligned}
\beta(G^k) &\leq \left(\sum_{i>0} \binom{k}{i} (m_1 + \cdots + m_p)^{k-i} A(H) n^i \right) + (m_1 + \cdots + m_p)^k \\
&= A(H) \left((n + m_1 + \cdots + m_p)^k - (m_1 + \cdots + m_p)^k \right) \\
&\quad + (m_1 + \cdots + m_p)^k.
\end{aligned}
$$

Dividing through by $(n + m_1 + \cdots + m_p)^k$ yields

$$
\frac{\beta(G^k)}{(n + m_1 + \cdots + m_p)^k} \leq A(H) \left(1 - \frac{(m_1 + \cdots + m_p)^k}{(n + m_1 + \cdots + m_p)^k} \right) + \frac{(m_1 + \cdots + m_p)^k}{(n + m_1 + \cdots + m_p)^k}.
$$

Letting $k \to \infty$ we get $A(G) \leq A(H)$. □

Suppose we have a coloring π of $\overline{G}$, the complement of G, and $c_0 \leq c_1 \leq \cdots \leq c_p$ are the sizes of the color classes. Then we have a decomposition of the original graph G into cliques:

$$
G \Rightarrow K_{c_0} \cup K_{c_1} \cup \cdots \cup K_{c_p}.
$$

Taking $H = K_{c_0}$ and applying the previous theorem, we deduce that $A(G) \leq 1/c_0$. Of course the bound is improved when c_0 is as large as possible. We summarize as follows.

COROLLARY 2.7. *For each coloring π of $\overline{G}$, let $c(\pi)$ denote the minimum cardinality of a color class in π, and let c denote the maximum of $c(\pi)$ over all colorings. Then $A(G) \leq \frac{1}{c}$.*

3. Lower bounds for $A(G)$. If one takes an independent set I in G, then the subset $I \times G^{k-1}$ is an independent set in G^k (this is inherent in Lemma 2.1), and it follows that $A(G) \geq \beta(G)/|V(G)|$. There are graphs (such as the complete graphs) for which equality holds (we will have more to say about such graphs later). The following lemma is often useful in showing when independent sets larger than $\beta(G)|V(G)|$ exist in the product of G and H.

LEMMA 3.1. *Let G and H be graphs and let I be an independent set of G where $|G - N[I]| = k$. Then $\beta(G \times H) \geq k\beta(H) + |I||V(H)|$.*

Proof. Let P be a maximum-sized independent set of H. Let $Q = (V(G) - N[I]) \times P$ and $R = I \times V(H)$. Then $Q \cup R$ is independent in $G \times H$ and $|Q \cup R| = k\beta(H) + |I||V(H)|$. □

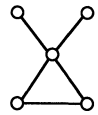


FIG. 1.

If G is the graph in Figure 1, then $\beta(G)/5 = 3/5$, but $\beta(G^2)/25 = 16/25 > 3/5$. Moreover, let $I = \{a, c\}$ and $H = G^k$. Then $|G - N[I]| = 2$, and applying the lemma to $G \times H$, we have $A(G) \geq 2/3$.

Yet for the graph G in Figure 1, what is $A(G)$? The next result provides the answer as well as giving $A(P_{2n+1})$ by proving the surprising fact that if the independence number of any graph is more than half the number of vertices, then the ultimate categorical independence ratio is 1.

THEOREM 3.2. *Let G be a graph with n vertices. If $\beta(G)/n > 1/2$ then $A(G) = 1$.*

Proof. Let I be an independent set of G with $|I| = \beta(G) > n/2$. Note that $\beta(G) > n - \beta(G)$.

In G^k denote the factors as G_i, $i = 1, 2, \ldots, k$ with I_i as the copy of I in the ith factor. Now form the set

$$J = \bigcup_P (\times_{i \in P} I_i) \times \left(\times_{j \notin P} (V(G_j) - I_j) \right)$$

where the union is taken over all $P \subseteq \{1, 2, \ldots, k\}$ with $|P| > k/2$. For any two vertices $\mathbf{x} = (x_1, \ldots, x_k), \mathbf{y} = (y_1, \ldots, y_k)$ in J, as each has more than half its coordinates in I, there is an i such that $x_i,\ y_i \in I$, and hence $\mathbf{x}$ and $\mathbf{y}$ are not adjacent. Thus J is an independent set.

Counting gives us the following:

$$|J| = \sum_{j > k/2} \binom{k}{j} \beta(G)^j (n - \beta(G))^{k-j}.$$

Completing the summation and taking it away again gives

$$|J| = \sum_{j=0}^{k} \binom{k}{j} \beta(G)^j (n - \beta(G))^{k-j} - \sum_{j \leq k/2} \binom{k}{j} \beta(G)^j (n - \beta(G))^{k-j},$$

and thus

$$|J| = n^k - \sum_{j \leq k/2} \binom{k}{j} (\beta(G))^j (n - \beta(G))^{k-j}.$$

Now since $\beta(G) > (n - \beta(G))$, $f(x) = \left(\frac{\beta(G)}{n - \beta(G)}\right)^x$ is an increasing function of x. It follows that for $j \leq k/2$,

$$\beta(G)^j (n - \beta(G))^{k-j} \leq \beta(G)^{k/2} (n - \beta(G))^{k/2}.$$

Therefore we get

$$|J| = n^k - \sum_{j \le k/2} \binom{k}{j} \beta(G)^{k/2} (n - \beta(G))^{k/2}$$
$$\ge n^k - 2^k \beta(G)^{k/2} (n - \beta(G))^{k/2}.$$

Dividing by n^k gives

$$|J|/n^k \ge 1 - \left(\frac{4\beta(G)(n - \beta(G))}{n^2} \right)^{k/2}.$$

Now as $4\beta(G)(n - \beta(G)) < n^2$, the right side tends to 1 as k goes to infinity, and since $A(G) \ge |J|/n^k$ for all k, we are done. □

We have seen that $A(C_{2n+1}) = n/(2n+1) < 1/2$. Thus the addition of a single edge to a graph (here to P_{2n+1}), while changing the ratio of independence number to order by an arbitrarily small amount, may greatly affect the ultimate categorical independence ratio.

We now consider a lower bound which is a companion result to Theorem 2.6.

THEOREM 3.3. *Let I be an independent subset of G. Then $A(G) \ge |I|/|N[I]|$*

Proof. Let $H = N[I]$ and $F = G - H$ and put $m = |V(F)|$ and $n = |V(H)|$. Let $J \subseteq G^k - F^k = (H \cup F)^k - F^k$ (that is, for $i = 1, 2, \ldots, k$ choose i of the coordinates and in these coordinates the entries will be taken from H and in the others the entries will be taken from F), with the extra condition that in the first coordinate in which the entries are taken from H they will be restricted to vertices from I. Then J is an independent set of G. This follows since if $\mathbf{x}, \mathbf{y} \in J$ then let i and j, respectively, be the least indices such that $x_i \in I$ and $y_j \in I$. If $i = j$ then $x_i = y_i$ or $x_i \perp y_i$; if $i < j$ then $y_i \in F$ thus $x_i \perp y_i$ and so $\mathbf{x} \perp \mathbf{y}$. In any event, $\mathbf{x}$ and $\mathbf{y}$ are nonadjacent. Now

$$|J| = \sum_{i>0} \binom{k}{i} |I| n^{i-1} m^{k-i} = \frac{|I|}{n} \sum_{i>0} \binom{k}{i} n^i m^{k-i} = \frac{|I|}{n} \left((n+m)^k - m^k \right).$$

Therefore,

$$\beta(G^k) \ge \frac{|I|}{n} \left((n+m)^k - m^k \right),$$

and dividing through by $(n+m)^k$ we get

$$\frac{\beta(G^k)}{(n+m)^k} \ge \frac{|I|}{n} \left(1 - \left(\frac{m}{n+m} \right)^k \right).$$

Thus as $k \to \infty$ we finally obtain $A(G) \ge |I|/n$. □

This theorem shows in particular that if G has an isolated vertex x, then $A(G) = 1$, since if $I = \{x\}$, then $A(G) \ge 1/1$.

The previous two theorems can be combined to yield the following corollary.

COROLLARY 3.4. *If G has an independent set I such that $|I| > |N(I)|$ then $A(G) = 1$.*

Proof. By the previous theorem $A(G) \ge |I|/|N[I]|$, but this latter term is greater than $1/2$ and thus by Theorem 3.2, $A(G) = 1$. □

We know from Theorem 3.2 that if $\beta(G)/|V(G)| > 1/2$ then $A(G) = 1$. What can we say if $\beta(G)/|V(G)| = 1/2$?

COROLLARY 3.5. *Suppose that G is a graph of order n with $\beta(G) = n/2$. Then $A(G) = 1/2$ if G has a perfect matching, and $A(G) = 1$ otherwise.*

Proof. If G has a perfect matching then G has a 1-regular spanning subgraph and so $A(G) \leq 1/2$. In fact, $A(G) = 1/2$ in this case as $A(G) \geq \beta(G)/n = 1/2$.

Now assume that G has no perfect matching. Let J be an independent set of size $n/2$ in G. Then there is no matching of J into $G - J$, and hence by Hall's theorem, there is a subset $I \subseteq J$ such that $|N(I)| < |I|$. Set $H = N[I]$. Then applying Theorem 3.3,

$$A(G) \geq \frac{|I|}{|I| + |N(I)|} > \frac{1}{2},$$

and hence by Theorem 3.2, $A(G) = 1$. □

We remark that there are graphs G for which $A(G) = 1/2$ and yet G has no perfect matching. For example, we see in the next section that for any $n \geq 1$, $A(K_2 \cup K_{2n+1}) = 1/2$ while clearly neither this graph nor any power has a perfect matching.

We can now determine quickly what the ultimate categorical independence ratio is for any bipartite graph G. Let G have order n. We can find a 2-coloring of G in polynomial time. Let the color classes be C_1 and C_2. Clearly

$$A(G) = \frac{\beta(G)}{n} \geq \frac{1}{2}.$$

If $|C_1| \neq |C_2|$, then $\beta(G) > n/2$, and hence $A(G) = 1$. Assume now that $|C_1| = |C_2|$. If there is a perfect matching in G (and this can be determined in polynomial time), then $A(G) = 1/2$. Otherwise, by Corollary 3.5, $A(G) = 1$. Thus we have the following corollary.

COROLLARY 3.6. *If G is biparite, then $A(G)$ can be determined in polynomial time.*

Again, we contrast this result with that for the ultimate independence ratio, where it is known [7] that $I(G) = 1/2$ for *any* bipartite graph G.

4. The ultimate categorical independence ratio for disjoint unions. It is of interest to see how the ultimate categorical independence ratio can change under graph operations. The independence number of the union of graphs changes in an obvious way, namely the sum of the independence numbers of its constituent parts. It is not so clear as to how the ultimate categorical independence ratio changes under disjoint union. For a graph G, it is clear that $(G \cup G)^k = 2^k G^k$. It follows that $A(G \cup G) = A(2G) = A(G)$. The next result shows that $A(G \cup H)$ is at least the maximum of $A(G)$ and $A(H)$.

THEOREM 4.1. *If G and H are any graphs, then $A(G \cup H) \geq \max\{A(G), A(H)\}$.*

Proof. Let $n_G = |V(G)|$ and $n_H = |V(H)|$. We show first that $A(G \cup H) \geq A(G)$. The other inequality follows similarly.

Note that

$$(G \cup H)^k = \bigcup_{i=0}^{k} \binom{k}{i} G^i \times H^{k-i},$$

and hence

$$\beta((G \cup H)^k) = \sum_{i=0}^{k} \binom{k}{i} \beta(G^i \times H^{k-i}).$$

By dropping the $i = 0$ term, we obtain

$$\begin{aligned}\frac{\beta((G \cup H)^k)}{(n_G + n_H)^k} &\geq \sum_{i=1}^{k} \frac{\binom{k}{i}\beta(G^i \times H^{k-i})}{(n_G + n_H)^k} \\ &\geq \sum_{i=1}^{k} \frac{\binom{k}{i}\beta(G^i)n_H^{k-i}}{(n_G + n_H)^k} \qquad \text{[Lemma 2.1]} \\ &= \sum_{i=1}^{k} \frac{\binom{k}{i}n_G^i n_H^{k-i}}{(n_G + n_H)^k} \cdot \frac{\beta(G^i)}{n_G^i}. \qquad (*)\end{aligned}$$

There are two cases. First, suppose $A(G) = \beta(G)/n_G$. Then $A(G) = \beta(G^i)/n_G^i$ for all i. In this case

$$\begin{aligned}\frac{\beta((G \cup H)^k)}{(n_G + n_H)^k} &\geq \sum_{i=1}^{k} \frac{\binom{k}{i}n_G^i n_H^{k-i}}{(n_G + n_H)^k} \cdot \frac{\beta(G^i)}{n_G^i} \\ &= A(G) \sum_{i=1}^{k} \frac{\binom{k}{i}n_G^i n_H^{k-i}}{(n_G + n_H)^k} \\ &= A(G)\left(1 - \frac{n_H^k}{(n_G + n_H)^k}\right).\end{aligned}$$

Thus

$$A(G \cup H) = \lim_{k\to\infty} \frac{\beta((G \cup H)^k)}{(n_G + n_H)^k} \geq \lim_{k\to\infty} A(G)\left(1 - \frac{n_H^k}{(n_G + n_H)^k}\right) = A(G).$$

Now we may assume that $A(G) > \beta(G)/n_G$. We choose $\varepsilon > 0$ and we will show that $A(G \cup H) \geq A(G) - \varepsilon$. Let

$$\gamma = A(G) - \frac{\varepsilon}{2}.$$

Since $\beta(G^i)/n_G^i$ is a nondecreasing sequence we can fix $J \geq 1$ so that

$$\frac{\beta(G^j)}{n_G^j} > \gamma \text{ for all } j \geq J \qquad \text{and} \qquad \frac{\beta(G^i)}{n_G^i} \leq \gamma \text{ for all } i < J.$$

Let

$$\psi = \frac{\max\{n_G, n_H\}}{n_G + n_H}$$

and

$$\mu = \begin{cases} \gamma - \beta(G)/n_G & \text{if } J > 1, \\ 0 & \text{if } J = 1. \end{cases}$$

Note that for $J > 1$, $\mu = \max\{\gamma - \beta(G^i)/n_G^i \mid i < J\}$. Also, $\psi \in (0,1)$ and $\mu \geq 0$ constants. Thus, taking $k > 2J$, from $(*)$ we get

$$\begin{aligned}\frac{\beta((G \cup H)^k)}{(n_G + n_H)^k} &\geq \sum_{i=1}^{k} \frac{\binom{k}{i}n_G^i n_H^{k-i}}{(n_G + n_H)^k} \cdot \gamma - \sum_{i=1}^{J-1} \frac{\binom{k}{i}n_G^i n_H^{k-i}}{(n_G + n_H)^k}\left(\gamma - \frac{\beta(G^i)}{n_G^i}\right) \\ &= \gamma \sum_{i=0}^{k} \frac{\binom{k}{i}n_G^i n_H^{k-i}}{(n_G + n_H)^k} - \gamma \frac{n_H^k}{(n_G + n_H)^k} - \sum_{i=1}^{J-1} \frac{\binom{k}{i}n_G^i n_H^{k-i}}{(n_G + n_H)^k}\left(\gamma - \frac{\beta(G^i)}{n_G^i}\right) \\ &\geq \gamma - \gamma \frac{n_H^k}{(n_G + n_H)^k} - (J-1)\binom{k}{J-1}\psi^k \mu.\end{aligned}$$

Therefore,

$$\frac{\beta((G \cup H)^k)}{(n_G + n_H)^k} \geq \left(A(G) - \frac{\varepsilon}{2}\right)\left(1 - \frac{n_H^k}{(n_G + n_H)^k}\right) - Ck^{J-1}\psi^k,$$

where $C = \frac{(J-1)\mu}{(J-1)!}$ is a nonnegative constant.

This shows that

$$\begin{aligned} A(G \cup H) &= \lim_{k\to\infty} \frac{\beta((G \cup H)^k)}{(n_G + n_H)^k} \\ &\geq \lim_{k\to\infty} \left(\left(A(G) - \frac{\varepsilon}{2}\right)\left(1 - \frac{n_H^k}{(n_G + n_H)^k}\right) - Ck^{J-1}\psi^k\right) \\ &\geq A(G) - \varepsilon \end{aligned}$$

for any $\varepsilon > 0$ and hence $A(G \cup H) \geq A(G)$.

In both cases we have that $A(G \cup H) \geq A(G)$. Similarly, we also have that $A(G \cup H) \geq A(H)$, and, therefore, it follows that $A(G \cup H) \geq \max\{A(G), A(H)\}$. □

As a corollary to this theorem and Corollary 2.7, we can determine the ultimate categorical independence ratio for the disjoint union of complete graphs.

COROLLARY 4.2.

$$A\left(\bigcup_{i=1}^{l} K_{m_i}\right) = \frac{1}{\min\{m_i : i = 1, \ldots, l\}}.$$

This corollary yields infinitely many graphs G for which $A(G)$ is not 1, not $1/2$, nor $\beta(G)/|V(G)|$; in fact, the disjoint union of complete graphs of order at least 2 where not all the cliques have the same size are such examples.

Finally, we also derive that there are graphs G with arbitrarily small $\beta(G)/|V(G)|$ for which $A(G)$ climbs up to 1. For example,

$$\frac{\beta(K_{1,2} \cup K_n)}{|V(K_{1,2} \cup K_n)|} = \frac{3}{n+3},$$

while $A(K_{1,2} \cup K_n) = 1$ as $A(K_{1,2} \cup K_n) \geq A(K_{1,2}) = 1$.

5. Universal graphs and the distribution of $A(G)$. Lemma 3.1 leads naturally to the next definition. A graph G is called *categorical-universal* if $\beta(G \times H) = \max\{\beta(G)|V(H)|, \beta(H)|V(G)|\}$ for all graphs H. A related notion of universal graphs was originally introduced in the Shannon capacity.

Of course if $G = \overline{K_n}$, then G is categorical-universal since $\overline{K_n} \times H$ contains no edges and thus $\beta(\overline{K_n} \times H) = n|V(H)| = \max\{\beta(\overline{K_n})|V(H)|, \beta(H)|V(\overline{K_n})|\}$. In fact, it can be shown that these are the *only* categorical-universal graphs.

A more restricted concept than a categorical-universal graph is the following: graph G is *self-universal* if $A(G) = \beta(G)/|V(G)|$. From Lemma 2.5 it follows that if a graph G is of order n and G^2 decomposes into nG, then G is self-universal. We have also seen that regular bipartite graphs and cliques are in the class. The next result greatly increases the known self-universal graphs by showing that it includes all Cayley graphs on an Abelian group. (A similar result holds for ultimate independence ratios [10].)

THEOREM 5.1. *If G is the Cayley graph of an Abelian group then G is self-universal.*

Proof. Let S be the generating set for G and $|V(G)| = n$. In what follows we do not distinguish between an element of the group and a vertex of the Cayley graph. The proof is by induction. Suppose that $\beta(G^i)/n^i = \beta(G)/n$ for $i = 1, 2, \ldots, k-1$.

CLAIM. *Let* $\mathbf{a} = (a_1, a_2, \ldots, a_k)$ *be any vertex of* G^k. *Then* $\{(ga_1, ga_2, \ldots, ga_k) \mid g \in V(G)\}$ *is isomorphic to* G.

Let $T = \{(ga_1, ga_2, \ldots, ga_k) \mid g \in V(G)\}$. Define $\phi : G \to T$ by $\phi(h) = (ha_1, ha_2, \ldots, ha_k)$. Now $g \sim h$ if and only if there exists $s \in S$ such that $gs = h$. Moreover, since G is Abelian for any $a \in G$ then $gas = gsa = ha$. Thus $ga \sim ha$ if and only if $g \sim h$. Since this holds in every coordinate, we have $\phi(g) \sim \phi(h)$ if and only if $g \sim h$. Thus the claim is proved.

For each $\mathbf{x} \in V(G^k)$ set $T_{\mathbf{x}} = \{g\mathbf{x} \mid g \in V(G)\}$. If $\mathbf{c} \in T_{\mathbf{x}} \cap T_{\mathbf{z}}$ then there exist $f, g \in G$ such that $c_i = fx_i = gz_i$ for $i = 1, 2, \ldots, k$. Thus $x_i = f^{-1}gz_i$ and it follows that any vertex of $T_{\mathbf{x}}$ is in $T_{\mathbf{z}}$ and conversely. Therefore $T_{\mathbf{x}} = T_{\mathbf{z}}$ and these decompose G^k into $|V(G^{k-1})|$ many copies of G. From Lemma 2.5 the result now follows. □

As a consequence we now know that both odd and even cycles are self-universal.

We can extend the class of known self-universal graphs even further.

THEOREM 5.2. *Let* G *be a graph of order* n *with an automorphism* f *that has a single orbit of size* n. *Then* G *is self-universal.*

Proof. We define an equivalence relation $\equiv$ on $V(G^{k+1})$ by $\mathbf{x} \equiv \mathbf{y}$ if $y = f^l(\mathbf{x})$ for some l (where f is applied coordinatewise). Now each class is of the form $\{\mathbf{x}, f(\mathbf{x}), f^2(\mathbf{x}), \ldots, f^{n-1}(\mathbf{x})\}$. We'll show that the subgraph induced by each such class is isomorphic to G.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_{k+1})$ and $\mathbf{y} = f^l(\mathbf{x}) = (f^l(x_1), f^l(x_2), \ldots, f^l(x_{k+1}))$ (for some $1 \le l \le n-1$) be any two elements of a class. If $x_1 \perp y_1 = f^l(x_1)$ then $\mathbf{x} \perp \mathbf{y}$. If $x_1 \sim y_1 = f^l(x_1)$, then $\mathbf{x} \sim \mathbf{y}$, as if $x_i = f^j(x_1)$; then $y_i = f^l(x_i) = f^{l+j}(x_1) \sim f^j(x_1) = x_i$. It follows that the subgraph of G^{k+1} induced by the class generated by $\mathbf{x}$ is isomorphic to G, and thus G^{k+1} decomposes into copies of G, and again by Lemma 2.5 we are done. □

We now turn to the distribution of the ultimate categorical independence ratio. From Theorem 3.2, we know that there is a gap between $1/2$ and 1. Clearly 0 is not the ultimate categorical independence ratio for any graph G, so $\{A(G) : G \text{ is a graph}\} \subseteq (0, 1/2] \cup \{1\}$. While we do not know if $A(G)$ can be irrational, we can show that the closure of the set above is in fact $[0, 1/2] \cup \{1\}$.

THEOREM 5.3. *For any rational number* $r \in (0, 1/2] \cup \{1\}$ *there is a graph* G_r *with* $A(G_r) = r$.

If $r = 1$, we may take G_r to be any graph with independence number greater than half the number of vertices. Otherwise, let $r = p/l$ (p, l positive integers) and G_r be the Cayley graph $\mathbb{Z}_l$ where x is joined to y if and only if $x - y \in \{p, p+1, \ldots, l-p\}$; it can be easily seen that $\beta(G_r) = p$, and from Theorem 5.1, $A(G_r) = \beta(G_r)/|V(G_r)| = p/l = r$.

6. Open problems. There are a number of open problems concerning the ultimate categorical independence ratio. While we have shown that every rational number $r \in (0, \frac{1}{2}] \cup \{1\}$ is the ultimate categorical independence ratio of a graph, we do not know the following.

PROBLEM 6.1. *Can* $A(G)$ *be irrational?*

There are also families of graphs (such as complete multipartite graphs) for which the determination of $A(G)$ is unknown. (Although, for many cases, the results here will provide a solution.)

PROBLEM 6.2. *What is* $A(G)$ *for a random graph* G? *Are almost all graphs* G *of order* n *self-universal?*

In terms of algorithmic considerations, we have shown that one can determine in polynomial time the ultimate categorical independence ratio of a bipartite graph.

PROBLEM 6.3. *Is $A(G)$ computable? If so, what is its complexity?*

Finally, in many cases, equality does hold in Theorem 4.1, and we do not know of any examples where equality does not hold.

PROBLEM 6.4. *Is $A(G \cup H) = \max\{A(G), A(H)\}$?*

REFERENCES

[1] G. CHARTRAND AND L. LESNIAK, *Graphs and Digraphs*, Prindle, Weber & Schmidt International Series, London, 1986.

[2] C. BERGE, *Graphs and Hypergraphs*, North Holland, New York, 1979.

[3] D. DUFFUS, B. SANDS, AND R. E. WOODROW, *On the chromatic number of product of graphs*, J. Graph Theory, 9 (1986), pp. 487–495.

[4] M. EL–ZAHAR AND N. SAUER, *The chromatic number of the product of two 4-chromatic graphs is 4*, Combinatorica, 5 (1985), pp. 121–126.

[5] D. GELLER AND S. STAHL, *The chromatic number and other functions of the lexicographic product*, J. Combin. Theory Ser. B, 19 (1975), pp. 87–95.

[6] E. N. GILBERT, Technical memorandum, Bell Telephone Laboratories, Murray Hill, NJ, 1972, manuscript.

[7] G. HAHN, P. HELL, AND S. POLJAK, *On the Ultimate Independence Ratio of a Graph*, Discrete Math., 127 (1994), pp. 213–220.

[8] F. HARARY, *Graph Theory*, Addison–Wesley, London, 1972.

[9] P. HELL AND F. S. ROBERTS, *Analogues of the Shannon capacity of a graph*, in Theory and Practice of Combinatorics, North-Holland Math. Stud., Vol. 60, North–Holland, Amsterdam-New York, 1982, pp. 155–168.

[10] P. HELL, X. YU, AND H. ZHOU, *Independence ratios of graph powers*, Discrete Math.,·27 (1994), pp. 213–220.

[11] A. J. W. HILTON, R. RADO, AND S. H. SCOTT, *A (< 5)-colour theorem for planar graphs*, Bull. London Math. Soc., 5 (1973), pp. 302–306.

[12] L. LOVÁSZ, *On the Shannon capacity of a graph*, IEEE Trans. Inform. Theory, 25 (1979), pp. 1–7.

[13] F. S. ROBERTS, *Graph Theory and its Applications to Problems of Society*, CBMS–NSF Monographs 29, Society for Industrial and Applied Mathematics, Philadelphia, 1978.

[14] ———, *On the mobile radio frequency assignment problem and the traffic light phasing problem*, Ann. New York Acad. Sci., 319 (1978), pp. 466–483.

[15] M. ROSENFELD, *On a Problem of C.E. Shannon in Graph Theory*, Proc. Amer. Math. Soc., 18 (1967), pp. 315–319.

[16] C. E. SHANNON, *The zero error capacity of a noisy channel*, I.R.E., Trans. on Inform. Theory, IT-2 (1956), pp. 8–19.

[17] P. M. WEICHSEL, *The Kronecker product of graphs*, Proc. Amer. Math. Soc., 13 (1962), pp. 47-52.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 301–308, May 1996

012

RANDOM WALKS ON REGULAR AND IRREGULAR GRAPHS*

DON COPPERSMITH†, URIEL FEIGE‡, AND JAMES SHEARER†

Abstract. For an undirected graph and an optimal cyclic list of all its vertices, the *cyclic cover time* is the expected time it takes a simple random walk to travel from vertex to vertex along the list until it completes a full cycle. The main result of this paper is a characterization of the cyclic cover time in terms of simple and easy-to-compute graph properties. Namely, for any connected graph, the cyclic cover time is $\Theta(n^2 d_{ave}(d^{-1})_{ave})$, where n is the number of vertices in the graph, d_{ave} is the average degree of its vertices, and $(d^{-1})_{ave}$ is the average of the inverse of the degree of its vertices. Other results obtained in the processes of proving the main theorem are a similar characterization of minimum resistance spanning trees of graphs, improved bounds on the cover time of graphs, and a simplified proof that the maximum commute time in any connected graph is at most $4n^3/27 + o(n^3)$.

Key words. random walks, graphs, electrical resistance

AMS subject classifications. 05C99, 60J15, 68R10

1. Introduction.

Let $G = G(V, E)$ be a simple connected graph on n vertices and m edges. For any vertex $v \in V$, d_v denotes the degree of v (the number of edges incident with v). We consider random walks on G, where at each step the random walk moves to a vertex chosen at random with uniform probability from the neighbors of the current vertex. For two vertices $u, v \in V$, the *hitting time* $H[u, v]$ is the expected number of steps it takes a walk that starts at u to reach v, and the *commute time* $C[u, v]$ is the expected number of steps that it takes a walk to go from u to v and back to u (that is, $C[u, v] = H[u, v] + H[v, u]$). The *cover time* of a graph $EC[G]$ is the expected number of steps it takes a random walk to visit all vertices of the graph, starting at the worst possible vertex (that maximizes this value).

Aleliunas et al. [1] showed that for any connected graph, $EC[G] < 2nm$. This bound has been refined by Kahn et al. [13], who proved a bound of $EC[G] \leq 4n^2 d_{ave}/d_{\min}$, where d_{ave} is the average degree of the graph, and $d_{\min}$ is its minimum degree. This bound takes into account the structure of the graph: for regular graphs the bound is low, $O(n^2)$, whereas for irregular graphs, those which have a high ratio $d_{ave}/d_{\min}$, the bound is higher.

In this paper we further study the relation between "regularity" of a graph and random walks. However, we use a different measure of regularity, namely $d_{ave}(d^{-1})_{ave}$, where $(d^{-1})_{ave}$ is the average of the inverse of the degrees of the vertices. This measure obtains its minimum 1 on regular graphs and its maximum $\Omega(n)$ on highly irregular graphs that have a linear number of vertices of constant degree and a linear number of vertices of degree $\Omega(n)$. It was argued in [11] that this measure is preferable to $d_{ave}/d_{\min}$, since it is more robust. Introduction of even a single vertex of small degree can cause $d_{ave}/d_{\min}$ to increase by a multiplicative factor of $\Theta(n)$, whereas $d_{ave}(d^{-1})_{ave}$ would increase by at most a constant factor.

Using our regularity measure, we provide an almost tight characterization of a property that is closely related to the cover time. The *cyclic cover time* $ECC[G]$ is

* Received by the editors December 27, 1993; accepted for publication (in revised form) June 30, 1995.

† IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 (copper@watson.ibm.com and shearer@watson.ibm.com).

‡ Department of Applied Math and Computer Science, The Weizmann Institute, Rehovot, Israel (feige@wisdom.weizmann.ac.il). The research of this author was supported by a Koret Foundation fellowship.

the expected number of steps it takes to visit all vertices of the graph in a prespecified cyclic order for the best cyclic order (that minimizes this value). That is,

$$ECC[G] = H[v_{i_1}, v_{i_2}] + H[v_{i_2}, v_{i_3}] + \cdots + H[v_{i_{n-1}}, v_{i_n}] + H[v_{i_n}, v_{i_1}]$$

where $(i_1, i_2, \ldots, i_n)$ is a permutation that minimizes the above sum. Clearly, $ECC[G] > EC[G]$.

Our main theorem states that for any graph G, $ECC[G]$ is characterized up to a constant factor by $d_{ave}(d^{-1})_{ave}$.

THEOREM 1. *For any connected graph G,*

$$\left(\sum_{v\in V} d_v\right)\left(\sum_{v\in V} \frac{1}{d_v+1}\right) \le ECC[G] \le \frac{10}{3}\left(\sum_{v\in V} d_v\right)\left(\sum_{v\in V} \frac{1}{d_v+1}\right).$$

Observe that $\sum_{v\in V} d_v = 2m = nd_{ave}$ and $\sum_{v\in V} \frac{1}{d_v} = n(d^{-1})_{ave}$. Thus the features of the graph that characterize its cyclic cover time can be abstracted to the single easily computable quantity $d_{ave}(d^{-1})_{ave}$ (in particular, it is computable in deterministic logarithmic space), with loss of only a constant factor in the accuracy of the value computed for the cyclic cover time.

A simple corollary of Theorem 1 is that for any connected graph, $EC[G] < \frac{10}{3}n^2 d_{ave}(d^{-1})_{ave}$. This is an improvement of the [13] bound on the cover time, since for every graph, $d_{ave}(d^{-1})_{ave} \le d_{ave}/d_{min}$. However, the bounds that we obtain on the cover time are far from tight. Observe that the bounds that we derive are $\Omega(n^2)$, whereas the cover time of a graph may be as low as $O(n \log n)$ (for a clique or a star). Better bounds on the cover time can sometimes be obtained from the relation $EC[G] \le \max_{u,v}[H[u,v]] \ln n$ [14]. The maximum hitting time on a graph can be computed in polynomial time. Alternatively, it can be bounded from above by the maximum commute time, which in turn can be bounded by $2mD$, where D is the diameter of the graph, or even characterized exactly by $2mR_{\text{eff}}$, where R_{eff} is the maximum *effective resistance* in the graph (see [5], and also §3). We remark that it is not known whether the diameter of a graph or its maximum effective resistance are computable in deterministic logspace.

2. A travelling salesperson formulation of cyclic cover time. For any undirected connected graph $G = G(V, E)$, we can construct the directed weighted complete graph $G' = G'(V, W)$, where W is the sequence of weights associated with the edges of G'. The vertex set of G' is identical to that of G. Any two vertices $u, v \in V$ are connected by two antiparallel directed edges in G'. The weight of edge (u, v) is set to be identical to the hitting time $H[u, v]$. The problem of computing the cyclic cover time of G is thus equivalent to solving *travelling salesperson* (TSP, finding the Hamiltonian cycle of least weight) on the directed graph G'. (One may regard the problem of finding the optimal cyclic cover tour as a *drunken salesperson* problem, where the salesperson wanders at random from town to town, searching for the next town on his or her list. What is the optimal ordering of towns on the list?)

The directed graph G' can be made into an undirected graph, which we also denote by G' (so as not to introduce additional notation). For any pair of vertices $u, v \in V$, the two antiparallel edges are replaced by a single edge with weight equal to the commute time $C[u, v]$. Let $TSP(G')$ denote the weight of the minimum weight Hamiltonian cycle in the undirected graph G'. Then $TSP(G')$ is equal to exactly twice the cyclic cover time on G, by the well-known fact that along any cycle, the

sum of hitting times going along one direction is equal to the sum of hitting times going along the opposite direction (see, e.g., [7]).

In order to obtain an upper bound on the cyclic cover time, we shall use the following relation between spanning trees and Hamiltonian cycles. Let $MST(G')$ denote the weight of the minimum weight spanning tree of G'. Clearly, $MST(G') \leq TSP(G')$. Furthermore, if the weights satisfy the triangle inequality, then it is a simple matter to show that $TSP(G') \leq 2MST(G')$ (see, e.g., [12]). Thus, in our case of cyclic cover time, it remains to bound $MST(G')$.

Remark. We do not know if exact computation of the cyclic cover time is NP-hard. The algorithm of Christofides [6, 12] approximates *metric* TSP within a factor of 3/2 and can be applied to approximate the cyclic cover time, as the commute time (weight of the edges in G') is computable in polynomial time, and satisfies the triangle inequality (that is, $C[u,w] \leq C[u,v] + C[v,w]$). By [2], there is some constant ϵ such that it is NP-hard to approximate metric TSP to within a ratio of $1+\epsilon$. We do not know whether the extra structure of our graph G' can be exploited to improve over the approximation ratio obtainable for metric TSP.

3. Random walks and electrical resistance. There is a well-known correspondence between random walks and resistance of electrical networks [8, 5, 15]. View each edge of G as a resistor of 1 ohm. The *effective resistance* between vertices u and v, denoted by $R[u,v]$, is the voltage that develops at u if a current of 1 amp is injected into u and v is grounded. In [5] it is shown that the effective resistance captures the commute time. Namely, for any connected graph with m edges, and any two vertices u and v,

$$C[u,v] = 2mR[u,v].$$

We shall use three properties about effective resistance:

1. *Serial connection:* resistors that are connected in series can be replaced by a single resistor whose resistance is the sum of the resistances.

2. *Parallel connection:* resistors that are connected in parallel can be replaced by a single resistor whose conductance (the inverse of resistance) is the sum of the conductances.

3. *Shortcut principle:* the effective resistance between any two vertices in a network does not increase if we "short cut" (connect by a resistor of resistance 0) vertices in the network.

The above three properties imply the following proposition.

PROPOSITION 2. *The effective resistance between any two vertices $u, v \in V$ satisfies*

$$R[u,v] \geq \frac{1}{d_u + 1} + \frac{1}{d_v + 1}.$$

Proof. We first show that if u and v are not connected by an edge in G (that is, $(u,v) \notin E$), then $R[u,v] \geq \frac{1}{d_u} + \frac{1}{d_v}$. Collapse all vertices of $V \setminus \{u,v\}$ to a single vertex w, obtaining a graph GS. By the shortcut principle, $R_{GS}[u,v] \leq R[u,v]$. By parallel connection, $R_{GS}[u,w] = \frac{1}{d_u}$. Likewise, $R_{GS}[w,v] = \frac{1}{d_v}$. By serial connection, $R_{GS}[u,v] = R_{GS}[u,w] + R_{GS}[w,v] = \frac{1}{d_u} + \frac{1}{d_v}$.

We now consider the case that $(u,v) \in E$. Construct from G a graph GW by removing the edge (u,v) and adding two new vertices w_1 and w_2, together with the four edges (u,w_1), (w_1,v), (u,w_2), and (w_2,v). Observe that $R_{GW}[u,v] = R[u,v]$,

since the pair of edges $[(u, w_i), (w_i, v)]$ (for $i = 1$ or for $i = 2$) can be replaced by a single 2 ohm resistor (serial connection principle), and thereafter the two 2 ohm parallel resistors can be replaced by a single 1 ohm resistor (parallel connection principle), giving back the original edge (u, v). In GW, there is no edge (u, v), and the degree of each of the vertices u and v is increased by 1. Continue as above. □

As a corollary we obtain the easy direction of Theorem 1.

COROLLARY 3. *For any connected graph, the cyclic cover time satisfies*

$$ECC \geq \left(\sum_{v \in V} d_v\right)\left(\sum_{v \in V} \frac{1}{d_v + 1}\right).$$

Proof. Consider an arbitrary cyclic ordering of the vertices of G. The sum of resistances along this cyclic order is at least $2\sum_{v \in V} \frac{1}{d_v+1}$. Hence the sum of commute times is at least $2m \cdot 2\sum_{v \in V} \frac{1}{d_v+1}$, and the cyclic cover time is at least $2m \sum_{v \in V} \frac{1}{d_v+1}$. □

The above bound is tight for complete graphs.

In order to obtain an upper bound on the cyclic cover time, or $TSP(G')$, we shall bound $MST(G')$. For reasons of convenience, the weights assigned to edges of G' are the effective resistances $R[u, v]$ (rather than the commute times $C[u, v]$). It will turn out that for the proof of Theorem 1, there is no need to consider edges of G' that were not edges in the original graph G, and they are assigned "infinite weight." Our goal is to upper bound R_{span}, the weight of the spanning tree of minimum weight (resistance). By previous discussion it follows that the cyclic cover time of G is at most $2mR_{span}$.

The approach of using R_{span} in order to bound the cover time originates from [1, 13]. Our current work was motivated by the conjecture in [11] that $R_{span} = \Theta(\sum_{v \in V} \frac{1}{d_v})$.

4. The excess resistance lemma.

DEFINITION. *The* excess resistance $\delta[(u, v)]$ *of edge* $(u, v) \in E$ *is defined by*

$$\delta[(u, v)] = R[u, v] - \left(\frac{1}{d_u + 1} + \frac{1}{d_v + 1}\right).$$

Observe that by Proposition 2, $\delta[(u, v)] \geq 0$, with equality if and only if $d_u = d_v$ and there are $d_u - 1$ intermediate vertices that are adjacent both to u and v (in addition to the one edge (u, v)).

The following identity (the *excess resistance lemma*) plays a fundamental role in our paper.

LEMMA 4. *For any graph $G = G(V, E)$ with c connected components, the sum of excess resistances along its edges satisfies*

$$\sum_{(u,v) \in E} \delta[(u, v)] = \sum_{v \in V} \frac{1}{d_v + 1} - c.$$

Proof. We present without proof an identity due to Foster [11, 15].

LEMMA 5. *Let $G = G(V, E)$ be a graph with n vertices and c connected components. The sum of effective resistances along the edges of G satisfies*

$$\sum_{(u,v) \in E} R[u, v] = n - c.$$

We return to the proof of Lemma 4. By definition of the excess resistance,

$$\sum_{(u,v)\in E} R[u,v] = \sum_{(u,v)\in E} \left(\frac{1}{d_u+1} + \frac{1}{d_v+1} + \delta[(u,v)] \right)$$

$$= \sum_v \frac{d_v}{d_v+1} + \sum_{(u,v)\in E} \delta[(u,v)].$$

The excess resistance identity follows from Foster's identity. □

5. Bounds on the commute time. The following theorem improves the leading constant (by a factor of 3) over a similar theorem proved in [11]. This results in improved time estimates for deciding undirected s-t-connectivity by space-efficient randomized algorithms [11].

THEOREM 6. *For any pair of vertices s and t in a connected graph, the commute time satisfies*

$$C[s,t] < 6m \sum_{v\in V} \frac{1}{d_v+1}.$$

Proof. Consider any simple path $p = s, v_1, v_2, \ldots, v_\ell, t$, connecting s and t (not necessarily the shortest path). We use the excess resistance lemma to bound the resistance along this path.

$$R[p] = R[s,v_1] + R[v_1,v_2] + \cdots + R[v_\ell,t]$$

$$= \frac{1}{d_s+1} + \frac{1}{d_{v_1}+1} + \delta[(s,v_1)] + \cdots + \frac{1}{d_{v_\ell}+1} + \frac{1}{d_t+1} + \delta[(v_\ell,t)]$$

$$< 2\sum_{v\in V} \frac{1}{d_v+1} + \sum_{(u,v)\in E} \delta[(u,v)] < 3\sum_{v\in V} \frac{1}{d_v+1}.$$

The sum of commute times along path p is an upper bound on $C[s,t]$, and it satisfies

$$C[p] = 2mR[p] < 6m \sum_{v\in V} \frac{1}{d_v+1}. \qquad \square$$

The above bound on the maximum commute time in a graph is the best of what is possible (up to a low-order term). This can be verified by considering specific examples, such as a path of n edges (and $n+1$ vertices), for which the commute time between the two endpoints is $2n^2$. A more striking demonstration of the tightness of Theorem 6 is provided by the following corollary.

COROLLARY 7. *For any two vertices s and t in a connected graph of n vertices, the commute time satisfies*

$$C[s,t] \le \frac{4}{27}n^3 + o(n^3).$$

Proof. We first transform G into a new graph G', for which almost all vertices have degree at least 2 by repeatedly deleting vertices of degree 1 (except for s and t),

and re-appending them as a path leading out of t. This does not change the number of edges in G nor the effective resistance between s and t. Hence $C_{G'}[s,t] = C[s,t]$. Note that in G' at most two vertices have degree 1 (s and the vertex at the end of the path leading out of t).

Assume now that G' has k vertices of degree at least $\sqrt{n}$ and $n-k$ vertices of degree less than $\sqrt{n}$. Then the number of edges in G is at most $\frac{k^2}{2}+(n-k)\sqrt{n}$, and the expression $\sum_{v\in V}\frac{1}{d_v+1}$ is at most $\frac{n-k}{3}+\frac{k}{\sqrt{n}}+1$. The product of these two expressions is maximized when $k \simeq 2n/3$, and the bound on the commute time follows. □

The above corollary is not new. Brightwell and Winkler [4] show that the maximum hitting time in a graph is at most $4n^3/27 + o(n^3)$ and, in fact, they find the graph for which the hitting time is maximized (the *lollipop* graph). A similar bound on the commute time is implicit in [7]. The same upper bound also holds for the cyclic cover time, as proven in [10].

6. Regular graphs. We now turn to bound R_{span}, the resistance of the spanning tree of minimum resistance. We start with the case of d-regular graphs. For d-regular graphs, Kahn et al. [13] prove that the cover time is bounded by $4n^2$. Their proof actually shows that $R_{span} \le 4n/d$, and hence the cyclic cover time is bounded by $4n^2$. We improve upon the bounds of [13].

THEOREM 8. *In connected d-regular graphs, the resistance of any spanning tree T satisfies*

$$\frac{2(n-1)}{d+1} \le R[T] < \frac{3(n-1)}{d+1}.$$

Proof. The lower bound on $R[T]$ is a trivial consequence of Proposition 2. The upper bound is by application of Lemma 4. □

Thus all spanning trees in a regular graph have roughly the same resistance. The *clique* demonstrates that there are d-regular graphs for which all spanning trees have the minimum possible value. The *necklace* (a cycle of cliques, where an edge is removed from each clique so as to allow the connection of adjacent cliques without violating regularity) demonstrates that there are d-regular graphs for which all spanning trees have nearly the maximum possible value. The following graph demonstrates that it is possible to obtain a ratio of nearly 3/2 between the minimum and maximum resistance spanning trees (for large d, and $n >> d$). Consider $\frac{n}{d+1}$ disjoint cliques, each of size $d+1$. In each clique, remove a cycle (of $(d+1)$ edges). Now create a cycle of cliques, where each clique is connected to each neighboring clique by $(d+1)$ *connecting* edges. It can be verified that the resistance of clique edges is roughly $2/d$, whereas the resistance of connecting edges is roughly $3/d$. Hence a spanning tree that is based mainly on connecting edges will have resistance 3/2 times as high as a spanning tree that is based mostly on clique edges.

7. Proof of the main theorem. Theorem 8 can be generalized to arbitrary graphs, showing that the resistance of any spanning tree satisfies

$$\frac{n-2}{d_{max}+1} + \sum_v \frac{1}{d_v+1} \le R[T] < \frac{n-1}{d_{min}+1} + 2\sum_v \frac{1}{d_v+1}.$$

For the purpose of proving Theorem 1, we need an upper bound on R_{span} that does not depend on d_{min}. Our problems would be solved if we could replace $\frac{n-1}{d+1}$ in Theorem 8 with $\sum_{v\in V}\frac{1}{d_v+1}$. However, the resulting proposition would be false.

The lower bound of Theorem 8 on $R[T]$ would be off by a factor of roughly 2, as can be seen by considering a bipartite graph $G_{a,b}$, with a left-hand-side vertices, b right-hand-side vertices, and $b >> a$. The effective resistance along any edge is $\frac{a+b-1}{ab}$ (by symmetry, and Foster's identity), and hence all spanning trees have resistance roughly b/a (assuming $b >> a$). For $G_{a,b}$, $\sum_{v\in V} \frac{1}{d_v+1} \simeq b/a$. Hence $R[T] \simeq \sum_{v\in V} \frac{1}{d_v+1}$.

The upper bound of Theorem 8 on $R[T]$ is violated to a more dramatic extent. Consider the graph composed of a clique on k vertices, with $\sqrt{k}$ additional vertices, each connected via *connecting edges* to $\sqrt{k}$ distinct vertices in the clique. For this graph, $\sum_{v\in V} \frac{1}{d_v+1} \simeq 2$, whereas the maximal resistance spanning tree (using all connecting edges) has resistance roughly $\sqrt{k}$.

Hence, unlike the case for regular graphs, it is not true in general that for any spanning tree T, $R[T] = O(\sum_{v\in V} \frac{1}{d_v+1})$. However, the following theorem shows that for any connected graph, the average resistance of a spanning tree (when the spanning tree is chosen at random) is $O(\sum_{v\in V} \frac{1}{d_v+1})$.

THEOREM 9. *For any connected graph G, the expected resistance of the spanning trees of G satisfies*

$$\mathcal{E}_T[R[T]] \leq \frac{10}{3} \sum_{v\in V} \frac{1}{d_v+1}.$$

Proof. The fraction of spanning trees that use edge (u,v) is exactly $R[(u,v)]$ [3]. Hence,

$$\mathcal{E}_T[R[T]] = \sum_{(u,v)\in E} (R[(u,v)])^2 = \sum_{(u,v)\in E} \left(\frac{1}{d_u+1} + \frac{1}{d_v+1} + \delta[(u,v)] \right)^2.$$

To bound the above sum we use the following technical lemma.

LEMMA 10. *For any edge (u,v),*

$$\left(\frac{1}{d_u+1} + \frac{1}{d_v+1} + \delta[(u,v)] \right)^2 \leq \frac{2}{d_u(d_u+1)} + \frac{2}{d_v(d_v+1)} + \frac{4}{3}\delta[(u,v)].$$

Proof. For any integer positive values of d_v and d_u, the above inequality holds for the minimum possible value $\delta = 0$ and maximum value $\delta = 1 - \frac{1}{d_u+1} - \frac{1}{d_v+1}$ (calculations omitted). Since the left-hand side of the inequality is quadratic in δ, and the right-hand side is linear in δ, the inequality must also hold for all intermediate values of δ. □

Hence

$$\mathcal{E}_T[R[T]] \leq \sum_{(u,v)\in E} \left(\frac{2}{d_u(d_u+1)} + \frac{2}{d_v(d_v+1)} + \frac{4}{3}\delta[(u,v)] \right)$$

$$\leq 2 \sum_{v\in V} \frac{d_v}{d_v(d_v+1)} + \frac{4}{3} \sum_{(u,v)\in E} \delta[(u,v)] \leq \frac{10}{3} \sum_{v\in V} \frac{1}{d_v+1}. \qquad \square$$

This completes the proof of Theorem 1. Observe that if we could improve the leading constant in Theorem 1 from 10/3 to 3, this would imply an alternative proof to $ECC \leq 4n^3/27 + o(n^3)$ [10], along the lines of Corollary 7. One might hope to achieve this by bounding the minimum spanning tree, or R_{span}, rather than the

average resistance of random spanning trees. However, it is not true that for any graph $R_{span} \leq 3\sum_{v\in V} \frac{1}{d_v+1}$. Consider a full binary tree of depth D (having $2^{D+1}-1$ vertices). Replace every leaf by a k-clique, with $k >> d$. It can be easily verified that $\sum_{v\in V} \frac{1}{d_v+1} \simeq \frac{5}{4} \cdot 2^D$, and that for any spanning tree T, $R[T] \simeq 4 \cdot 2^D$. Hence $R_{span} \simeq \frac{16}{5} \sum_{v\in V} \frac{1}{d_v+1}$.

Acknowledgments. We gratefully acknowledge helpful conversations with Prabhakar Raghavan, Madhu Sudan, and Prasad Tetali.

REFERENCES

[1] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff, *Random walks, universal traversal sequences, and the complexity of maze problems*, in Proc. of 20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1979, pp. 218–223.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and hardness of approximation problems*, in Proc. of 33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, PA, 1992, pp. 14–23.

[3] B. Bollobas, *Graph Theory: An Introductory Course*, Springer, New York, 1979.

[4] G. Brightwell and P. Winkler, *Maximum hitting times for random walks on graphs*, Random Structures Algorithms, 1 (1990), pp. 263–276.

[5] A. Chandra, P. Raghavan, W. Ruzzo, R. Smolensky, and P. Tiwari, *The electrical resistance of a graph captures its commute and cover times*, in Proc. 21st Annual ACM Symposium on Theory of Computing, Seattle, WA, 1989, pp. 574–586.

[6] N. Christofides, *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem*, Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[7] D. Coppersmith, P. Tetali, and P. Winkler, *Collisions among random walks on a graph*, SIAM J. Discrete Math., 6 (1993), pp. 363–374.

[8] P. G. Doyle and J. L. Snell, *Random Walks and Electrical Networks*, The Mathematical Association of America, Washington, DC, 1984.

[9] U. Feige, *A tight upper bound on the cover time for random walks on graphs*, Random Structures Algorithms, 6 (1995), pp. 51–54.

[10] ———, *A randomized time-space tradeoff of $\tilde{O}(m\hat{R})$ for USTCON*, in Proc. of 34th Annual Symposium on Foundations of Computer Science, Palo Alto, CA, 1993, pp. 238–246.

[11] R. Foster, *The Average Impedance of an Electrical Network*, Contributions to Applied Mechanics (Reissner anniversary volume), Edwards Brothers, Inc., Ann Arbor, MI, 1949, pp. 333–340.

[12] M. Garey and D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[13] J. D. Kahn, N. Linial, N. Nisan, and M. E. Saks, *On the cover time of random walks on graphs*, J. Theoret. Probab., 2 (1989), pp. 121–128.

[14] P. C. Matthews, *Covering problems for Brownian motion on spheres*, Ann. Probab., 16 (1988), pp. 189–199.

[15] P. Tetali, *Random walks and the effective resistance of networks*, J. Theoret. Probab., 4 (1991), pp. 101–109.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 309–316, May 1996

013

THE $L(2,1)$-LABELING PROBLEM ON GRAPHS*

GERARD J. CHANG† AND DAVID KUO†

Abstract. An $L(2,1)$-labeling of a graph G is a function f from the vertex set $V(G)$ to the set of all nonnegative integers such that $|f(x) - f(y)| \geq 2$ if $d(x,y) = 1$ and $|f(x) - f(y)| \geq 1$ if $d(x,y) = 2$. The $L(2,1)$-labeling number $\lambda(G)$ of G is the smallest number k such that G has an $L(2,1)$-labeling with $\max\{f(v) : v \in V(G)\} = k$. In this paper, we give exact formulas of $\lambda(G \cup H)$ and $\lambda(G + H)$. We also prove that $\lambda(G) \leq \Delta^2 + \Delta$ for any graph G of maximum degree Δ. For odd-sun-free (OSF)-chordal graphs, the upper bound can be reduced to $\lambda(G) \leq 2\Delta + 1$. For sun-free (SF)-chordal graphs, the upper bound can be reduced to $\lambda(G) \leq \Delta + 2\chi(G) - 2$. Finally, we present a polynomial time algorithm to determine $\lambda(T)$ for a tree T.

Key words. $L(2,1)$-labeling, T-coloring, union, join, chordal graph, perfect graph, tree, bipartite matching, algorithm

AMS subject classifications. 05C15, 05C78

1. Introduction.

The *channel assignment problem* is to assign a channel (nonnegative integer) to each radio transmitter so that interfering transmitters are assigned channels whose separation is not in a set of disallowed separations. Hale [11] formulated this problem into the notion of the *T-coloring* of a graph, and the T-coloring problem has been extensively studied over the past decade (see [4, 5, 7, 13, 14, 16, 17, 19]).

Roberts [15] proposed a variation of the channel assignment problem in which "close" transmitters must receive different channels and "very close" transmitters must receive channels that are at least two channels apart. To formulate the problem in graphs, the transmitters are represented by the vertices of a graph; two vertices are "very close" if they are adjacent in the graph and "close" if they are of distance two in the graph. More precisely, an *$L(2,1)$-labeling* of a graph G is a function f from the vertex set $V(G)$ to the set of all nonnegative integers such that $|f(x) - f(y)| \geq 2$ if $d(x,y) = 1$ and $|f(x) - f(y)| \geq 1$ if $d(x,y) = 2$. A *k-$L(2,1)$-labeling* is an $L(2,1)$-labeling such that no label is greater than k. The *$L(2,1)$-labeling number* of G, denoted by $\lambda(G)$, is the smallest number k such that G has a k-$L(2,1)$-labeling.

Griggs and Yeh [10] and Yeh [21] determined the exact values of $\lambda(P_n)$, $\lambda(C_n)$, and $\lambda(W_n)$, where P_n is a *path* of n vertices, C_n is a *cycle* of n vertices, and W_n is an *n-wheel* obtained from C_n by adding a new vertex adjacent to all vertices in C_n. For the n-cube Q_n, Jonas [12] showed that $n + 3 \leq \lambda(Q_n)$. Griggs and Yeh [10] showed that $\lambda(Q_n) \leq 2n + 1$ for $n \geq 5$. They also determined $\lambda(Q_n)$ for $n \leq 5$ and conjectured that the lower bound $n + 3$ is the actual value of $\lambda(Q_n)$ for $n \geq 3$. Using a coding theory method, Whittlesey, Georges, and Mauro [20] proved that

$$\lambda(Q_n) \leq 2^k + 2^{k-q+1} - 2, \quad \text{where } n \leq 2^k - q \text{ and } 1 \leq q \leq k + 1.$$

In particular, $\lambda(Q_{2^k-k-1}) \leq 2^k - 1$. As a consequence, $\lambda(Q_n) \leq 2n$ for $n \geq 3$.

* Received by the editors March 10, 1993; accepted for publication (in revised form) August 1, 1995.

† Department of Applied Mathematics, National Chiao Tung University, Hsinchu 30050, Taiwan, Republic of China (gjchang@math.nctu.edu.tw). This research was supported in part by the National Science Council under grant NSC82-0208-M009-050. The research of the first author was supported in part by DIMACS.

For a tree T with maximum degree $\Delta \geq 1$, Griggs and Yeh [10] showed that $\lambda(T)$ is either $\Delta+1$ or $\Delta+2$. They proved that the $L(2,1)$-labeling problem is NP-complete for general graphs and conjectured that the problem is also NP-complete for trees.

For a general graph G of maximum degree Δ, Griggs and Yeh [10] proved that $\lambda(G) \leq \Delta^2+2\Delta$. The upper bound was improved to be $\lambda(G) \leq \Delta^2+2\Delta-3$ when G is 3-connected and $\lambda(G) \leq \Delta^2$ when G is of diameter two. Griggs and Yeh conjectured that $\lambda(G) \leq \Delta^2$ in general. To study this conjecture, Sakai [18] considered the class of chordal graphs. She showed that $\lambda(G) \leq (\Delta+3)^2/4$ for any chordal graph G. For a unit interval graph G, which is a very special chordal graph, she also proved that $2\chi(G) - 2 \leq \lambda(G) \leq 2\chi(G)$.

The purpose of this paper is to study Griggs and Yeh's conjectures. We also study $L(2,1)$-labeling numbers of the union and the join of two graphs to generalize results on the n-wheel that is the join of C_n and K_1. For this purpose and a further reason that will become clear in §3, we introduce a related problem, which we call the $L'(2,1)$-labeling problem. The definitions of an *$L'(2,1)$-labeling* f, a *k-$L'(2,1)$-labeling* f, and the *$L'(2,1)$-labeling number* $\lambda'(G)$ are the same as those of an $L(2,1)$-labeling f, a k-$L(2,1)$-labeling f, and the $L(2,1)$-labeling number $\lambda(G)$, respectively, except that the function f is required to be one-to-one. There is a natural connection between $\lambda'(G)$ and the path partition number $p_v(G^c)$ of the complement G^c of G. For any graph G, the *path partition number* $p_v(G)$ is the minimum number k such that $V(G)$ can be partitioned into k paths.

The rest of this paper is organized as follows. Section 2 gives general properties of $\lambda(G)$ and $\lambda'(G)$. Section 3 studies $\lambda(G \cup H)$, $\lambda(G+H)$, $\lambda'(G \cup H)$, and $\lambda'(G+H)$. Section 4 proves that $\lambda(G) \leq \Delta^2 + \Delta$ for a general graph G of maximum degree Δ. This result improves on Griggs and Yeh's result $\lambda(G) \leq \Delta^2 + 2\Delta$. However, there is still a gap in the conjecture $\lambda(G) \leq \Delta^2$. Section 5 studies the upper bounds for subclasses of chordal graphs. Section 6 presents a polynomial time algorithm to determine $\lambda(T)$ of a tree T.

A referee points out that Georges, Mauro, and Whittlesey [8] also solved $\lambda(G+H)$ and $p_v(G+H)$ by a different approach. They actually gave the solutions without introducing the notion of λ'; see the remarks after Lemmas 2.3 and 3.4.

2. Basic properties of λ and λ'.

LEMMA 2.1. *$\lambda(G) \leq \lambda(H)$ and $\lambda'(G) \leq \lambda'(H)$ for any subgraph G of a graph H.*

LEMMA 2.2. *$\lambda(G) \leq \lambda'(G)$ for any graph G. $\lambda(G) = \lambda'(G)$ if G is of diameter at most two.*

LEMMA 2.3. *$p_v(G) = \lambda'(G^c) - |V(G)| + 2$ for any graph G.*

Proof. Suppose f is a $\lambda'(G^c)$-$L'(2,1)$-labeling of G^c. Note that for any two vertices x and y in $V(G)$, if $f(x) = f(y) + 1$, then $(x,y) \notin E(G^c)$ and so $(x,y) \in E(G)$. Consequently, a subset of vertices whose labels form a consecutive segment of integers form a path in G. However, there are at most $\lambda'(G^c) - |V(G)| + 2$ such consecutive segments of integers. Thus $p_v(G) \leq \lambda'(G^c) - |V(G)| + 2$.

On the other hand, suppose $V(G)$ can be partitioned into $k \equiv p_v(G)$ paths in G, say, $(v_{i,1}, v_{i,2}, \ldots, v_{i,n_i})$ for $1 \leq i \leq k$. Consider a dummy path $(v_{0,1})$ and define f by

$$f(v_{i,j}) = \begin{cases} -2, & \text{if } i = 0 \text{ and } j = 1; \\ f(v_{i-1,n_{i-1}}) + 2, & \text{if } 1 \leq i \leq k \text{ and } j = 1; \\ f(v_{i,j-1}) + 1, & \text{if } 1 \leq i \leq k \text{ and } 2 \leq j \leq n_i. \end{cases}$$

It is straightforward to check that f is a $(k+|V(G)|-2)$-$L'(2,1)$-labeling of G^c. Hence $\lambda'(G^c) \leq k + |V(G)| - 2$; i.e., $p_v(G) \geq \lambda'(G^c) - |V(G)| + 2$. □

Remark. Georges, Mauro, and Whittlesey [8, Thm. 1.1] proved that for any graph G of n vertices the following two statements hold.

(i) $\lambda(G) \leq n-1$ if and only if $p_v(G^c) = 1$.

(ii) Suppose r is an integer greater than 1. $\lambda(G) = n + r - 2$ if and only if $p_v(G^c) = r$.

Note that an $L(2,1)$-labeling is precisely a proper vertex coloring with some extra conditions on all vertex pairs of distance at most two. So, $\lambda(G)$ has a natural relation with the chromatic number $\chi(G)$.

For any fixed positive integer k, the *kth power* of a graph G is the graph G^k whose vertex set $V(G^k) = V(G)$ and edge set $E(G^k) = \{(x,y) : 1 \leq d_G(x,y) \leq k\}$.

LEMMA 2.4. $\chi(G) - 1 \leq \lambda(G) \leq 2\chi(G^2) - 2$ *for any graph* G.

Proof. $\chi(G) - 1 \leq \lambda(G)$ follows from definitions. $\lambda(G) \leq 2\chi(G^2) - 2$ follows from the fact that for any proper vertex coloring f of G^2, $2f - 2$ is an $L(2,1)$-labeling of G. □

The *neighborhood* $N(x)$ of a vertex x is the set of all vertices y adjacent to x. The *closed neighborhood* $N[x]$ of x is $\{x\} \cup N(x)$.

LEMMA 2.5 (see [10]). $\lambda(G) \geq \Delta + 1$ *for any graph* G *of maximum degree* Δ. *If* $\lambda(G) = \Delta + 1$, *then* $f(v) = 0$ *or* $\Delta + 1$ *for any* $\lambda(G)$-$L(2,1)$-*labeling* f *and any vertex* v *of maximum degree* Δ. *In this case,* $N[x]$ *contains at most two vertices of degree* Δ *for any* $x \in V(G)$.

LEMMA 2.6. $\lambda'(C_3) = \lambda'(C_4) = 4$ *and* $\lambda'(C_n) = n - 1$ *for* $n \geq 5$.

Proof. The cases of C_3 and C_4 are easy to verify. For $n \geq 5$, $\lambda'(G) \geq n - 1$ by definition. Let $v_0, v_1, \ldots, v_{n-1}$ be vertices of C_n such that v_i is adjacent to v_{i+1} for $0 \leq i \leq n-1$, where $v_n \equiv v_0$. Consider the following labeling:

$$f(v_i) = \begin{cases} i/2, & \text{if } 0 \leq i \leq n-1 \text{ and } i \text{ is even;} \\ \lceil n/2 \rceil + \lceil i/2 \rceil - 1, & \text{if } 0 \leq i \leq n-1 \text{ and } i \text{ is odd.} \end{cases}$$

It is straightforward to check that f is an $(n-1)$-$L'(2,1)$-labeling of C_n. So $\lambda'(C_n) \leq n - 1$. □

LEMMA 2.7. $\lambda'(P_1) = 0$, $\lambda'(P_2) = 2$, $\lambda'(P_3) = 3$, *and* $\lambda'(P_n) = n - 1$ *for* $n \geq 4$.

Proof. The cases of P_1, P_2, P_3, and P_4 are easy to verify. For $n \geq 5$, $\lambda'(P_n) \geq n-1$ by definition. Last, $\lambda'(P_n) \leq \lambda'(C_n) = n - 1$ by Lemmas 2.1 and 2.6. □

3. Union and join of graphs. Suppose G and H are two graphs with disjoint vertex sets. The *union* of G and H, denoted by $G \cup H$, is the graph whose vertex set is $V(G) \cup V(H)$ and edge set is $E(G) \cup E(H)$. The *join* of G and H, denoted by $G + H$, is the graph obtained from $G \cup H$ by adding all edges between vertices in $V(G)$ and vertices in $V(H)$.

LEMMA 3.1. $\lambda(G \cup H) = \max\{\lambda(G), \lambda(H)\}$ *for any two graphs* G *and* H.

Proof. $\lambda(G \cup H) \geq \max\{\lambda(G), \lambda(H)\}$ follows from Lemma 2.1 and the fact that G and H are subgraphs of $G \cup H$. On the other hand, an $L(2,1)$-labeling of G together with an $L(2,1)$-labeling of H makes an $L(2,1)$-labeling of $G \cup H$. Hence $\lambda(G \cup H) \leq \max\{\lambda(G), \lambda(H)\}$. □

LEMMA 3.2. $\lambda'(G \cup H) = \max\{\lambda'(G), \lambda'(H), |V(G)| + |V(H)| - 1\}$ *for any two graphs* G *and* H.

Proof. $\lambda'(G \cup H) \geq \max\{\lambda'(G), \lambda'(H)\}$ follows from Lemma 2.1 and the fact that G and H are subgraphs of $G \cup H$. $\lambda'(G \cup H) \geq |V(G)| + |V(H)| - 1$ follows from the definition of λ'.

Assume f is a $\lambda'(G)$-$L'(2,1)$-labeling of G. There are no two consecutive integers $x < y$ in $[0, \lambda'(G)]$ that are not labels of vertices of G; otherwise we can "compact" the function f to get a $(\lambda'(G)-1)$-$L'(2,1)$-labeling f' of G defined by

$$f'(v) = \begin{cases} f(v), & \text{if } f(v) < x; \\ f(v) - 1, & \text{if } f(v) > x. \end{cases}$$

For the case where $\lambda'(G) \geq |V(G)| + |V(H)| - 1$, there are at least $|V(H)|$ pairwise nonconsecutive integers in $[0, \lambda'(G)]$ that are not labels of vertices of G. We can use them to label the vertices of H. This yields a $\lambda'(G)$-$L'(2,1)$-labeling of $G \cup H$. For the case where $\lambda'(H) \geq |V(G)| + |V(H)| - 1$, similarly, there exists a $\lambda'(H)$-$L'(2,1)$-labeling of $G \cup H$. For the case where $\max\{\lambda'(G), \lambda'(H)\} \leq |V(G)| + |V(H)| - 1$, without loss of generality, we may assume that $|V(G)| \geq |V(H)|$. Let f be a k-$L'(2,1)$-labeling of G such that $k \leq |V(G)| + |V(H)| - 1$ and there are no two consecutive integers in $[0, k]$ that are not labels of vertices of G. Such an f exists for $k = \lambda'(G)$. If $k \leq |V(G)| + |V(H)| - 3$, then $k \leq 2|V(G)| - 3$ and so there exist two consecutive labels $x < y$. In this case, we can "separate" f to get a $(k+1)$-$L'(2,1)$-labeling f' defined by

$$f'(v) = \begin{cases} f(v), & \text{if } f(v) \leq x; \\ f(v) + 1, & \text{if } f(v) \geq y. \end{cases}$$

Continuing this process, we obtain a k-$L'(2,1)$-labeling such that $|V(G)|+|V(H)|-2 \leq k \leq |V(G)| + |V(H)| - 1$ and there are no two consecutive integers in $[0, k]$ that are not labels of vertices of G. Using $|V(H)|$ nonlabels in $[0, |V(G)| + |V(H)| - 1]$ to label the vertices in H, we get a $(|V(G)| + |V(H)| - 1)$-$L'(2,1)$-labeling of $G \cup H$. By the conclusions of the above three cases, $\lambda'(G \cup H) \leq \max\{\lambda'(G), \lambda'(H), |V(G)| + |V(H)| - 1\}$. □

LEMMA 3.3. *$p_v(G \cup H) = p_v(G) + p_v(H)$ for any two graphs G and H.*

Proof. The proof is obvious. □

LEMMA 3.4. *$\lambda(G + H) = \lambda'(G + H) = \lambda'(G) + \lambda'(H) + 2$ for any two graphs G and H.*

Proof. $\lambda(G + H) = \lambda'(G + H)$ follows from Lemma 2.2 and the fact that $G + H$ is of diameter at most two. Also,

$\lambda'(G + H)$
$= p_v((G + H)^c) + |V(G + H)| - 2$ (by Lemma 2.3)
$= p_v(G^c \cup H^c) + |V(G)| + |V(H)| - 2$
$= p_v(G^c) + p_v(H^c) + |V(G)| + |V(H)| - 2$ (by Lemma 3.3)
$= \lambda'(G) + \lambda'(H) + 2$ (by Lemma 2.3). □

Remark. Georges, Mauro, and Whittlesey [8, Cor. 4.6] proved that $\lambda(G + H) = \max\{|V(G)| - 1, \lambda(G)\} + \max\{|V(H)| - 1, \lambda(H)\} + 2$.

LEMMA 3.5. *$p_v(G + H) = \max\{p_v(G) - |V(H)|, p_v(H) - |V(G)|, 1\}$ for any two graphs G and H.*

Proof.

$p_v(G + H)$
$= \lambda'((G + H)^c) - |V(G + H)| + 2$ (by Lemma 2.3)
$= \lambda'(G^c \cup H^c) - |V(G)| - |V(H)| + 2$
$= \max\{\lambda'(G^c), \lambda'(H^c), |V(G)| + |V(H)| - 1\} - |V(G)| - |V(H)| + 2$ (by Lemma 3.2)
$= \max\{\lambda'(G^c) - |V(G)| + 2 - |V(H)|, \lambda'(H^c) - |V(H)| + 2 - |V(G)|, 1\}$
$= \max\{p_v(G) - |V(H)|, p_v(H) - |V(G)|, 1\}$ (by Lemma 2.3). □

Cographs are defined recursively by the following rules.

(1) A vertex is a cograph.

(2) If G is a cograph, then so is its complement G^c.

(3) If G and H are cographs, then so is their union $G \cup H$.

Note that the above definition is the same as one with (2) replaced by the following.

(4) If G and H are cographs, then so is their join $G + H$.

There is a linear time algorithm to identify whether a graph is a cograph (see [3]). In the case of a positive answer, the algorithm also gives a *parsing tree.* Therefore, we have the following consequences.

THEOREM 3.6. *There is a linear time algorithm to compute* $\lambda(G)$, $\lambda'(G)$, *and* $p_v(G)$ *for a cograph* G.

4. Upper bound of λ in terms of maximum degree. For any fixed positive integer k, a *k-stable set* of a graph G is a subset S of $V(G)$ such that every two distinct vertices in S are of distance greater than k. Note that 1-stability is the usual stability.

THEOREM 4.1. $\lambda(G) \leq \Delta^2 + \Delta$ *for any graph* G *with maximum degree* Δ.

Proof. Consider the following labeling scheme on $V(G)$. Initially, all vertices are unlabeled. Let $S_{-1} = \emptyset$. When S_{i-1} is determined and not all vertices in G are labeled, let

$$F_i = \{x \in V(G) : x \text{ is unlabeled and } d(x,y) \geq 2 \text{ for all } y \in S_{i-1}\}.$$

Choose a *maximal* 2-stable subset S_i of F_i; i.e., S_i is a 2-stable subset of F_i but S_i is not a proper subset of any 2-stable subset of F_i. Note that in the case where $F_i = \emptyset$, i.e., for any unlabeled vertex x there exists some vertex $y \in S_{i-1}$ such that $d(x,y) < 2$, $S_i = \emptyset$. In any case, label all vertices in S_i by i. Then increase i by one and continue the above process until all vertices are labeled. Assume k is the maximum label used, and choose a vertex x whose label is k. Let

$I_1 = \{i : 0 \leq i \leq k-1 \text{ and } d(x,y) = 1 \text{ for some } y \in S_i\}$,

$I_2 = \{i : 0 \leq i \leq k-1 \text{ and } d(x,y) \leq 2 \text{ for some } y \in S_i\}$,

$I_3 = \{i : 0 \leq i \leq k-1 \text{ and } d(x,y) \geq 3 \text{ for all } y \in S_i\}$.

It is clear that $|I_2| + |I_3| = k$. Since the total number of vertices y with $1 \leq d(x,y) \leq 2$ is at most $\deg(x) + \sum\{\deg(y) - 1 : (y,x) \in E(G)\} \leq \Delta + \Delta(\Delta - 1) = \Delta^2$, we have $|I_2| \leq \Delta^2$. Also, there exist only $\deg(x) \leq \Delta$ vertices adjacent to x, so $|I_1| \leq \Delta$. For any $i \in I_3$, $x \notin F_i$; otherwise $S_i \cup \{x\}$ is a 2-stable subset of F_i, which contradicts the choice of S_i. That is, $d(x,y) = 1$ for some vertex y in S_{i-1}; i.e., $i - 1 \in I_1$. So, $|I_3| \leq |I_1|$. Then,

$$\lambda(G) \leq k = |I_2| + |I_3| \leq |I_2| + |I_1| \leq \Delta^2 + \Delta. \qquad \square$$

Jonas [12] proved that $\lambda(G) \leq \Delta^2 + 2\Delta - 4$ if $\Delta(G) \geq 2$. For the case of $\Delta = 3$, this bound improves the bound in Theorem 4.1 from 12 to 11.

5. Subclasses of chordal graphs. A graph is *chordal* (or *triangulated*) if every cycle of length greater than three has a *chord*, which is an edge joining two nonconsecutive vertices of the cycle. Chordal graphs have been extensively studied as a subclass of perfect graphs (see [9]). For any graph G, $\chi(G)$ denotes the chromatic number of G and $\omega(G)$ the maximum size of a clique in G. It is easy to see that $\omega(G) \leq \chi(G)$ for any graph G. A graph G is *perfect* if $\omega(H) = \chi(H)$ for any vertex-induced subgraph H of G. In conjunction with the domination theory in graphs, the following subclasses of chordal graphs have been studied (see [1, 2, 6]). An *n-sun* is a chordal graph with a Hamiltonian cycle $(x_1, y_1, x_2, y_2, \ldots, x_n, y_n, x_1)$ in which each

x_i is of degree exactly two. An SF-chordal (resp., OSF-chordal, 3SF-chordal) graph is a chordal which contains no n-sun with $n \geq 3$ (resp. odd $n \geq 3$, $n = 3$) as an induced subgraph. SF-chordal graphs are also called *strongly chordal* graphs by Farber (see [6]). Strongly chordal graphs include directed path graphs, interval graphs, unit interval graphs, block graphs, and trees. A vertex x is *simple* if $N[y] \subseteq N[z]$ or $N[z] \subseteq N[y]$ for any two vertices $y, z \in N[x]$. Consequently, for any simple vertex x, $N[x]$ is a clique and x has a *maximum neighbor* $m \in N[x]$; i.e., $N[y] \subseteq N[m]$ for any $y \in N[x]$. Farber [6] proved that G is a strongly chordal graph if and only if every vertex-induced subgraph of G has a simple vertex.

THEOREM 5.1. *$\lambda(G) \leq 2\Delta$ for any OSF-chordal graph G with maximum degree Δ.*

Proof. First, $\lambda(G) \leq 2\chi(G^2) - 2$ by Lemma 2.4. By Corollary 3.11 of [2], G^2 is perfect and so $\chi(G^2) = \omega(G^2)$. Since G is OSF-chordal, it is 3SF-chordal. By Theorem 3.8 of [1], $\omega(G^2) = \Delta + 1$. The above inequality and equalities imply that $\lambda(G) \leq 2\Delta$. □

THEOREM 5.2. *$\lambda(G) \leq \Delta + 2\chi(G) - 2$ for any strongly chordal graph G with maximum degree Δ.*

Proof. We shall prove the theorem by induction on $|V(G)|$. The theorem is obvious when $|V(G)| = 1$. Suppose $|V(G)| > 1$. Choose a simple vertex v of G. Since $G - v$ is also strongly chordal, by the induction hypothesis,

$$\lambda(G - v) \leq \Delta(G - v) + 2\chi(G - v) - 2 \leq \Delta + 2\chi(G) - 2.$$

Let f be a $\lambda(G - v)$-$L(2,1)$-labeling of $G - v$. Note that v is adjacent to $\deg(v)$ vertices, which form a clique in G. Let m be the maximum neighbor of v. Since every vertex of distance two from v is adjacent to m, there are $\deg(m) - \deg(v)$ vertices that are of distance two from v. Therefore, there are at most $3\ \deg(v) + \deg(m) - \deg(v) \leq \Delta + 2\omega(G) - 2 = \Delta + 2\chi(G) - 2$ numbers used by f to be avoided by v. Hence there is still at least one number in $[0, \Delta + 2\chi(G) - 2]$ that can be assigned to v in order to extend f into a $(\Delta + 2\chi(G) - 2)$-$L(2,1)$-labeling. □

Although a strongly chordal graph is OSF-chordal, the upper bounds in Theorems 5.1 and 5.2 are incomparable. Theorem 5.2 is a generalization of the result that $\lambda(T) \leq \Delta + 2$ for any nontrivial tree of maximum degree Δ. We conjecture that $\lambda(G) \leq \Delta + \chi(G)$ for any strongly chordal graph G with maximum degree Δ.

6. A polynomial algorithm for λ on trees. For a tree T with maximum degree Δ, Griggs and Yeh [10] proved that $\lambda(T) = \Delta + 1$ or $\Delta + 2$. They also conjectured that it is NP-complete to determine if $\lambda(T) = \Delta + 1$. On the contrary, this section gives a polynomial time algorithm to determine if $\lambda(T) = \Delta + 1$. Although not necessary, the following two preprocessing steps reduce the size of a tree before we apply the algorithm.

First, check if there is a vertex x whose closed neighborhood $N[x]$ contains three or more vertices of degree Δ. If the answer is positive, then $\lambda(T) = \Delta + 2$ by Lemma 2.5.

Next, check if there is a leaf x whose unique neighbor y has degree less than Δ. If there is such a vertex x, then $T - x$ also has maximum degree Δ. By Lemma 2.1 and precisely the same arguments as in the proof of Theorem 4.1 of [10], $\lambda(T - x) \leq \lambda(T) \leq \max\{\lambda(T - x), \deg(x) + 2\} \leq \lambda(T - x)$ and so $\lambda(T) = \lambda(T - x)$. Determining $\lambda(T)$ is then the same as determining $\lambda(T - x)$. Continue this process until any leaf of the tree is adjacent to a vertex of degree Δ.

Regardless of whether we apply the above two steps to reduce the tree size or not, from now on we assume that T' is a tree of at least two vertices and whose maximum degree is Δ. For any fixed positive integer k, the following algorithm determines if T' has a k-$L(2,1)$-labeling or not. We in fact only need to apply the algorithm for $k = \Delta + 1$.

For technical reasons, we may assume that T' is rooted at a leaf r', which is adjacent to r. Let $T = T' - r'$ be rooted at r. We can consider T' as the tree resulting from T by adding a new vertex r' that is adjacent to r only. For any vertex v in T, let $T(v)$ be the subtree of T rooted at v and $T'(v')$ be the tree resulting from $T(v)$ by adding a new vertex v' that is adjacent to v only. $T'(v')$ is considered to be rooted at the leaf v'. Note that $T(r) = T$ and $T'(r') = T'$. Denote

$$S(T(v)) = \{(a,b) : \text{ there is a } k\text{-}L(2,1)\text{-labeling } f \text{ on } T'(v') \text{ with } f(v') = a \text{ and } f(v) = b\}.$$

Note that $\lambda(T) \leq k$ if and only if $S(T(r)) \neq \emptyset$. Now suppose $T(v) - v$ contains s trees $T(v_1), T(v_2), \ldots, T(v_s)$ rooted at $v_1, v_2, \ldots, v_s$, respectively, where each v_i is adjacent to v in $T(v)$. Note that $T(v)$ can be considered as identifying $v_1', v_2', \ldots, v_s'$ to a vertex v on the disjoint union of $T'(v_1'), T'(v_2'), \ldots, T'(v_s')$.

For a system of sets $(A_i)_{i=1}^s \equiv (A_1, A_2, \ldots, A_s)$, a system of distinct representatives (SDR) is an s-tuple $(a_i)_{i=1}^s \equiv (a_1, a_2, \ldots, a_s)$ of s distinct elements such that $a_i \in A_i$ for $1 \leq i \leq s$.

THEOREM 6.1. *$S(T(v)) = \{(a,b) : 0 \leq a \leq k,\ 0 \leq b \leq k,\ |a-b| \geq 2$, and $(A_i)_{i=1}^s$ has an SDR, where $A_i = \{c : c \neq a$ and $(b,c) \in S(T(v_i))\}\}$.*

Proof. Denote by S the set on the right-hand side of the equality in the theorem.

Suppose $(a,b) \in S(T(v))$. There is a k-$L(2,1)$-labeling f of $T'(v')$ such that $f(v') = a$ and $f(v) = b$. Of course, $0 \leq a \leq k$, $0 \leq b \leq k$, and $|a-b| \geq 2$. Let f_i be the function f restricted on $T'(v_i')$ by viewing v_i' the same as v. Then f_i is a k-$L(2,1)$-labeling of $T'(v_i')$ with $f_i(v_i') = f(v) = b$ and $f_i(v_i) = f(v_i) \neq f(v') = a$, i.e., $(b, f(v_i)) \in S(T(v_i))$ and $f(v_i) \in A_i$. Thus $(f(v_i))_{i=1}^s$ is an SDR of $(A_i)_{i=1}^s$. This proves $S(T(v)) \subseteq S$.

On the other hand, suppose $(a,b) \in S$. Then $0 \leq a \leq k$, $0 \leq b \leq k$, $|a-b| \geq 2$, and $(A_i)_{i=1}^s$ has an SDR $(c_i)_{i=1}^s$. Let f_i be a k-$L(2,1)$-labeling of $T'(v_i')$ such that $f_i(v_i') = b$ and $f(v_i) = c_i$. Consider the labeling f of T' defined by $f(x) = f_i(x)$ for $x \in V(T(v_i))$ and $f(v') = a$. It is straightforward to confirm that f is a k-$L(2,1)$-labeling of $T'(v')$ with $f(v') = a$ and $f(v) = b$; i.e., $(a,b) \in S(T(v))$. □

Our algorithm for determining if a tree has a k-$L(2,1)$-labeling recursively applies the above theorem with the initial condition that for any leaf v of T,

$$S(T(v)) = \{(a,b) : 0 \leq a \leq k, 0 \leq b \leq k, |a-b| \geq 2\}.$$

To decide if the tree T' has a k-$L(2,1)$-labeling, we calculate $S(T(v))$ for all vertices v of the tree T. The algorithm starts from the leaves and works toward r. For any vertex v, whose children are $v_1, v_2, \ldots, v_s$, we use $S(T(v_1)), \ldots, S(T(v_s))$ to calculate $S(T(v))$ by Theorem 6.1. More precisely, for any (a,b) with $0 \leq a \leq k$, $0 \leq b \leq k$, $|a-b| \geq 2$, we check if $(a,b) \in S(T(v))$ by the following method. Construct a bipartite graph $G = (X, Y, E)$ with

$$X = \{x_1, x_2, \ldots, x_s\}, \qquad Y = \{0, 1, \ldots, k\},$$

and

$$E = \{(x_i, c) : c \neq a \text{ and } (b,c) \in S(T(v_i))\}.$$

We can use any well-known algorithm to find a maximum matching of the bipartite graph G. Then $(a,b) \in S(T(v))$ if and only if G has a matching of size s. Note that for any vertex v we need to solve the bipartite matching problem $O(k^2)$ times. Therefore, the complexity of the above algorithm is $O(|V(T)|k^2 g(2k))$, where $g(n)$ is the complexity of solving the bipartite matching problem of n vertices. The well-known flow algorithm gives $g(n) = O(n^{2.5})$.

Acknowledgments. The authors wish to extend their gratitude to the referee and to Jerry Griggs for many constructive suggestions for the revision of this paper.

REFERENCES

[1] G. J. CHANG AND G. L. NEMHAUSER (1984), *The k-domination and k-stability problems on sun-free chordal graphs*, SIAM J. Alg. Disc. Meth., 5, pp. 332–345.

[2] ——— (1985), *Covering, packing and generalized perfection*, SIAM J. Alg. Disc. Meth., 6, pp. 109–132.

[3] D. G. CORNEIL, Y. PERL, AND L. K. STEWART (1985), *A linear recognition algorithm for cographs*, SIAM J. Comput., 14, pp. 926–934.

[4] M. B. COZZENS AND F. S. ROBERTS (1982), *T-Colorings of graphs and the channel assignment problem*, Congr. Numer., 35, pp. 191–208.

[5] M. B. COZZENS AND D. I. WANG (1984), *The general channel assignment problem*, Congr. Numer., 41, pp. 115–129.

[6] M. FARBER (1983), *Characterization of strongly chordal graphs*, Discrete Math., 43, pp. 173–189.

[7] Z. FÜREDI, J. R. GRIGGS, AND D. J. KLEITMAN (1989), *Pair labelings with given distance*, SIAM J. Discrete Math., 2, pp. 491–499.

[8] J. GEORGES, D. MAURO, AND M. WHITTLESEY (1994), *Relating path covering to vertex labellings with a condition at distance two*, Discrete Math., 135, pp. 103–111.

[9] M. C. GOLUMBIC (1980), *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York.

[10] J. R. GRIGGS AND R. K. YEH (1992), *Labeling graphs with a condition at distance* 2, SIAM J. Discrete Math., 5, pp. 586–595.

[11] W. K. HALE (1980), *Frequency assignment: Theory and applications*, in Proc. IEEE, 68, pp. 1497–1514.

[12] K. JONAS (1993), *Graph Coloring Analogues With a Condition at Distance Two: $L(2,1)$-Labelings and List λ-Labelings*, Ph.D. thesis, Dept. of Math., University of South Carolina, Columbia, SC.

[13] A. RAYCHAUDHURI (1985), *Intersection Assignment, T-Coloring and Powers of Graphs*, Ph.D. thesis, Dept. of Math., Rutgers University, New Brunswick, NJ.

[14] ———, *Further results on T-coloring and frequency assignment problems*, submitted.

[15] F. S. ROBERTS (1988), private communication to J. R. Griggs.

[16] ——— (1990), *From garbage to rainbows: Generalizations of graph colorings and their applications*, in Proc. of the Sixth International Conference on the Theory and Applications of Graphs, Y. Alavi, G. Chartrand, O. R. Oellermann, and A. J. Schwenk, eds., John Wiley, New York.

[17] ——— (1991), *T-colorings of graphs: Recent results and open problems*, Discrete Math., 93, pp. 229–245.

[18] D. SAKAI (1994), *Labeling chordal graphs: Distance two condition*, SIAM J. Discrete Math., 7, pp. 133–140.

[19] B. TESMAN (1989), *T-Colorings, List T-Colorings, and Set T-Colorings of Graphs*, Ph.D. thesis, Dept. of Math., Rutgers University, New Brunswick, NJ.

[20] M. A. WHITTLESEY, J. P. GEORGES, AND D. W. MAURO (1995), *On the λ-number of Q_n and related graphs*, SIAM J. Discrete Math., 8, pp. 499–506.

[21] R. K. YEH (1990), *Labeling Graphs with a Condition at Distance Two*, Ph.D. thesis, Dept. of Math., University of South Carolina, Columbia, SC.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 317–338, May 1996

014

REALIZING DEGREE SEQUENCES IN PARALLEL*

SRINIVASA R. ARIKATI† AND ANIL MAHESHWARI‡

Abstract. A sequence d of integers is a degree sequence if there exists a (simple) graph G such that the components of d are equal to the degrees of the vertices of G. The graph G is said to be a realization of d. We provide an efficient parallel algorithm to realize d; the algorithm runs in $O(\log n)$ time using $O(n+m)$ CRCW PRAM processors, where n and m are the number of vertices and edges in G. Before our result, it was not known if the problem of realizing d is in NC.

Key words. design and analysis of algorithms, parallel computation, graph algorithms, degree sequence, majorization, PRAM

AMS subject classifications. 68Q22, 68R10

1. Introduction.

1.1. Problem definition. An important problem in graph algorithms is to compute a (simple undirected) graph satisfying the given degree constraints. An integer sequence d of length n is called a *degree sequence* if there exists a graph G on n vertices such that the degrees of its vertices are equal to the components of the sequence d. The graph G is said to be a *realization* of the sequence d. A pair (r, s) of integer sequences is called a *bipartite sequence* if there exists a bipartite graph $H = (X \cup Y, E)$ such that the components of r (respectively, s) are equal to the degrees of the vertices in X (respectively, Y). Degree sequences and bipartite sequences have been extensively studied in graph theory [6, 15, 21, 26]. Because of the strong connections between the structural properties of a graph and the degrees of its vertices, these sequences find significant applications in the areas of communication networks, structural reliability, and stereochemistry (cf. [7, 26]).

1.2. Previous results. Given an integer sequence d, there are two problems of interest: the *decision problem* is to test if d is realizable; the *search problem* is to compute a realization of d. A characterization of degree sequences known as the Erdős–Gallai inequalities [10] results in an efficient sequential algorithm for the decision problem. Another characterization called the Havel–Hakimi characterization (cf. [15]) leads to an efficient sequential algorithm for the search problem. In the case of bipartite sequences, a characterization known as the Gale–Ryser theorem [13, 22, 24] leads to efficient sequential algorithms for the decision as well as the search problems. Recently, degree sequence problems have gained a lot of attention; see for example [2–4, 9, 21, 23, 25, 26].

The Erdős–Gallai inequalities and the Gale–Ryser theorem imply efficient parallel algorithms for the decision problems on degree sequences and bipartite sequences, respectively. Recently, a parallel algorithm for a special case of the search problem, in which the maximum degree is bounded by the square-root of the sum of the degrees, is presented in [9]; it runs in $O(\log^4 n)$ time using $O(n^{10})$ EREW PRAM processors.

* Received by the editors May 18, 1994; accepted for publication (in revised form) August 11, 1995. This work was partially supported by the EEC ESPRIT Basic Research Action No. 7141 (ALCOM II). A part of this paper appeared in 5th *International Symposium on Algorithms and Computation*, China, 1994.

† MPI Informatik, Im Stadtwald, 66123 Saarbrücken, Germany (arikati@mpi-sb.mpg.de).

‡ MPI Informatik, Germany. Current address: School of Computer Science, Carleton University, Ottawa, Ontario K1S 5B6, Canada (maheshwa@chaos.scs.carleton.ca).

Network-flow based proofs [11, 12] give rise to randomized parallel algorithms for the search problems on degree sequences and bipartite sequences.

1.3. Our results. The main contributions of this paper are deterministic parallel algorithms for the search problems on degree sequences and bipartite sequences. Our results are as follows:

- an efficient parallel algorithm for realizing bipartite sequences that runs in $O(\log n)$ time using $O(n)$ EREW PRAM processors, where n is the number of vertices in the realization;
- a new proof of a relation between degree sequences and bipartite sequences;
- an efficient parallel algorithm for realizing degree sequences that runs in $O(\log n)$ time using $O(n+m)$ CRCW PRAM processors, where n and m denote the number of vertices and edges in the realization.

The complexity results of this paper are with respect to the PRAM computational model. For details on the PRAM and NC, see [18, 19]. The *work*, i.e., *time*$\times$*processor* product, of our parallel algorithm for realizing bipartite sequences is $o(n^2)$, whereas there are bipartite graphs that have $\Omega(n^2)$ edges (e.g., a complete bipartite graph on n vertices). The complexity results of this paper are feasible since the graphs computed by our algorithms possess implicit representations; i.e., the graphs can be stored in $O(n)$ space, and the adjacency information between any two vertices can be reported in constant time [27].

Our result for realizing bipartite sequences is based on a nontrivial parallelization of the techniques from the theory of majorization [16, 22]. Our algorithm for realizing a degree sequence d is based on a new proof of a relation between degree sequences and bipartite sequences and it proceeds as follows. From d, we compute an appropriate bipartite sequence (c, c), and then compute a realization H of (c, c). Using the graph H, we compute a symmetric bipartite graph that leads to a realization of d. The computation of the symmetric bipartite graph from H is the crucial step, for which we provide two alternate parallel algorithms: the first one has higher complexity than the second. The latter algorithm exploits the implicit structure of the bipartite graph H computed by our algorithm and thus is efficient. The former algorithm does not assume any structural knowledge of H and can work with any realization of (c, c). It is based on several interesting lemmas, which may be of independent interest in their own right.

1.4. Organization of the paper. The rest of the paper is organized as follows. In §2 we introduce notation and state preliminaries. In §3 we state some of the classical characterizations of degree sequences and present simple algorithms for realizing the degree sequences corresponding to multigraphs and trees. In §4 we prove a relation between degree sequences and bipartite sequences. In §5 we present the required results from the theory of majorization, including an algorithm for computing unit transformations. In §6 we present a parallel algorithm for realizing bipartite sequences. In §7 we provide parallel algorithms for realizing degree sequences.

2. Preliminaries.

2.1. Basic definitions. In a *multigraph* $G = (V, E)$, V is a set of vertices and E is a multiset of edges (multiple edges may exist between two vertices but no self-loops). By a *graph* $G = (V, E)$, we mean a simple graph—without multiple edges and self-loops. A bipartite graph H with the bipartition $X \cup Y$ is denoted by $H = (X \cup Y, E)$. In a multigraph $G = (V, E)$, $d_G(v)$ denotes the degree of a vertex v and $N_G(v)$ denotes the multiset of the neighbors of v (we omit the subscript, if no confusion

arises). Similarly, $N_G(U)$ is defined as the union of the neighbors of the vertices of $U \subseteq V$. By definition, G is a graph if and only if the following hold for all $v \in V$: (i) $v \notin N_G(v)$ and (ii) $N_G(v)$ is a set. If (u, v) is an edge of a graph G, we say that $(u, v) \in G$. Throughout, by a sequence we mean a sequence of nonnegative integers.

2.2. Graph matching. A *matching* M in a graph $G = (V, E)$ is a collection of edges such that no two edges of M are incident at a common vertex. The size of M, denoted by $|M|$, is the number of edges in it. M is called a *perfect matching* if it matches all vertices of G. M is called a *maximum matching* if it has maximum size among all matchings. M is called a *maximal matching* if no other matching properly contains M. We will need the following theorem due to Hall (cf. [21]).

THEOREM 2.1 (the Hall theorem). *Let $H = (X \cup Y, E)$ be a bipartite graph. Then H has a matching that matches all vertices of X, if and only if $|N(A)| \geq |A|$ for every $A \subseteq X$.*

We need the following two lemmas; the first lemma can be proved easily.

LEMMA 2.2. *Let M (respectively, M') be a maximal (respectively, maximum) matching in a graph G. Then $|M| \geq \frac{1}{2}|M'|$.*

LEMMA 2.3. *Let $H = (X \cup Y, E)$ be a bipartite graph such that* (i) *$d(x) \geq 1$ for all $x \in X$ and* (ii) *the inequality $d(x) \geq d(y)$ holds for every edge (x, y) of H. Then H has a matching that matches all vertices of X.*

Proof. We show that H satisfies the sufficiency part of the Hall theorem. We use induction on $|A|$, where $A \subseteq X$. Since $d(x) \geq 1$ for all $x \in X$, the basis case, $|A| = 1$, follows. Consider now the case that $|A| = k$, where $k \geq 2$. We need to prove that $|N(A)| \geq k$. Assume the contrary, namely that $|N(A)| < k$. Pick any vertex $z \in A$ and put $A' = A - z$. By induction, $|N(A')| \geq k - 1$. Since $N(A') \subseteq N(A)$, it follows $(A') = N(A)$ and $|N(A')| = |N(A)| = k - 1$. Consider now the subgraph $H' = (A' \cup N(A'), E')$ of H. By induction, H' satisfies the sufficiency condition of the Hall theorem and hence has a perfect matching. Let the edges of the perfect matching be $(x_1, y_1), (x_2, y_2), \ldots, (x_{k-1}, y_{k-1})$. Using condition (ii) of the lemma, we obtain $\sum_{x \in A'} d(x) = \sum_{i=1}^{k-1} d(x_i) \geq \sum_{i=1}^{k-1} d(y_i) = \sum_{y \in N(A')} d(y)$. Since $d(z) \geq 1$ and $N(A) = N(A')$, we have $\sum_{x \in A} d(x) > \sum_{y \in N(A)} d(y)$, which contradicts $\sum_{x \in A} d(x) \leq \sum_{y \in N(A)} d(y)$; the latter inequality holds for any bipartite graph because every edge incident to a vertex in A contributes a 1 to both sides of the inequality. This completes the induction step and hence the lemma. □

2.3. Digraphs. In a digraph $D = (V, E)$, E is the set of arcs (directed edges); the arc from u to v will be denoted by the ordered pair (u, v). The indegree (respectively, outdegree) of a vertex v, denoted by $d_D^-(v)$ (respectively, $d_D^+(v)$), is the number of arcs into (respectively, from) v. Call D *symmetric* if it has only symmetric arcs: (u, v) is an arc if and only if (v, u) is an arc. We will require the following lemma.

LEMMA 2.4 (see [12]). *Let $\tilde{D} = (V, \tilde{E})$ be a digraph such that indegree of each vertex v equals its outdegree, i.e., $d_{\tilde{D}}^-(v) = d_{\tilde{D}}^+(v) = d(v)$, and that $\sum_{v \in V} d_{\tilde{D}}^+(v)$ is even. Then there exists a symmetric digraph $D = (V, E)$ such that $d_D^-(v) = d_D^+(v) = d(v)$.*

Proof. If $\tilde{D}$ is symmetric, then take $D := \tilde{D}$. Assume that $\tilde{D}$ is not symmetric and let $\hat{D} = (V, \hat{E})$ be the "asymmetric part" of $\tilde{D}$: $(u, v) \in \hat{E}$ iff $(u, v) \in \tilde{E}$ and $(v, u) \notin \tilde{E}$. Observe that $d_{\hat{D}}^-(v) = d_{\hat{D}}^+(v)$ for all $v \in V$. Define a *trail* to be a sequence of (not necessarily distinct) vertices $v_1, \ldots, v_k, v_1$ such that $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k), (v_k, v_1)$ are distinct arcs of $\hat{D}$. Call a trail *even* if it consists of an even number of edges, otherwise it is an *odd* trail.

Assume that $\hat{D}$ has an even trail, say $v_1, v_2, \ldots, v_{2k}, v_1$. Change $\tilde{D}$ as follows: delete the arcs (v_2, v_3), (v_4, v_5), ..., (v_{2k-2}, v_{2k-1}), (v_{2k}, v_1) and add the arcs (v_2, v_1), (v_4, v_3), ..., (v_{2k-2}, v_{2k-3}), (v_{2k}, v_{2k-1}). Notice that this process does not create any multiple arcs or self-loops in $\tilde{D}$. Further, the hypothesis of the lemma is maintained. We repeat the above-mentioned process until $\hat{D}$ contains no even trails. Then, it follows that $\sum_{v \in V} d^+_{\hat{D}}(v)$ is even as $\sum_{v \in V} d^+_{\tilde{D}}(v)$ is even. Further, using $d^-_{\hat{D}}(v) = d^+_{\hat{D}}(v)$ for $v \in V$, we can decompose the arcs of $\hat{D}$ into odd directed cycles. There are an even number of such cycles. Moreover, any two such cycles must be vertex-disjoint, otherwise they create an even trail. Let $u_0, u_1, \ldots, u_{2k}, u_0$ and $v_0, v_1, \ldots, v_{2\ell}, v_0$ be any two odd cycles in $\hat{G}$. The fact that $(u_0, v_0) \notin \hat{D}$ implies that either both (u_0, v_0) and (v_0, u_0) are arcs in $\tilde{D}$ or none is an arc in $\tilde{D}$. We distinguish between these two cases.

Case 1. Both (u_0, v_0) and (v_0, u_0) are arcs in $\tilde{D}$. Change $\tilde{D}$ as follows: delete the arcs (u_0, v_0), (v_0, u_0), (u_1, u_2), (u_3, u_4), ..., (u_{2k-1}, u_{2k}) and (v_1, v_2), (v_3, v_4), ..., $(v_{2\ell-1}, v_{2\ell})$; add the arcs (u_2, u_1), (u_4, u_3), ..., (u_{2k}, u_{2k-1}) and (v_2, v_1), (v_4, v_3), ..., $(v_{2\ell}, v_{2\ell-1})$.

Case 2. None of (u_0, v_0) and (v_0, u_0) is an arc in $\tilde{D}$. Change $\tilde{D}$ as follows: delete the arcs (u_0, u_1), (u_2, u_3), ..., (u_{2k}, u_0), and (v_0, v_1), (v_2, v_3), ..., $(v_{2\ell}, v_0)$; add the arcs (u_0, v_0), (v_0, u_0), (u_2, u_1), (u_4, u_3), ..., (u_{2k}, u_{2k-1}) and (v_2, v_1), (v_4, v_3), ..., $(v_{2\ell}, v_{2\ell-1})$.

In each case no multiple arcs or self-loops are created, and the number of odd cycles in $\tilde{D}$ decreases. Eventually, $\tilde{D}$ contains no odd cycles and, hence, it becomes symmetric. The proof is completed by taking $D := \tilde{D}$. □

2.4. Parallel techniques. The complexity results of this paper are with respect to the PRAM. This is the synchronous parallel model in which all processors share a common memory. In this paper, we need the following techniques previously developed in parallel computing: Euler tour in a graph [5], merging and cross-ranking [14], sorting [8], and maximal matching in a graph [17]. For other techniques such as parallel prefix and list ranking, see [18, 19].

Consider a sequence of n elements $\{x_1, x_2, \ldots, x_n\}$ drawn from a set S with a binary associative operation $*$. The *prefix sums* of this sequence are the n partial sums (or products) defined by $s_i = x_1 * x_2 * \ldots x_i$, $1 \leq i \leq n$. Consider a linked list L of n nodes whose order is specified by an array S such that $S(i)$ contains a pointer to the node following node i on L for $1 \leq i \leq n$. We assume $S(i) = 0$ when i is the end of the list. The *list ranking* problem is to determine the distance of each node i from the end of the list. The *rank* of an element x in a given sequence X is the number of elements of X that are less than or equal to x. Let A and B be two sorted sequences. The *cross-ranking* problem is to find the rank of each element of A in B and vice-versa.

3. Characterizations and algorithmic aspects.

3.1. Multigraphs. Realizability problems, in general, tend to be simpler if multiple edges are allowed. We show that this is the case in parallel computation too. We first discuss realizing degree sequences of bipartite multigraphs and then show how to reduce the general case to the bipartite case. Recall that in a multigraph, multiple edges may exist between a pair of vertices but self-loops are not allowed.

Let (r, s) be a pair of sequences where $r = (r_1, \ldots, r_m)$ and $s = (s_1, \ldots, s_n)$. Our problem is to compute a bipartite multigraph $H = (X \cup Y, E)$ satisfying the degree constraints r and s. It is easy to prove that H exists iff $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} s_j$. To

realize (r, s) in parallel, test if $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} s_j$ and stop if the test fails. Then, compute the prefix sums of r and s and store them in the arrays R and S, respectively. Cross-rank S in R using the algorithm of [14]. Connect y_j to all the corresponding x_i's using the required number of multiple edges.

We now discuss the general case. Given a sequence d, the problem is to compute a multigraph with degree sequence d. The following lemma characterizes d [6, 15, 21]. The proof given below results in a simple parallel algorithm.

LEMMA 3.1. *The sequence $d = (d_1, \ldots, d_n)$, where $d_1 = \max(d)$, is the degree sequence of a multigraph if and only if $\sum_{i=1}^{n} d_i$ is even and $d_1 \leq \sum_{i=2}^{n} d_i$.*

Proof. We prove the sufficiency part, the other part being trivial. Sort the sequence d into nonincreasing order; i.e., let $d_1 \geq d_2 \geq \cdots \geq d_n$. Let $v_1, v_2, \ldots, v_n$ be the vertices of the multigraph G to be computed. Put $m = \frac{1}{2}\sum_{i=1}^{n} d_i$ and let p be the index such that $\sum_{i=1}^{p} d_i \leq m < \sum_{i=p+1}^{n} d_i$. We distinguish between two cases.

Case 1. $\sum_{i=1}^{p} d_i = m$. Define sequences r and s by $r = (d_1, d_2, \ldots, d_p)$ and $s = (d_{p+1}, \ldots, d_n)$. Then r and s have the same component sum and thus (r, s) can be realized, using the procedure given above, as a bipartite multigraph $G = (X \cup Y, E)$, where $X = \{v_1, \ldots, v_p\}$ and $Y = \{v_{p+1}, \ldots, v_n\}$.

Case 2. $\sum_{i=1}^{p} d_i < m$. Put $k = \sum_{i=p+1}^{n} d_i - m$ and define sequences r and s as follows: $r = (d_1 - k, d_2, \ldots, d_p, d_{p+1} - k)$ and $s = (d_{p+2}, \ldots, d_n)$. Then r and s have the same component sum $(= m-k)$ and thus (r, s) can be realized, using the procedure given above, as a bipartite multigraph $H = (X \cup Y, E)$, where $X = \{v_1, \ldots, v_{p+1}\}$ and $Y = \{v_{p+2}, \ldots, v_n\}$. By adding k multiple edges between v_1 and v_{p+1} in H we get the required multigraph G. □

3.2. Trees. We now discuss the degree sequences of trees. The following characterization of such sequences is well known [6, 15, 21]. We will present a proof that leads to a simple parallel algorithm to realize these sequences.

LEMMA 3.2. *The sequence $d = (d_1, \ldots, d_n)$ is the degree sequence of a tree iff all d_i's are positive and $\sum_{i=1}^{n} d_i = 2n - 2$.*

Proof. The necessity part is trivial and we prove the other part. Sort the sequence d into nonincreasing order; denote the resulting sequence also by d. Let $v_1, v_2, \ldots, v_n$ be the vertices of the tree G to be computed. If $d_1 = 2$ then G is a path and we are done. So assume that $d_1 \geq 3$ and let k be the largest index such that $d_k \geq 3$. Put $m = \sum_{i=1}^{k}(d_i - 2)$. Let A be the set of d_i's that are equal to 1. The following claim will be proved after we describe the computation of G.

Claim. $|A| = m + 2$.

First we compute a path consisting of vertices $v_{n-1}, v_1, v_2, \ldots, v_{n-m-2}, v_n$.[1] Delete v_n and v_{n-1} from A. Then we compute "stars" using A and $v_1, v_2, \ldots, v_k$ as follows: connect the first $d_1 - 2$ vertices of A to v_1, connect the next $d_2 - 2$ vertices of A to v_2, ..., and connect the remaining $d_k - 2$ vertices of A to v_k. This completes the computation of G.

We now prove the above claim. Observe that $\sum_{i=1}^{k} d_i = m+2k$ and that there are $n-k-|A|$ d_i's that are equal to 2. So $2n-2 = \sum_{i=1}^{n} d_i = m+2k+2(n-k-|A|)+|A|$. The claim follows by rearranging the terms. □

3.3. Graphs. We now discuss degree sequences of graphs. Let d be an integer sequence of length n, where $n > d_1 \geq d_2 \geq \cdots \geq d_n \geq 0$. The proofs of the following results may be found, e.g., in [6, 15, 21].

[1] Observe that $d_{n-1} = d_n = 1$.

The Erdős–Gallai inequalities (EGI). The sequence d is realizable if and only if $\sum_{i=1}^{n} d_i$ is even and $\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$ for $k = 1, 2, \ldots, n$.

The Havel–Hakimi characterization. The sequence d is realizable iff the numbers $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n$ are realizable.

Using EGI, we can test in linear time if d is realizable. We can use the second characterization to derive an efficient sequential algorithm to compute a realization of d.

In parallel computation, the inequalities of EGI can be tested optimally, and this implies an optimal parallel algorithm to test if d is a degree sequence. As for the problem of computing a realization of d, a proof of the EGI using network flows is given in [12] and this proof results in a randomized parallel algorithm. The proof consists of two steps.

Step 1. Define a network with edge capacities in unary and solve the maximum flow problem on this network; then, construct a digraph D.

Step 2. Obtain a symmetric digraph from D.

Results of [20] imply a randomized parallel algorithm for Step 1. Based on the proof of Lemma 2.4, Step 2 can be performed in NC.

4. Degree sequences and bipartite sequences. We study in this section a relation between degree sequences and bipartite sequences. The main result of this section is a new proof of a theorem given in [25]. The proof presented in [25] is via a cycle of eight implications and results in an inherently sequential algorithm to compute realizations of degree sequences, whereas our proof is simple and helps us to design a parallel algorithm.

Throughout this section, d denotes an integer sequence of length n, where $n > d_1 \geq d_2 \geq \cdots \geq d_n \geq 0$. Let $\mu = \max\{k : d_k \geq k\}$. Define a new sequence $c = (c_1, \ldots, c_n)$, where

$$c_i = \begin{cases} d_i + 1 & \text{if } i \leq \mu, \\ d_i & \text{otherwise.} \end{cases}$$

Observe that $c_i = d_i + 1 \geq \mu + 1$ for $1 \leq i \leq \mu$ and $c_j = d_j \leq \mu$ for $\mu + 1 \leq j \leq n$.

THEOREM 4.1. *The sequence d is a degree sequence iff (c, c) is a bipartite sequence.*

Before we prove this theorem, we remark that the "+1" is required in the definition of c, since there are sequences d such that (d, d) is a bipartite sequence but d is not a degree sequence; for example, take $d = (4, 2, 2, 2)$. The proof is based on the following lemmas.

LEMMA 4.2. *If d is a degree sequence then (c, c) is a bipartite degree sequence.*

Proof. Let $G = (V, E)$ be a realization of d. We obtain a bipartite realization $H = (X \cup Y, E')$ of (c, c) as follows: X and Y are two copies of V; if $(v_i, v_j) \in G$ then $(x_i, y_j), (x_j, y_i) \in H$; further, $(x_i, y_i) \in H$ for all $1 \leq i \leq \mu$. □

We need a few definitions before presenting the next lemmas. Let $H = (X \cup Y, E)$ be a realization of (c, c). A vertex x_i (or y_i) is called a *high-degree* vertex if $1 \leq i \leq \mu$, otherwise it is a *low-degree* vertex. An edge (x_i, y_i) is called a *high-degree* edge if $1 \leq i \leq \mu$. Similarly, an edge (x_j, y_j) is called a *low-degree* edge if $\mu + 1 \leq j \leq n$. Edges of the form (x_i, y_j), where $i \neq j$, are neither high-degree edges nor low-degree edges. High-degree and low-degree edges play a very important role in our algorithms.

A pair of edges (x_α, y_β) and (x_γ, y_δ) form an *exchange pair* if $(x_\alpha, y_\delta), (x_\gamma, y_\beta) \notin H$ (see Figure 1). An *exchange* on the edges (x_α, y_β) and (x_γ, y_δ) consists of deleting (x_α, y_β) and (x_γ, y_δ) and inserting (x_α, y_δ) and (x_γ, y_β). The following two lemmas

imply that, given any realization H of (c, c), we can always obtain another realization H' such that H' contains all high-degree edges and no low-degree edges.

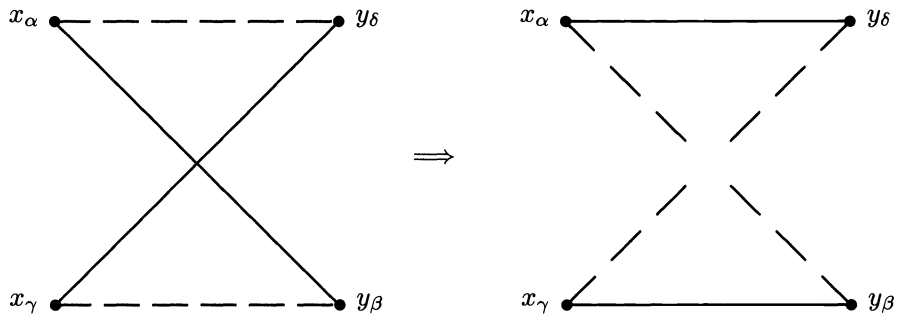


FIG. 1. *An exchange operation. A solid line indicates the presence of an edge and a dashed line its absence.*

LEMMA 4.3. *Let H be a realization of (c, c) such that $(x_i, y_i) \notin H$ for some $1 \leq i \leq \mu$. Then there exist k and ℓ such that (x_i, y_k) and (x_ℓ, y_i) form an exchange pair, where $k > \mu$ and $k \neq \ell$.*

Proof. Since $c_i \geq \mu+1$ there exists a k such that $k > \mu$ and $(x_i, y_k) \in H$. Further, $c_k \leq \mu$ and $c_i \geq \mu + 1$ imply that there exists an $\ell \neq k$ such that $(x_\ell, y_i) \in H$ and $(x_\ell, y_k) \notin H$. □

LEMMA 4.4. *Let H be any realization of (c, c) such that $(x_j, y_j) \in H$ for some $j > \mu$. Then there exist k and ℓ such that (x_j, y_j) and (x_ℓ, y_k) form an exchange pair, where $k \leq \mu$ and $k \neq \ell$.*

Proof. Since $c_j \leq \mu$ there exists a k such that $k \leq \mu$ and $(x_j, y_k) \notin H$. Further, $c_k \geq \mu + 1$ implies that there exists an $\ell \neq k$ such that $(x_\ell, y_k) \in H$ and $(x_\ell, y_j) \notin H$. □

LEMMA 4.5. *If (c, c) is a bipartite degree sequence, then d is a degree sequence.*

Proof. Let H be any realization of (c, c). Assume that $(x_i, y_i) \notin H$ for some $1 \leq i \leq \mu$. Let k and ℓ be as defined in Lemma 4.3. Perform an exchange, by deleting the edges (x_i, y_k) and (x_ℓ, y_i) and adding the edges (x_i, y_i) and (x_ℓ, y_k). Observe that this process does not destroy any existing high-degree edges in H. We repeat this process until H contains all high-degree edges. Assume now that H contains a low-degree edge, say (x_j, y_j) for some $j > \mu$. Let k and ℓ be as defined in Lemma 4.4. Delete the edges (x_j, y_j) and (x_ℓ, y_k) and add the edges (x_j, y_k) and (x_ℓ, y_j). Observe that this process does not destroy any existing high-degree edges and does not create any new low-degree edges in H. We repeat this process until H contains no low-degree edges.

Define a digraph $\tilde{D} = (V, \tilde{E})$ on the vertex set $V = \{v_1, \ldots, v_n\}$, where $(v_i, v_j) \in \tilde{D}$ iff $(x_i, y_j) \in H$ and $(x_j, y_i) \notin H$. Then $d^-_{\tilde{D}}(v_i) = d^+_{\tilde{D}}(v_i) = d_i$ for all $1 \leq i \leq n$. By Lemma 2.4, we obtain a symmetric digraph $D = (V, E)$ such that $d^-_D(v_i) = d^+_D(v_i) = d_i$ for all i. Obtain an undirected graph G from D by replacing the two symmetric arcs (v_i, v_j) and (v_j, v_i) with the edge (v_i, v_j). Then G is a realization of d. □

Lemmas 4.2 and 4.5 imply Theorem 4.1. It is fairly easy to see that the proof of Lemma 4.5 implies a simple sequential algorithm to realize d from any realization of (c, c). In §7, parallel algorithms to achieve the same objective will be presented.

5. Majorization and unit transformations. Throughout this section, let $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ be sequences of length n, where $a_1 \geq a_2 \geq \cdots \geq a_n \geq 0$ and $b_1 \geq b_2 \geq \cdots \geq b_n \geq 0$.

5.1. Majorization. Majorization has been studied, over several decades, in the theory of inequalities [22, 16]. It captures the intuitive notion that the components of a vector are "less nearly equal" than are the components of another vector. Formally, we say that a *majorizes* b, denoted by $a \succeq b$, if $\sum_{i=1}^{k} a_i \geq \sum_{i=1}^{k} b_i$ for $k = 1, ..., n$, with equality for $k = n$. For example, $(4, 2, 1, 0) \succeq (2, 2, 2, 1)$.

Majorization was used by economists in measuring inequality of incomes and in studying the principle of transfers (cf. [22]). If $a_i \geq a_j + 2$ for some i and j, we say that the sequence $c = (c_1, \ldots, c_n)$, defined by $c_i = a_i - 1, c_j = a_j + 1$, and $c_k = a_k$ for $k \neq i, j$, is obtained from a by a *unit transformation* from i to j. Clearly, $a \succeq c$. A classical result, known as the Muirhead lemma (cf. [22]), states that the converse is also true: if $a \succeq b$ then a can be transformed to b by performing a finite number of (successive) unit transformations on a. The following lemma presents the details.

LEMMA 5.1. *Suppose that* $a \succeq b$. *Define a sequence* $\delta = (\delta_1, \ldots, \delta_n)$ *by* $\delta_i = \max\{0, (a_i - b_i)\}$, *and define* $\Delta(a, b) = \sum_{i=1}^{n} \delta_i$. *Then* a *can be transformed to* b *by performing* $\Delta(a, b)$ *unit transformations. Further,* $\Delta(a, b)$ *equals the minimum number of unit transformations required to transform* a *to* b.

Proof. If $a = b$ then $\Delta(a, b) = 0$. Assume that $a \neq b$ and let i be the smallest index such that $a_i \neq b_i$. Observe that $a_i > b_i$, since $a \succeq b$. Let $j > i$ be the smallest index such that $a_j < b_j$. Define c to be the sequence obtained from a by performing a unit transformation from i to j. Clearly, $a \succeq c \succeq b$. By repeating this process on the sequence c, we can obtain b.

To prove the second part of the lemma, observe that if c is any sequence obtained from a by a unit transformation, then $\Delta(c, b) \geq \Delta(a, b) - 1$. □

5.2. An algorithm for computing unit transformations. In this subsection, we discuss the computation of unit transformations that are required to transform a to b. The basic idea is to compute the numbers $t(i, j)$, for $1 \leq i, j \leq n$, so that a can be transformed to b by performing $t(i, j)$ unit transformations from position i to position j. We give an example to illustrate the idea.

Example 1. Let $a = (15, 12, 9, 8, 5, 4, 4, 0)$ and $b = (11, 10, 9, 9, 9, 3, 3, 3)$. Put $c = a - b = (4, 2, 0, -1, -4, 1, 1, -3)$ and define an array P (respectively, M) whose components are given by $P_i = \max\{0, c_i\}$ (respectively, $M_i = \max\{0, -c_i\}$) for $1 \leq i \leq 8$; i.e., $P = (4, 2, 0, 0, 0, 1, 1, 0)$ and $M = (0, 0, 0, 1, 4, 0, 0, 3)$. Now, P_1 and M_4 are the left-most positive components of P and M, respectively. As $M_4 = 1 < P_1$, we subtract 1 from P_1 and M_4 and add 1 to $t(1, 4)$. Then P and M become (3,2,0,0,0,1,1) and (0,0,0,0,4,0,0,3), respectively. Now P_1 and M_5 are the left-most positive components of P and M and, as $P_1 = 3 < M_5$, we subtract 3 from P_1 and M_5 and add 3 to $t(1, 5)$. Then P and M become (0,2,0,0,0,1,1) and (0,0,0,0,1,0,0,3), respectively. We continue this process until all the components of P and M become 0's, and we get $t(1, 4) = 1$, $t(1, 5) = 3$, $t(2, 5) = t(2, 8) = t(6, 8) = t(7, 8) = 1$. □

A simple linear-time sequential algorithm, based on the procedure explained in the above example, for computing unit transformations is as follows. Compute the arrays P and M and then scan both the arrays from left to right, starting at the position $i = 1$ in P and $j = 1$ in M. In each step compute the appropriate $t(i, j)$ and either increment i or j. Since the arrays P and M are scanned only once and the number of reported $t(i, j)$'s is linear, the algorithm runs in linear time.

A parallel implementation of this algorithm is presented in Algorithm 1. In step 2 of the algorithm, $P'[i] := \sum_{k=1}^{i} P[i]$ for $1 \leq i \leq n$. Moreover, the arrays P' and M' are obtained in the sorted order. After cross-ranking the elements of P' in M' and vice-versa, we know precisely the appropriate $t(i,j)$'s for each value of i and j. For each value of i we can store the $t(i,j)$'s in an array, with respect to increasing j. Alternatively, we can store the $t(i,j)$'s in an $n * n$ matrix, without initializing the matrix, by a standard method (see [1]).

1. Compute $c_i := a_i - b_i$ for $1 \leq i \leq n$ and the arrays P and M.
2. Compute prefix sums of P and M and store them in the arrays P' and M', respectively.
3. Cross-rank the arrays P' and M' by the algorithm of [14] and then compute $t(i,j)$'s.

ALGORITHM 1. *An algorithm for computing unit transformations.*

THEOREM 5.2. *Let a and b be sequences of length n such that $a \succeq b$. Algorithm 1 computes the numbers $t(i,j)$, for $1 \leq i,j \leq n$, such that a can be transformed to b by performing $t(i,j)$ unit transformations from the position i to j in a. The algorithm runs in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors.*

Proof. The proof of correctness is straightforward. To analyze the complexity, note that the prefix sums and the cross-ranking of two sorted arrays can be computed in $O(\log n)$ time using $O(n/\log n)$ processors. □

5.3. Properties of unit transformations. In this subsection we present properties of the numbers $t(i,j)$ computed by Algorithm 1. These properties will be used to design a parallel algorithm for realizing bipartite sequences.

Observe $t(i,j) = 0$ whenever $i \geq j$ or $a_j \geq b_j$. We need a few definitions to state additional properties of $t(i,j)$'s. For $1 \leq i \leq n$, define $A(i)$ to be the set of all j such that Algorithm 1 reports a unit transformation from i to j; i.e., $A(i) = \{j : t(i,j) \neq 0\}$ (see Example 2). Observe that $A(i) = \emptyset$ if and only if $a_i \leq b_i$. Similarly, define $B(j) = \{i : t(i,j) \neq 0\}$, and note that $B(j) = \emptyset$ iff $a_j \geq b_j$. Furthermore, if $k \in A(i)$ (respectively, $B(j)$), then $k > i$ (respectively, $k < j$) and $a_k < b_k$ (respectively, $a_k > b_k$). The elements of $A(i)$ and $B(j)$ will always be listed in the increasing order.

Example 2. In Example 1,

$A(1) = \{4,5\}$, $A(2) = \{5,8\}$, $A(6) = A(7) = \{8\}$;
$B(4) = \{1\}$, $B(5) = \{1,2\}$, $B(8) = \{2,6,7\}$;
all other $A(i)$'s and $B(j)$'s are $\emptyset$. □

LEMMA 5.3.

1. *If $A(i) \neq \emptyset$, then $a_i = b_i + \sum_{j \in A(i)} t(i,j)$.*
2. *If $B(j) \neq \emptyset$, then $b_j = a_j + \sum_{i \in B(j)} t(i,j)$.*

Proof. The proof follows from the definition of $t(i,j)$'s in Algorithm 1. □

Let $\alpha(i) = \min\{A(i)\}$.[2] For $1 \leq i,j \leq n$, define (see Example 3)

$$\beta(i,j) = \begin{cases} 0 & \text{if } i > j \text{ or } a_j \geq b_j, \\ a_j + \sum_{k \in B(j), k \geq i+1} t(k,j) & \text{otherwise.} \end{cases}$$

For $1 \leq i \leq n$, define

$$\gamma(i) = \begin{cases} \beta(i+1, \alpha(i)) & \text{if } A(i) \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

[2] By definition, $\max(\emptyset) = 0$ and $\min(\emptyset) = \infty$.

Example 3. In Example 2,

$\alpha(1) = 4$, $\alpha(2) = 5$, $\alpha(6) = \alpha(7) = 8$;
$\beta(3,5) = \beta(4,5) = \beta(5,5) = a_5 = 5$, $\beta(1,5) = a_5 + t(2,5) = 5 + 1 = 6$;
$\gamma(1) = \beta(2, \alpha(1)) = \beta(2,4) = \beta(3,4) = \beta(4,4) = a_4 = 8$. □

LEMMA 5.4. *Let j be an integer such that the elements of $B(j)$ are $i_1, i_2, \ldots, i_k$, where $k \geq 2$. Then $A(i_q) = \{j\}$ for all $2 \leq q \leq k-1$, and $\alpha(i_p) = j$ for all $2 \leq p \leq k$.*

Proof. The proof follows from definitions. □

LEMMA 5.5. *Let j be an integer such that the elements of $B(j)$ are $i_1, i_2, \ldots, i_k$, where $k \geq 2$. Then $\gamma(i_k) = a_j$ and $\gamma(i_p) = a_j + \sum_{q=p+1}^{k} t(i_q, j)$ for all $2 \leq p \leq k-1$. Furthermore, if $\alpha(i_1) = j$, then $\gamma(i_1) = \gamma(i_2) + t(i_2, j)$.*

Proof. By Lemma 5.4, $\gamma(i_k) = j$. So $\gamma(i_k) = \beta(i_k + 1, j)) = a_j$; the second equality follows from the fact that $t(\ell, j) = 0$ for $\ell \geq i_k + 1$. Consider now any p such that $2 \leq p \leq k-1$. We have

$$\begin{aligned}\gamma(i_p) &= \beta(i_{p+1} + 1, \alpha(i_p)) \\ &= \beta(i_{p+1} + 1, j) \qquad \text{(by Lemma 5.4)} \\ &= a_j + \sum_{q=i_{p+1}}^{k} t(i_q, j).\end{aligned}$$

The proof of the remaining part is similar. □

6. Realizing bipartite sequences. We present in this section a parallel algorithm for computing a bipartite graph that realizes a given pair of sequences. Throughout this section, $X = \{x_1, x_2, \ldots, x_m\}$, $Y = \{y_1, y_2, \ldots, y_n\}$, and r and s denote two nonnegative integer sequences, where $n \geq r_1 \geq r_2 \geq \cdots \geq r_m \geq 0$ and $m \geq s_1 \geq s_2 \geq \cdots \geq s_n \geq 0$. Given the pair (r, s), the problem is to compute a bipartite graph $H = (X, Y, E)$, if it exists, such that $d(x_i) = r_i$ and $d(y_j) = s_j$.

6.1. The characterization. The following theorem, known as the Gale–Ryser theorem, characterizes bipartite sequences [13, 24, 11, 22]. Before stating the theorem, we require a definition. The *conjugate sequence* of r, denoted by $r^\star = (r_1^\star, \ldots, r_n^\star)$, is defined as $r_k^\star = |\{i : r_i \geq k\}|$ for $k = 1, \ldots, n$. Both r and $r^\star$ may be visualized by means of a (0,1)-matrix of size $m \times n$ in which the ith row contains r_i 1's left-justified; then $r_k^\star$ is the number of 1's in the kth column.

THEOREM 6.1 (the Gale–Ryser theorem). *The pair (r, s) is a bipartite sequence iff $r^\star \succeq s$.*

Theorem 6.1 suggests a simple parallel algorithm to decide if (r, s) is a bipartite sequence. Moreover, a network-flow based proof of this theorem (cf. [11]), combined with the results of [20], leads to a randomized parallel algorithm to realize (r, s).

6.2. An algorithm. We present in this section a parallel algorithm for computing a bipartite graph $H = (X \cup Y, E)$ that realizes the pair (r, s). It follows from the Gale–Ryser theorem that the pair $(r, r^\star)$ is a bipartite sequence. In the corresponding realization $F = (X, Y, E')$, the neighbors of y_j are $x_1, \ldots, x_p$, where $p = r_j^\star$. Represent $N_F(y_j)$ by the interval $[1, \ldots, p]$ of integers. Our algorithm is based on the following lemma.

LEMMA 6.2. *Suppose that (r, s) is a bipartite sequence. If the sequence t is obtained from s by a unit transformation, then (r, t) is also a bipartite sequence.*

Proof. Let $H = (X, Y, E)$ be a realization of (r, s) and t be obtained from s by a unit transformation from i to j. Since $s_i \geq s_j + 2$, there exists a neighbor, say x_k,

of y_i that is not a neighbor of y_j. Let H' be obtained from H by deleting the edge (x_k, y_i) and adding the edge (x_k, y_j). Then H' is a realization of (r, t). □

We say that the graph H' constructed in the above proof is obtained from H by a *transfer of a neighbor* between x_i and x_j. The main steps in our algorithm for computing a realization of (r, s) are the following steps.

Step 1. Compute $r^\star$. Test if $r^\star \succeq s$. If not, declare that *(r,s) is not a bipartite sequence* and STOP.

Step 2. Obtain the realization $F = (X, Y, E')$ of $(r, r^\star)$ by computing $N_F(y_j) = [1, \ldots, r_j^\star]$.

Step 3. Transfer *appropriate neighbors* among the vertices of Y in F to obtain H.

The parallel implementation of Step 3, which is the difficult step, is based on the properties of the unit transformations (§5.3) required to transform the sequence $r^\star$ to s. For convenience of notation, we use the sequences a and b in place of $r^\star$ and s, respectively. Apply Algorithm 1 to the pair (a, b) and let $t(i, j)$, $1 \leq i, j \leq n$, be the unit transformations computed by the algorithm. Recall from §5.3 the definitions of $A(i)$, $B(j)$, $\alpha(i)$, $\beta(i, j)$, and $\gamma(i)$. Consider an i such that $A(i) \neq \emptyset$. We define intervals $T(i, j)$, where $j \in A(i)$, as follows. In the realization F of (r, a), all vertices x_p such that $p \in T(i, j)$ will be "transferred" from y_i to y_j to obtain the required bipartite graph H. Let the elements of $A(i)$ (in the increasing order) be $j_1 = \alpha(i), j_2, \ldots, j_k$. See Figure 2 and define

$$T(i, j_1) = [\gamma(i) + 1, \ldots, \gamma(i) + t(i, j_1)] \text{ and}$$

$$T(i, j_p) = \left[a_i - \sum_{q=2}^{p} t(i, j_q) + 1, \ldots, a_i - \sum_{q=2}^{p-1} t(i, j_q) \right] \text{ for } 2 \leq p \leq k.$$

Observe that $|T(i, j_p)| = t(i, j_p)$ for all $1 \leq p \leq k$. We state in Algorithm 2 the procedure for realizing (r, s).

Input: A pair (r, s) of sequences.
Output: A bipartite graph $H = (X, Y, E)$, if it exists. H is specified implicitly by the neighborhoods of vertices in Y.

1. Compute $r^\star$. Test if $r^\star \succeq s$. If the test fails, then declare that *(r,s) is not a bipartite sequence* and STOP.
2. Set $(a, b) := (r^\star, s)$ and apply Algorithm 1 to compute the unit transformations $t(i, j)$ for $1 \leq i, j \leq n$.
3. Compute the bipartite graph F which realizes (r, a). F is represented implicitly: for each $1 \leq j \leq n$, compute the set of neighbors of y_j, namely $N_F(y_j) = [1, \ldots, a_j]$.
4. Compute $A(i)$ and $B(j)$ for $1 \leq i, j \leq n$.
5. Compute $\beta(i, j)$ for all i and j such that $t(i, j) \neq 0$; compute $\gamma(i)$ for $1 \leq i \leq n$.
6. Compute the intervals $T(i, j)$ for $1 \leq i \leq n$ and $j \in A(i)$.
7. For all i such that $a_i > b_i$, compute $N_H(y_i) := N_F(y_i) - \bigcup_{j \in A(i)} T(i, j)$.
8. For all j such that $a_j < b_j$, compute $N_H(y_j) := N_F(y_j) \cup (\bigcup_{i \in B(j)} T(i, j))$.
9. For all i such that $a_i = b_i$, set $N_H(y_i) := N_F(y_i)$.

ALGORITHM 2. *Algorithm for computing a bipartite graph.*

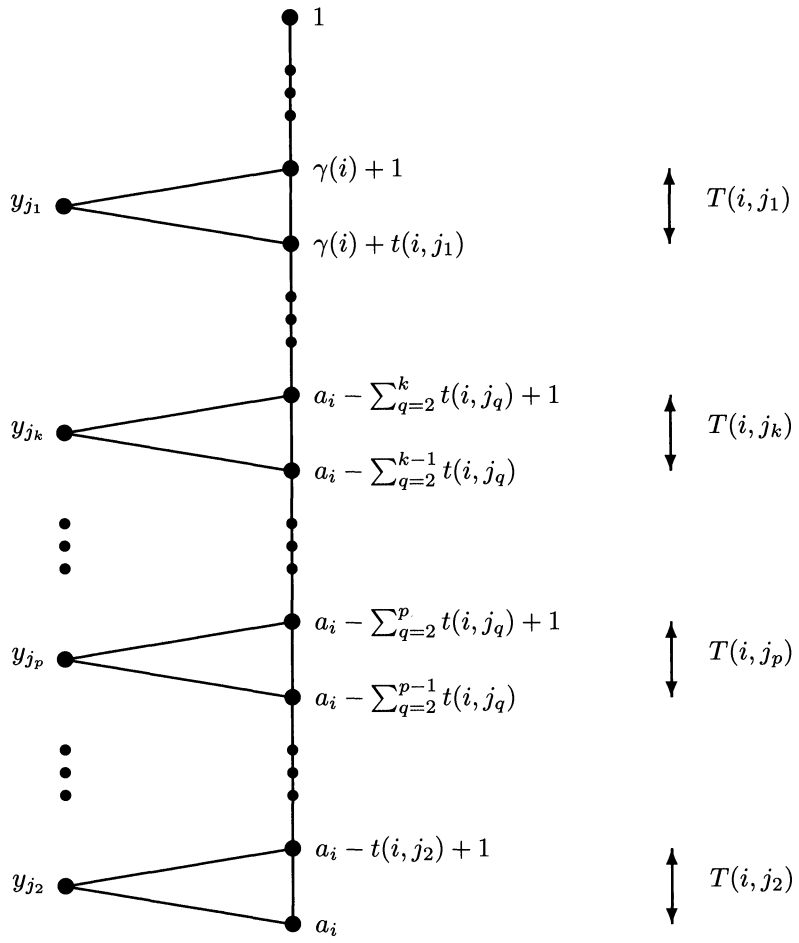


FIG. 2. *Defining* $T(i,j)$. *For example,* $T(i,j_p)$ *is the set of elements* $a_i-\sum_{q=2}^{p} t(i,j_q)+1, a_i-\sum_{q=2}^{p} t(i,j_q)+2,\ldots,a_i-\sum_{q=2}^{p-1} t(i,j_q)$.

Example 4. Let $r=(4,3,3,2,2,2,1)$ and $b=s=(4,3,3,3,3,2,2)$. Then
$X=\{x_1,\ldots,x_7\}$, $Y=\{y_1,\ldots,y_6\}$, and $a=r^*=(7,6,3,1,0,0)$.
Applying Algorithm 1 to the pair (a,b) we get
$t(1,4)=2$, $t(1,5)=1$, $t(2,5)=1$, $t(2,6)=2$;
$A(1)=\{4,5\}$, $A(2)=\{5,6\}$;
$B(4)=\{1\}$, $B(5)=\{1,2\}$, $B(6)=\{2\}$;
$\alpha(1)=4$, $\alpha(2)=5$, $\gamma(2)=a_5=0$, $\gamma(1)=a_4=1$.
The realization F of (r,a) has the following implicit representation. Recall that $(x_k,y_i)\in F$ iff $k\in N_F(y_i)$, and $N_F(y_i)=[1,a_i]$ is represented by its endpoints. We set
$N_F(y_1)=[1,7]$, $N_F(y_2)=[1,6]$, $N_F(y_3)=[1,3]$,
$N_F(y_4)=[1,1]$, $N_F(y_5)=N_F(y_6)=\emptyset$,
$T(1,4)=[\gamma(1)+1,\gamma(1)+t(1,4)]=[2,3]$,
$T(1,5)=[a_1-t(1,5)+1,a_1]=[7,7]$,

$$T(2,5) = [\gamma(2)+1, \gamma(2)+t(2,5)] = [1,1],$$
$$T(2,6) = [a_2 - t(2,6)+1, a_2] = [5,6].$$

The desired realization H of (r,s) has the following implicit representation:

$$N_H(y_1) = N_F(y_1) - (T(1,4) \cup T(1,5)) = [1,1] \cup [4,6],$$
$$N_H(y_2) = N_F(y_2) - (T(2,5) \cup T(2,6)) = [2,4],$$
$$N_H(y_3) = N_F(y_3) = [1,3],$$
$$N_H(y_4) = N_F(y_4) \cup T(1,4) = [1,3],$$
$$N_H(y_5) = N_F(y_5) \cup (T(1,5) \cup T(2,5)) = [1,1] \cup [7,7],$$
$$N_H(y_6) = N_F(y_6) \cup T(2,6) = [5,6]. \qquad \square$$

LEMMA 6.3. *The bipartite multigraph $H = (X \cup Y, E)$ computed in Algorithm 2 is a simple bipartite graph.*

Proof. It suffices to show that there are no multiple edges in H that are incident to any vertex y_j. This is equivalent to proving that $N_H(y_j)$ is a set (i.e., it has no duplicate elements). If $a_j \geq b_j$, $N_H(y_j)$ is a set as $N_F(y_j)$ is one. Consider now the case that $a_j < b_j$. Then $N_H(y_j) = N_F(y_j) \cup J$, where $J = \bigcup_{i \in B(j)} T(i,j)$. We prove that $N_H(y_j)$ has no duplicates by showing that J has no duplicates and that $N_F(y_j) \cap J = \emptyset$. Recall that $N_F(y_j) = [1, \ldots, a_j]$.

Let the elements of $B(j)$ (in the increasing order) be $i_1, \ldots, i_k$. Since $j \in A(i_1)$, we have $\alpha(i_1) \leq j$. First consider the case $\alpha(i_1) = j$.

Case 1. $k = 1$. Then $J = T(i_1,j) = [\gamma(i_1)+1, \ldots, \gamma(i_1)+t(i_1,j)] = [a_j + 1, \ldots, a_j + t(i_1,j)]$. Thus J is a set and $N_F(y_j) \cap J = \emptyset$.

Case 2. $k \geq 2$. By Lemma 5.4, $\alpha(i_q) = j$ for $2 \leq q \leq k$. By definition, $T(i_q,j) = [\gamma(i_q)+1, \ldots, \gamma(i_q)+t(i_q,j)]$ for $1 \leq q \leq k$. Then, by Lemma 5.5, $T(i_q,j) = [\gamma(i_q)+1, \ldots, \gamma(i_{q-1})]$ for $2 \leq q \leq k$ and $T(i_1,j) = [\gamma(i_1)+1, \ldots, \gamma(i_1)+t(i_1,j)]$. Hence, for distinct p and q such that $1 \leq p,q \leq k$, we obtain $T(i_p,j) \cap T(i_q,j) = \emptyset$ and thus J has no duplicates. Further, $J = \cup_{p=1}^{k} T(i_p,j) = [\gamma(i_k)+1, \ldots, \gamma(i_k) + \sum_{p=1}^{k} t(i_p,j)]$, and so $\min(J) = \gamma(i_k)$. We now obtain $N_F(y_j) \cap J = \emptyset$, as $\max(N_F(j)) = a_j = \gamma(i_k) = \min(J) - 1$ (the second equality follows from Lemma 5.5).

Now consider the other case, namely $\alpha(i_1) < j$. Let $j_1 = \alpha(i_1), j_2, \ldots, j_p = j$ be the elements of $A(i_1)$ that are $\leq j$. By definition

$$T(i_1,j) = \left[a_{i_1} - \sum_{q=2}^{p} t(i_1,j_q) + 1, \ldots, a_{i_1} - \sum_{q=2}^{p-1} t(i_1,j_q) \right].$$

Case 1. $k = 1$. Then $J = T(i_1,j)$ and

$$\begin{aligned}
\max(N_F(j)) &= a_j \\
&< b_j \\
&\leq b_{i_1} \qquad \text{(as } i_1 \leq j \text{ and } b \text{ is nonincreasing)} \\
&= a_{i_1} - \sum_{u \in A(i_1)} t(i_1,u) \qquad \text{(by Lemma 5.3)} \\
&< a_{i_1} - \sum_{q=2}^{p} t(i_1,j_q) \\
&= \min(J) - 1.
\end{aligned}$$

Hence, $N_H(y_j) \cap J = \emptyset$.

Case 2. $k \geq 2$. Put $I = \bigcup_{p=2}^{k} T(i_p,j)$; then $J = T(i_1,j) \cup I$. We claim that I is a set. By Lemma 5.4, $\alpha(i_p) = j$ for all $2 \leq p \leq k$. By Lemma 5.5, $T(i_p,j) =$

$[\gamma(i_p)+1,\ldots,\gamma(i_{p-1})]$ for $3 \le p \le k$, and $T(i_2,j) = [\gamma(i_2)+1,\ldots,\gamma(i_2)+t(i_2,j)]$. Using $\gamma(i_k) = a_j$, we obtain $I = [a_j+1,\ldots,\gamma(i_2)+t(i_2,j)]$, completing the claim. Now,

$$\begin{aligned}
\max(I) &= \gamma(i_2) + t(i_2,j) \\
&= a_j + \sum_{q=3}^{k} t(i_q,j) + t(i_2,j) \qquad \text{(by Lemma 5.5)} \\
&< a_j + \sum_{u\in B(j)} t(u,j) \\
&= b_j \qquad \text{(by Lemma 5.3)} \\
&\le b_{i_1} \qquad \text{(as } i_1 \le j \text{ and } b \text{ is nonincreasing)} \\
&= a_{i_1} - \sum_{u\in A(i_1)} t(i_1,u) \qquad \text{(by Lemma 5.3)} \\
&< a_{i_1} - \sum_{q=2}^{p} t(i_1,j_q) \\
&= \min(T(i_1,j)) - 1.
\end{aligned}$$

Hence $T(i_1,j) \cap I = \emptyset$, implying that J has no duplicates. Finally, $\max(N_F(y_j)) = a_j = \min(I) - 1 = \min(J) - 1$ and, hence, $N_F(y_j) \cap J = \emptyset$. □

LEMMA 6.4. *In the graph $H = (X \cup Y, E)$ computed by Algorithm 2, $d_H(x_i) = r_i$ and $d_H(y_j) = b_j$ $(= s_j)$.*

Proof. Clearly, $d_H(x_i) = d_F(x_i) = r_i$. If $a_j = b_j$ then $d_H(y_j) = b_j$, since $N_H(y_j) = N_F(y_j)$ and $d_F(y_j) = a_j$. Suppose next that $a_j < b_j$. Then

$$\begin{aligned}
d_H(y_j) &= |N_F(y_j)| \\
&= |N_F(y_j)| + \sum_{i\in B(j)} |T(i,j)| \\
&= a_j + \sum_{i\in B(j)} t(i,j) \\
&= b_j.
\end{aligned}$$

Now consider the remaining case: $a_j > b_j$. Let $i = j$ for convenience and put $I = \sum_{u\in A(i)} T(i,u)$. Then $N_H(y_i) = N_F(y_i) - I$. Let the elements of $A(i)$ be $j_i = j, j_2, \ldots, j_k$. Now

$$\begin{aligned}
\gamma(i) + t(i,j_1) &\le a_{j_1} + \sum_{q\in B(j_1)} t(q,j_1) \\
&= b_{j_1} \\
&\le b_i \\
&= a_i - \sum_{q=1}^{k} t(i,j_q) \\
&< a_i - \sum_{q=2}^{k} t(i,j_q).
\end{aligned}$$

Thus $I = [\gamma(i)+1, \ldots, \gamma(i)+t(i,j_1)] \cup [a_i - 1, \ldots, a_i - \sum_{q=2}^{k} t(i,j_q)]$ and, hence, $I \subseteq N_F(y_i)$. Now

$$\begin{aligned} |I| &= t(i,j_1) + \sum_{q=2}^{k} t(i,j_q) \\ &= \sum_{u \in A(i)} t(i,u) \\ &= a_i - b_i. \end{aligned}$$

Finally, $d_H(y_i) = |N_H(y_i)| = |N_F(y_i)| - |I| = a_i - (a_i - b_i) = b_i$. □

We now outline the parallel complexity of Algorithm 2. As mentioned in §1, the graphs computed by our algorithms possess implicit representations: the graphs can be stored in $O(n)$ space. For ease of notation, we assume that n denotes the total number of components of r and s (i.e., $n := n + m$). To analyze step 1, we first sort the sequences r and s using the algorithm of [8] and store the sorted sequences in the arrays R and S, respectively. Sorting can be performed in $O(\log n)$ time using $O(n)$ EREW PRAM processors. We show how to perform all other steps in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors. $r^\star$ can be computed by cross-ranking [14] the array R with the array $(1, 2, \ldots, n)$. Step 2 can be performed by using Algorithm 1. It reports $t(i,j)$'s in a lexicographic order. Steps 4, 5, and 6 can be implemented by performing the appropriate prefix sums. In step 7, observe that $\bigcup_{j \in A(i)} T(i,j)$ is union of at most three intervals, and hence can be computed within the same resource bounds. Therefore, Algorithm 2 can be implemented in $O(\log n)$ time using $O(n)$ EREW PRAM processors. We summarize the result in the following.

THEOREM 6.5. *Given a pair (r,s) of nonnegative integer sequences, a bipartite graph that realizes (r,s) can be computed in $O(\log n)$ time using $O(n)$ EREW PRAM processors, where n is the total number of components of r and s. Moreover, if the sequences r and s are given as sorted sequences, then the graph can be computed in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors.*

7. Realizing degree sequences. Let $d = (d_1, \ldots, d_n)$ be a nonnegative integer sequence, where $n > d_1 \geq d_2 \geq \cdots \geq d_n \geq 0$. In this section we present parallel algorithms to compute a graph $G = (V,E)$ on the vertex set $V = \{v_1, \ldots, v_n\}$ that realizes d. Our parallel algorithms are based on the proof of Theorem 4.1 and these are the main steps.

Step 1. From d, compute the sequence (c,c) and a realization $H = (X,Y,E)$ of (c,c). From H compute another realization H' of (c,c) such that $(x_i, y_i) \in H'$ iff $1 \leq i \leq \mu$.

Step 2. Compute from H' a symmetric digraph $D = (V,E)$ such that $d_D^-(v_i) = d_D^+(v_i) = d_i$. Then compute G.

We first discuss the parallel implementation of Step 2; the essential ideas are described in the proof of Lemma 2.4. Compute the digraph $\tilde{D} = (V, \tilde{E})$ on the vertex set $V = \{v_1, \ldots, v_n\}$, where $(v_i, v_j) \in \tilde{D}$ iff $(x_i, y_j) \in H'$ and $(x_j, y_i) \notin H'$. Then, the indegree of each vertex in $\tilde{D}$ is same as its outdegree. Compute Eulerian tours in $\tilde{D}$ using the algorithm of [5]; this algorithm computes an Eulerian tour for each connected component of $\tilde{D}$. Each trail is stored in an array. Using parallel list ranking, compute the number of edges in each trail. Consider first the even trails. Using the array representation of the trails, delete alternate arcs and insert the appropriate arcs as stated in the proof of Lemma 2.4. After this step, we are left with an even number of

odd trails and we group them in pairs. For each pair, determine if it belongs to Case 1 or Case 2 in the proof of Lemma 2.4. In either case, delete and insert appropriate arcs. We now have the required symmetric digraph $D = (V, E)$. Compute an undirected graph G from D by replacing the two symmetric arcs (v_i, v_j) and (v_j, v_i) with the edge (v_i, v_j). Then G is the desired realization of d.

We now analyze the complexity of step 2. The graph H' is represented in terms of adjacency lists: for each $y \in Y$, we maintain $N_{H'}(y)$ as a list. Let m denote the number of edges of G. The digraph $\tilde{D}$ can be computed in $O(\log n)$ time using $O(m)$ CRCW processors as follows. For each vertex, sort the vertices adjacent to it by the algorithm of [8] and test in $O(\log n)$ time if $(x_j, y_i) \in H$. The algorithm of [5] computes Eulerian tours in $\tilde{D}$ in $O(\log n)$ time using $O(m)$ CRCW PRAM processors. Parallel list ranking requires $O(\log n)$ time using $O(m)$ processors and the lists can be represented as consecutive elements in an array. Once we have the array representation, all other steps can be performed within the claimed complexity bounds. Hence the overall complexity of performing step 2 is $O(\log n)$ time using $O(m)$ CRCW PRAM processors.

In the remainder of this section, we discuss the parallel implementation of step 1. Recall from §4 the definitions of high-degree and low-degree edges. In step 1, we wish to compute a realization of (c, c) which has all high-degree edges and no low-degree edges. The main steps involved in obtaining such a realization are (i) compute a bipartite realization H of (c, c) using, for example, Algorithm 2; (ii) compute appropriate exchange pairs in H to obtain all high-degree edges; and (iii) compute appropriate exchange pairs in the resulting graph to remove the low-degree edges.

We provide two alternate parallel algorithms for implementing step 1. The first algorithm is simple and intuitive and is based on several interesting lemmas. It has high complexity, since it proceeds by reducing the computation to that of recursively computing maximal matchings. On the other hand, the second algorithm is efficient, though involved: it is based on the implicit structure of H computed by Algorithm 2. We present the first algorithm in §7.1 and the second one in §7.2.

7.1. The first algorithm. Recall that H is a realization of (c, c). If there are missing high-degree edges in H, then we execute the procedure described in §7.1.1. This is followed by executing the procedure for low-degree edges, as described in §7.1.2. After performing these two steps, we are left with a bipartite realization H' of H, which has all high-degree edges and no low-degree edges. This gives the outline of the first algorithm and we summarize the result in the following.

THEOREM 7.1. *Given an integer sequence d, a realization of d can be computed in $O(\log^4 n)$ time using $O(n^3)$ CRCW PRAM processors.*

Proof. The proof and the complexity analysis follow from Lemmas 2.4, 7.4, and 7.5. □

7.1.1. Procedure for high-degree edges. In this subsection, we present a parallel algorithm to compute appropriate exchange pairs in the realization H, so that by performing the corresponding exchanges (see Figure 1) we obtain another realization of (c, c) having all high-degree edges. The procedure has two phases.

In the first phase, we restrict ourselves to the subgraph of H induced by the high-degree vertices. Compute a maximal number of exchange pairs in this subgraph as follows. Define a graph G' on the vertex set $\{1, 2, \ldots, \mu\}$. There is an edge between i and j in G' iff $(x_i, y_i), (x_j, y_j) \notin H$, and the edges (x_i, y_j) and (x_j, y_i) form an exchange pair. Compute a maximal matching M in G' and then perform the corresponding

exchanges in H. For notational simplicity, let H be the resulting bipartite graph after performing these exchanges.

In the second phase, we obtain the rest of the missing high-degree edges in H. Define as follows a bipartite graph $\mathcal{H} = (\mathcal{P}, \mathcal{Q}, \mathcal{E})$ over the missing high-degree edges and the corresponding exchange pairs in H. $\mathcal{P} = \{p_i : (x_i, y_i) \notin H, 1 \leq i \leq \mu\}$ and $\mathcal{Q} = \{q_{lk} : (x_l, y_k) \notin H, l \neq k \text{ and } n \geq k > \mu\}$. Further, $(p_i, q_{lk}) \in \mathcal{E}$ iff the edges (x_i, y_k) and (x_l, y_i) form an exchange pair in H. The following lemma establishes an important property of the bipartite graph $\mathcal{H}$.

LEMMA 7.2. *Suppose that there is at least one high-degree edge missing in H and let the bipartite graph $\mathcal{H} = (\mathcal{P}, \mathcal{Q}, \mathcal{E})$ be as defined above. Then* (i) $d(p_i) \geq 1$ *for all i and* (ii) *if $(p_i, q_{lk}) \in \mathcal{E}$, then $d(p_i) \geq d(q_{lk})$, where $d(z)$ is the degree of vertex z in $\mathcal{H}$.*

Proof. Part (i) follows from Lemma 4.3. We say a vertex x_i is a *high-degree neighbor* of y_k, if $(x_i, y_k) \in H$ and $1 \leq i \leq \mu$. Let A_1 be the set of high-degree neighbors of y_k, which are not neighbors of y_i in H. Let A_2 be the set of high-degree neighbors of y_k which are also neighbors of y_i in H. Observe that $d(q_{lk}) \leq |A_1| + |A_2|$. To prove part (ii) of the lemma, we show that $d(p_i) \geq |A_1| + |A_2|$. This is achieved by proving that $N_{\mathcal{H}}(p_i) \supseteq B_1 \cup B_2$, where $|B_1| \geq |A_1|$, $|B_2| \geq |A_2|$, and $B_1 \cap B_2 = \emptyset$. Recall that c_i is the degree of y_i in H and that $c_i > \mu \geq c_k$ (as $i \leq \mu$ and $k > \mu$). This implies that $|A_1^{'}| \geq |A_1|$, where $A_1^{'} = \{\alpha : x_\alpha \in N_H(y_i) - N_H(y_k), \alpha \neq k\}$. The edges (x_i, y_k) and (x_α, y_i) form an exchange pair for every $\alpha \in A_1^{'}$ and we define $B_1 = \{q_{\alpha k} : \alpha \in A_1^{'}\}$. Observe that $B_1 \subseteq N_{\mathcal{H}}(p_i)$. We define B_2 now. If $(x_j, y_i) \in H$ for some $1 \leq j \leq \mu$ and $(x_j, y_j) \notin H$, then $(x_i, y_j) \notin H$ by the fact that a maximal matching is found in Phase 1 of Algorithm 3. Hence $|A_2^{'}| \geq |A_2|$, where $A_2^{'} = \{\beta : (x_i, y_\beta) \in H, \beta > \mu, \beta \neq k\}$. For every $\beta \in A_2^{'}$, there exists a vertex, say $x_{\beta'}$, such that $\beta^{'} \neq \beta$ and (x_i, y_β) and $(x_{\beta'}, y_i)$ form an exchange pair in H. Define $B_2 = \{q_{\beta'\beta} : \beta \in A_2^{'}\}$ and note that $B_2 \subseteq N_{\mathcal{H}}(p_i)$. To complete the proof, observe that $B_1 \cap B_2 = \emptyset$. □

COROLLARY 7.3. *In $\mathcal{H} = (\mathcal{P}, \mathcal{Q}, \mathcal{E})$, there exists a matching that matches all vertices of $\mathcal{P}$.*

Proof. The proof follows from Lemmas 2.3 and 7.2. □

We discuss some algorithmic aspects of Lemma 7.2. We compute the bipartite graph $\mathcal{H} = (\mathcal{P}, \mathcal{Q}, \mathcal{E})$ from H. From Corollary 7.3, we know that any maximum matching in $\mathcal{H}$ matches all vertices in $\mathcal{P}$. By finding a maximum matching in $\mathcal{H}$, we can compute the corresponding exchange pairs in H and hence obtain all the missing high-degree edges. Unfortunately, only randomized parallel algorithms are known for computing a maximum matching [19]. In order to solve our problem deterministically, we resort to the special structure of $\mathcal{H}$ stated in Lemma 7.2 and Corollary 7.3. The solution is presented in Algorithm 3.

LEMMA 7.4. *Algorithm 3 computes a bipartite realization of (c, c) having all high-degree edges. It runs in $O(\log^4 n)$ time using $O(n^3)$ CRCW PRAM processors.*

Proof. We first show the correctness of the algorithm. In every step, observe that (c, c) is the degree sequence of the bipartite graph H. Consider an iteration of the while loop. The correctness of Phase 1 is obvious. Assume that H has some high-degree edges missing after Phase 1 and let $\mathcal{H}$ be as defined in Phase 2. Then $\mathcal{E}$ is not empty by condition (i) of Lemma 7.2. Thus H has more high-degree edges at the end of Phase 2 than it had in the beginning.

We now discuss the complexity of the algorithm. By Lemma 2.2 and Corollary 7.3, the maximal matching computed in step 2 of Phase 2 has size at least $\frac{1}{2}|\mathcal{P}|$. Thus in each iteration of the while loop, the number of missing high-degree edges in H

While there are missing high-degree edges in H **do**
begin
Phase 1:

1. Compute a graph G' on the vertex set $\{1, 2, ..., \mu\}$; there is an edge between i and j if and only if (x_i, y_j) and (x_j, y_i) form an exchange pair.
2. Compute a maximal matching M in G' using the algorithm of [17].
3. From M, compute the exchange pairs in H and perform exchanges to obtain the corresponding high-degree edges; let the resulting graph be H.

Phase 2:

1. Compute a bipartite graph $\mathcal{H} = (\mathcal{P}, \mathcal{Q}, \mathcal{E})$ from H, where $\mathcal{P} = \{p_i : (x_i, y_i) \notin H, 1 \leq i \leq \mu\}$, $\mathcal{Q} = \{q_{lk} : (x_l, y_k) \notin H, l \neq k \text{ and } n \geq k > \mu\}$, and $(p_i, q_{lk}) \in \mathcal{E}$ if and only if the edges (x_i, y_k) and (x_l, y_i) form an exchange pair in H.
2. Compute a maximal matching M in $\mathcal{H}$ using the algorithm of [17].
3. From M, compute the exchange pairs in H and perform exchanges to obtain the corresponding high-degree edges; let the resulting graph be H.

end.

ALGORITHM 3. *A parallel algorithm to obtain the missing high-degree edges in H.*

drops down by at least a factor of 2, which implies that the while loop is executed for at most $O(\log n)$ times. In each iteration, the graph G' can be computed in $O(1)$ time using $O(n^2)$ processors and the graph $\mathcal{H}$ in $O(1)$ time using $O(n^3)$ processors. Computing maximal matchings in G' and $\mathcal{H}$ takes $O(\log^3 n)$ time using $O(n^3)$ CRCW PRAM processors. Hence the overall complexity is as stated in the lemma. □

7.1.2. Procedure for low-degree edges. Let H be the bipartite graph computed by Algorithm 3; H has all high-degree edges and possibly some low-degree edges. Our aim in this subsection is to transform H into another bipartite realization H' of (c, c) such that H' has all high-degree edges and no low-degree edges. Our algorithm for achieving this is similar to Algorithm 3, and we present below the necessary modifications.

As before, there are two phases. In the first phase, we restrict ourselves to the subgraph of H induced by the low-degree vertices. Define a graph G' on the vertex set $\{\mu + 1, \ldots, n\}$. There is an edge between i and j in G' iff the edges (x_i, y_i) and (x_j, y_j) form an exchange pair. In the second phase, define as follows a bipartite graph $\mathcal{H} = (\mathcal{P}, \mathcal{Q}, \mathcal{E})$ over the low-degree edges and the corresponding exchange pairs in H. $\mathcal{P} = \{p_i : (x_i, y_i) \in H, \mu < i \leq n\}$, and $\mathcal{Q} = \{q_{lk} : (x_l, y_k) \in H, l \neq k \text{ and } 1 \leq k \leq \mu\}$. Further, $(p_i, q_{lk}) \in \mathcal{E}$ iff the edges (x_i, y_i) and (x_l, y_k) form an exchange pair in H. The results of Lemma 7.4 hold with this definition of $\mathcal{H}$ also. The rest of the discussion is similar to that of previous subsection. We conclude with the following.

LEMMA 7.5. *A bipartite realization of (c, c) having all high-degree edges and no low-degree edges can be computed in $O(\log^4 n)$ time using $O(n^3)$ CRCW PRAM processors.*

7.2. An efficient algorithm. We present an efficient computation of the bipartite graph H' introduced at the beginning of this section. The computation is based on the structure of the bipartite graph computed by Algorithm 2.

The main result of this section is the following.

THEOREM 7.6. *Given an integer sequence d of length n, a graph that realizes d can be computed in $O(\log n)$ time using $O(n + m)$ CRCW PRAM processors, where*

m is the number of edges in the realization.

7.2.1. The structure. We present in this section the structure of the bipartite graphs F and H computed by Algorithm 2.

Both graphs F and H (defined on the vertex set $X \cup Y$) are specified by giving the neighbors of the vertices in Y. For $y_i \in Y$, $N_F(y_i)$ is an interval of vertices, namely $N_F(y_i) = \{x_k : 1 \leq k \leq a_i\}$, and we denote $N_F(y_i)$ by the interval $[1, \ldots, a_i]$. $N_H(y_i)$ is a union of at most two disjoint intervals of vertices (cf. the proof of Lemma 6.3): $N_H(y_i) = \{x_k : k \in (L_i \cup M_i)\}$, where L_i and M_i are defined below; we denote $N_H(y_i)$ by $L_i \cup M_i$. Recall the definitions of $t(i,j)$, $A(i)$, $B(j)$, $\alpha(i)$, $\beta(i,j)$, $\gamma(i)$, and $T(i,j)$ from §6. To define L_i and M_i, we distinguish between the following cases.

Case 1. $a_i = b_i$. Then $L_i = [1, \ldots, a_i]$ and $M_i = \emptyset$.

Case 2. $a_i > b_i$. Let the elements of $A(i)$ be $j_1, \ldots, j_k$. Then $L_i = [1, \ldots, \gamma(i)]$, and $M_i = [\gamma(i) + t(i,j_1) + 1, \ldots, a_i - \sum_{q=2}^{k} t(i,j_q)]$.

Case 3. $a_i < b_i$. Put $j = i$ and let the elements of $B(j)$ be $i_1, \ldots, i_k$.

Case 3.1. $\alpha(i_1) = j$. Then $L_j = [1, \ldots, a_j + \sum_{q=1}^{k} t(i_q, j)]$ and $M_j = \emptyset$.

Case 3.2. $\alpha(i_1) < j$. Let $j_1 = \alpha(i_1), j_2, \ldots, j_p = j$ be the elements of $A(i_1)$ that are $\leq j$. Then $L_j = [1, \ldots, a_j + \sum_{q=2}^{k} t(i_q, j)]$ and $M_i = [a_{i_1} - \sum_{q=2}^{p} t(i_1, j_q) + 1, \ldots, a_{i_1} - \sum_{q=2}^{p-1} t(i_1, j_q)]$.

Example 5. In Example 4, $L_1 = [1,1]$, $M_1 = [4,6]$; $L_2 = \emptyset$, $M_2 = [2,4]$; $L_3 = [1,3]$, $M_3 = \emptyset$; $L_4 = [1,3]$, $M_4 = \emptyset$; $L_5 = [1,1]$, $M_5 = [7,7]$; $L_6 = \emptyset$, $M_6 = [5,6]$. □

LEMMA 7.7. *For all* $1 \leq i \leq n$, $\max(L_i) < \min(M_i)$.

Proof. The neighborhoods $N_H(y_i)$ are defined in the proof of Lemma 6.3. It is routine to verify that $\max(L_i) < \min(M_i)$ holds in all the cases considered in that proof. □

7.2.2. The algorithm. Our algorithm computes four bipartite graphs, namely $H_0, H_1, H_2, H_3 = H'$, on the same vertex set $X \cup Y$. These graphs are represented implicitly using the neighborhoods of vertices in Y. Throughout, $N_i(y)$ denotes the set of neighbors of $y \in Y$ in H_i, where $i \in \{0,1,2,3\}$.

Algorithm 4 has three steps. In the first step, we compute H_0 and H_1; they are the realizations of $(c, c^\star)$ and (c, c), respectively.

In the second step, we compute H_2 that has all high-degree edges, i.e., $(x_i, y_i) \in H_2$ for all $1 \leq i \leq \mu$; no new low-degree edges are created in this step. Suppose the high-degree edge (x_i, y_i) is absent in H_1. To create the edge (x_i, y_i) by performing an exchange, we need to find vertices x_l, y_k such that (x_i, y_k) and (x_l, y_i) form an exchange pair. To make sure that no multiple edges are created when parallel exchanges are done, we select $k = \alpha(i)$ (see Lemma 7.11). Further, we impose the condition $l \neq k$ in order to avoid creating any low-degree edges. More precisely, we choose $l = \theta(i) = \min\{\ell : \ell \neq \alpha(i), x_\ell \in N_1(y_i) - N_1(y_{\alpha(i)})\}$. See step 2 of Algorithm 4.

In the last step, we compute H_3 by deleting from H_2 all low-degree edges without destroying the high-degree edges. Suppose the low-degree edge (x_j, y_j) is present in H_2. To destroy the edge (x_j, y_j) by performing an exchange, we need to find vertices x_l, y_k such that (x_j, y_j) and (x_l, y_k) form an exchange pair. To make sure that no multiple edges are created when parallel exchanges are done, we select $k = \tau(j) = i$, where $j \in T(i,j)$ (see Lemma 7.14). Further, we impose the condition $l \neq k$ in order to avoid destroying the high-degree edges. More precisely, we choose $l = \psi(j) = \min\{\ell : \ell \neq \tau(j), x_\ell \in N_2(y_{\tau(j)}) - N_2(y_j)\}$. See step 3 of Algorithm 4.

The rest of this section is devoted to proving the following important result.

1. Let H_0 and H_1, respectively, be the realizations of $(c, c^\star)$ and (c, c) computed by Algorithm 2.
2. Compute $I = \{i : i \leq \mu, (x_i, y_i) \notin H_1\}$. Compute a new bipartite graph H_2 from H_1 by doing in parallel the following for all $i \in I$: Delete the edges $(x_i, y_{\alpha(i)})$ and $(x_{\theta(i)}, y_i)$ and add the edges (x_i, y_i) and $(x_{\theta(i)}, y_{\alpha(i)})$.
3. Compute $J = \{j : j > \mu, (x_j, y_j) \in H_2\}$. Compute a new bipartite graph H_3 from H_2 by doing in parallel the following for all $j \in J$: Delete the edges (x_j, y_j) and $(x_{\psi(j)}, y_{\tau(j)})$ and add the edges $(x_{\psi(j)}, y_j)$ and $(x_j, y_{\tau(j)})$.
4. Output H_3.

ALGORITHM 4. *An efficient algorithm.*

LEMMA 7.8. *Algorithm* 4 *computes a realization of* (c, c) *that has all high-degree edges and no low-degree edges. It runs in* $O(\log n)$ *time using* $O(n)$ *EREW PRAM processors.*

For ease of notation, we let $a = c^\star$ and $b = c$. Recall from §6 that the condition $k \in T(i, j)$ means: "y_j gets x_k from y_i in H_0;" i.e., $(x_k, y_i) \in H_0 - H_1$ and $(x_k, y_j) \in H_1 - H_0$.

The following two lemmas are used to prove the correctness of step 2 of Algorithm 4.

LEMMA 7.9. *Let* $i \leq \mu$ *be such that* $(x_i, y_i) \notin H_1$. *Then there exists a unique* j *such that* $i \in T(i, j)$. *Furthermore,* $j = \alpha(i)$ *and* $j > \mu$.

Proof. The condition $i \leq \mu$ implies that $a_i \geq \mu$, and thus $(x_i, y_i) \in H_0$. Further, the condition $(x_i, y_i) \notin H_1$ implies that $a_i > b_i$ as $N_1(y_i) \subseteq N_0(y_i)$ otherwise. Let the elements of $A(i)$ be $j_1, \ldots, j_k$. The condition $(x_i, y_i) \notin H_1$ implies that there exists a j_ℓ such that $i \in T(i, j_\ell)$. From the definition of $T(i, j)$ it follows that j_ℓ is unique. We now show that j_ℓ satisfies the other properties of j stated in the lemma. Suppose for contradiction that $j_\ell \neq \alpha(i)$, i.e., $j_\ell > \alpha(i)$. Then $T(i, j_\ell) = [a_i - \sum_{q=2}^{\ell} t(i, j_q) + 1, \ldots, a_i - \sum_{q=2}^{\ell-1} t(i, j_q)]$, and

$$\begin{aligned}
b_i &= |N_1(y_i)| \\
&= |L_i| + |M_i| \\
&\leq \max(M_i) \qquad \text{(by Lemma 7.7)} \\
&= a_i - \sum_{q=2}^{k} t(i, j_q) \\
&\leq a_i - \sum_{q=2}^{\ell} t(i, j_q) \\
&< \min(T(i, j_\ell)) \\
&\leq i \qquad (\text{as } i \in T(i, j_\ell)) \\
&\leq \mu,
\end{aligned}$$

which contradicts $b_i \geq \mu + 1$. To prove the remaining part, observe that $\gamma(i) \geq a_{j_\ell}$ and that i belongs to $T(i, j_\ell)$ only if $\gamma(i) < i$; the last inequality is possible only if $j_\ell > \mu$ (as $a_{j_\ell} \geq \mu + 1$ if $j_\ell \leq \mu$). □

LEMMA 7.10. *Let* $\mu < j \leq n$. *There exists at most one* i *such that* $i \in T(i, j)$. *In that case,* $i \leq \mu$ *and* $\alpha(i) = j$.

Proof. That there exists at most one i is clear from the definition of $T(i, j)$

(cf. Figure 2). Observe that i belongs to $T(i,j)$ only if $i \leq \mu$. Then $\alpha(i) = j$ by Lemma 7.9. □

LEMMA 7.11. *The bipartite multigraph H_2 computed in step 2 of Algorithm 4 is a simple bipartite graph that realizes (c,c). Moreover, $(x_i, y_i) \in H_2$ if $1 \leq i \leq \mu$.*

Proof. Let the set I be as defined in step 2 of the algorithm. Then, for distinct i and i' in I we obtain $\alpha(i) \neq \alpha(i')$ using Lemmas 7.9 and 7.10. Thus no multiple edges are created when new edges are added in step 2, and hence H_2 is a simple graph. The proof of the remaining part of the lemma is obvious. □

The following two lemmas are needed to prove the correctness of step 3 of Algorithm 4.

LEMMA 7.12. *Suppose $(x_j, y_j) \in H_2$ for some $j > \mu$. Then $(x_j, y_j) \in H_1$. Further, there exists a unique i such that $j \in T(i,j)$, $\alpha(i) < j$, and $i \leq \mu$.*

Proof. Recall that $\theta(i) \neq \alpha(i)$ in step 2 of Algorithm 4. So no new "low-degree edges" are created in H_2; i.e., no edge of the form (x_k, y_k), where $k \geq \mu + 1$, is introduced into H_2. Thus $(x_j, y_j) \in H_1$. The rest of the proof is similar to the proof of Lemma 7.9. □

LEMMA 7.13. *Let $i \leq \mu$. There exists at most one j such that $j \in T(i,j)$. In that case, $j > \alpha(i)$ and $j > \mu$.*

Proof. The proof is similar to the proof of Lemma 7.10. □

LEMMA 7.14. *The bipartite multigraph H_3 computed in step 3 of Algorithm 4 is a simple bipartite graph that realizes (c,c). Moreover, $(x_i, y_i) \in H_3$ iff $1 \leq i \leq \mu$.*

Proof. Let J, $\tau(j)$ and $\psi(j)$ be as in step 3 of the algorithm. For distinct j and j' in J we obtain $\tau(j) \neq \tau(j')$ using Lemmas 7.12 and 7.13. Thus no multiple edges are created when new edges are added in step 3 and hence H_3 is a simple graph. Clearly, H_3 is a realization of (c,c) and $(x_j, y_j) \notin H_3$ for all $\mu + 1 \leq j \leq n$. By Lemma 7.11, $(x_i, y_i) \in H_2$ for all $1 \leq i \leq \mu$. The fact that $\psi(j) \neq \tau(j)$ implies that no such edge (high-degree edge) of H_2 is destroyed when some edges are deleted in step 3. □

Proof of Lemma 7.8. The correctness of the algorithm follows from Lemma 7.14. Recall that the graphs H_0, H_1, H_2, H_3 are represented implicitly using the neighborhoods of vertices in Y. The complexity bounds for step 1 follow from Theorem 6.5.

Suppose the neighborhood of some vertex y_k changes in step 2. Then one of the following must hold: $k \in I$ or $k = \alpha(i)$ for some $i \in I$. The following claim states that the neighborhood of y_k changes by exactly one vertex.

Claim 1. $|N_2(y_k) - N_1(y_k)| = 1$.

Consider first the case $k \in I$. Lemma 7.9 implies that $\alpha(i) \geq \mu + 1$ for all $i \in I$. Since $k \leq \mu$, $k \neq \alpha(i)$ for any $i \in I$. Consider now the case $k = \alpha(i)$ for some $i \in I$. Then i is unique by Lemmas 7.9 and 7.10, completing the proof of the claim. Since $N_1(y)$ is a union of at most two disjoint intervals of vertices for all $y \in Y$, we can compute all $\theta(i)$'s in constant time using $O(n)$ processors. Further, the graph H_2 in step 2 can be computed in constant time using $O(n)$ processors.

Suppose the neighborhood of some vertex y_k changes in step 3. Then one of the following must hold: $k \in J$ or $k = \tau(j)$ for some $j \in J$. The following claim states that the neighborhood of y_k changes by exactly one vertex.

Claim 2. $|N_3(y_k) - N_2(y_k)| = 1$.

Consider first the case $k \in J$. Lemma 7.12 implies that $\tau(j) \leq \mu$ for all $j \in J$. Since $k \geq \mu + 1$, $k \neq \tau(j)$ for any $j \in J$. Consider now the case $k = \tau(j)$ for some $j \in J$. Then j is unique by Lemmas 7.12 and 7.13, completing the proof of the claim. All $\tau(j)$'s can be computed in constant time using $O(n)$ processors. Claim 1 implies that $N_2(y)$ is a union of at most three disjoint intervals of vertices for all $y \in Y$.

Hence all $\psi(j)$'s can be computed in constant time using $O(n)$ processors. Further, the graph H_3 in step 3 can be computed in constant time using $O(n)$ processors. □

Acknowledgments. We thank the referees for their comments and suggestions for improving the presentation of this paper. We also thank Shiva Chaudhuri for useful discussions.

REFERENCES

[1] A. Aho, J. Hopcroft, and J. Ullman, *Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974, Exercises, pp. 71–72.
[2] S. R. Arikati, *New Results in Algorithmic Graph Theory*, Ph.D. thesis, Department of Mathematics, University of Illinois, Chicago, 1993.
[3] S. R. Arikati and U. N. Peled, *Degree sequences and majorization*, Linear Algebra Appl., 199 (1994), pp. 179–211.
[4] T. Asano, *Graphical degree sequence problems with connectivity requirements*, Lecture Notes in Computer Science 762, Springer-Verlag, New York, Berlin, 1993, pp. 38–47.
[5] B. Awerbuch, A. Israeli, and Y. Shiloach, *Finding Euler circuits in logarithmic parallel time*, in Proc. ACM Symposium on Theory of Computing, ACM Press, New York, 1984, pp. 249–257.
[6] C. Berge, *Graphs and Hypergraphs*, North-Holland, Amsterdam, 1973.
[7] F.T. Boesch, ed., *Large-Scale Networks: Theory and Design*, IEEE Press, New York, 1976.
[8] R. Cole, *Parallel merge sort*, SIAM J. Comput., 17 (1988), pp. 770–785.
[9] A. Dessmark, A. Lingas, and O. Garrido, *On the parallel complexity of maximum f-matching and the degree sequence problem*, Lecture Notes in Computer Science 841, Springer-Verlag, New York, Berlin, 1994, pp. 316–325.
[10] P. Erdős and T. Gallai, *Graphs with prescribed degrees of vertices*, Mat. Lapok, II (1960), pp. 264–272. (In Hungarian.)
[11] L. R. Ford, Jr. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.
[12] D. R. Fulkerson, A. J. Hoffman, and M. H. McAndrew, *Some properties of graphs with multiple edges*, Canad. J. Math., 17 (1965), pp. 166–177.
[13] D. Gale, *A theorem on flows in networks*, Pacific J. Math., 7 (1957), pp. 1073–1082.
[14] T. Hagerup and C. Rüb, *Optimal merging and sorting on the EREW PRAM*, Infor. Proc. Letters, 33 (1989), pp. 181–185.
[15] F. Harary, *Graph Theory*, Addison-Wesley, New York, 1969.
[16] G. H. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities*, Cambridge University Press, New York, 1952.
[17] A. Israeli and Y. Shiloach, *An improved parallel algorithm for maximal matching*, Inform. Process. Lett., 22 (1986), pp. 57–60.
[18] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley, New York, 1992.
[19] R. M. Karp and V. Ramachandran, *Parallel algorithms for shared-memory machines*, in Handbook of Theoretical Computer Science, Vol. A, J. van Leeuwen, ed., Elsevier, Amsterdam, 1990, pp. 869–942.
[20] R. M. Karp, E. Upfal, and A. Wigderson, *Constructing a maximum matching in random NC*, Combinatorica, 6 (1986), pp. 35–48.
[21] L. Lovász and M. Plummer, *Matching Theory*, Academic Press, Budapest, Hungary, 1986.
[22] A. W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and its Applications*, Academic Press, New York, 1979.
[23] U. N. Peled and M. K. Srinivasan, *The polytope of degree sequences*, Linear Algebra Appl., 114 (1989), pp. 349–377.
[24] H. J. Ryser, *Combinatorial properties of matrices of zeros and ones*, Canad. J. Math., 9 (1957), pp. 371–377.
[25] G. Sierksma and H. Hoogeveen, *Seven criteria for integer sequences being graphic*, J. Graph Theory, 15 (1991), pp. 223–231.
[26] R. I. Tyshkevich, A. A. Chernyak, and Zh. A. Chernyak, *Graphs and degree sequences* I, Cybernetics, 23 (1987), pp. 734–745.
[27] J. van Leeuwen, *Graph algorithms*, in Handbook of Theoretical Computer Science, Vol. A, J. van Leeuwen, ed., Elsevier, Amsterdam, 1990, pp. 525–631.

SIAM J. DISCRETE MATH.
Vol. 9, No. 2, pp. 339–348, May 1996

015

THE TOTAL INTERVAL NUMBER OF A GRAPH II: TREES AND COMPLEXITY*

THOMAS M. KRATZKE† AND DOUGLAS B. WEST‡

Abstract. A *multiple-interval representation* of a simple graph G assigns each vertex a union of disjoint real intervals so that vertices are adjacent if and only if their assigned sets intersect. The *total interval number* $I(G)$ is the minimum of the total number of intervals used in such a representation of G. For triangle-free graphs, $I(G) = |E(G)| + t(G)$, where $t(G)$ is the minimum number of pairwise edge-disjoint trails that together contain an endpoint of each edge. This yields the NP-completeness of testing $I(G) = |E(G)| + 1$ (even for triangle-free 3-regular planar graphs) and an alternative proof that HAMILTONIAN CYCLE is NP-complete for line graphs. It also yields a linear-time algorithm to compute $I(G)$ for trees and a characterization of the trees requiring $|E(G)| + t$ intervals for fixed t. Further corollaries include the Aigner–Andreae bound of $I(G) \leq \lfloor (5n-3)/4 \rfloor$ for n-vertex trees (achieved by subdividing every edge of a star), a characterization of the extremal trees, and a shorter proof of the extremal bound $\lfloor (5m+2)/4 \rfloor$ for connected graphs.

Key words. interval representation, complexity, tree, trail

AMS subject classifications. 05C35, 05C05, 05C38, 05C45, 05C85, 68Q25

1. Introduction.

An *intersection representation* of a graph G assigns each vertex v a set $f(v)$ such that u, v are adjacent if and only if $f(u) \cap f(v) \neq \emptyset$. Conversely, the graph is the *intersection graph* of the sets in the representation. The most well-studied class of intersection graphs are the *interval graphs*, which are the intersection graphs obtainable by assigning each vertex a single interval on the real line. More generally, an intersection representation f that assigns each vertex a union of intervals on the real line is a *multiple-interval representation* of G. Let $|f(v)|$ denote the number of (pairwise disjoint) intervals whose union is $f(v)$. If $|f(v)| = k$, then we say that $f(v)$ *consists of* k intervals or that v is *assigned* k intervals.

In two natural ways, multiple-interval representations can measure how far a graph is from being an interval graph. The *interval number* of G is $i(G) = \min_f \max_{v \in V(G)} |f(v)|$, where the minimum is taken over all multiple-interval representations of G. The *total interval number* of G is $I(G) = \min_f \sum_{v \in V(G)} |f(v)|$, which can be viewed as minimizing the average number of intervals assigned per vertex instead of the maximum number. Always $I(G) \leq ni(G)$ for n-vertex graphs; the interval graphs without isolated vertices have interval number 1 and total interval number n.

Interval number has been studied for many years, beginning with [10] and [4]. Although introduced in [4], total interval number was not studied until Aigner and Andreae [1] obtained the maximum value of $I(G)$ for several classes of graphs on n vertices, including trees ($\lfloor (5n-3)/4 \rfloor$), 2-connected outerplanar graphs ($\lfloor 3n/2 - 1 \rfloor$), triangle-free planar graphs ($2n-3$), and triangle-free graphs ($\lceil (n^2+1)/4 \rceil$). For the latter three classes, they conjectured that the upper bounds would still hold when the "2-connected" or "triangle-free" restrictions were removed. In [7], we proved these conjectures for outerplanar and general graphs on n vertices, and we also proved the Aigner–Andreae conjecture that $\max I(G) = \lfloor (5m+2)/4 \rfloor$ if G is a connected graph

*Received by the editors February 25, 1993; accepted for publication (in revised form) August 11, 1995. This research was supported in part by ONR grant N00014-85K0570.

†Metron, Inc., Reston, VA 22091 (kratzke@metsci.com).

‡University of Illinois, Urbana, IL 61801 (west@math.uiuc.edu).

with m edges. The proof of their conjecture for planar graphs is quite lengthy and will appear in a later paper in this series. Other papers will study the maximum total interval number for cacti or Husimi trees on n vertices and for connected graphs with m edges having lower bounds on minimum vertex degree, connectivity, or edge-connectivity. Most of this work appeared in the dissertation of the first author, which was accepted in 1987 [6].

In this paper, we present a linear-time algorithm to compute the total interval number of a tree (§3). This is based on the equality $I(G) = m + t$ for a triangle-free graph with m edges, where t is the minimum number of edge-disjoint trails needed to touch every edge of the graph (§2). From this characterization, we also obtain the NP-completeness of testing $I(G) = m+1$ even for triangle-free 3-regular planar graphs and an alternative proof of the NP-completeness of HAMILTONIAN CYCLE for line graphs. A closer examination of the algorithm for trees yields a characterization of the trees requiring $m + t$ intervals for fixed t (§4). This in turn yields short proofs of the Aigner–Andreae extremal bound for trees and the extremal bound in [7] for connected graphs (§5).

2. Trail covers and complexity. We use n for the number of vertices of a graph G, m for the number of edges, $N(v)$ for the set of neighbors of v, and $x \leftrightarrow y$ for "x is adjacent to y."

In studying $I(G)$, we allow $f(v) = \emptyset$, so that isolated vertices contribute nothing to the count of intervals. As in the study of $i(G)$, it is natural to define the *depth* of a representation to be the maximum number of vertices to which a single point is assigned; the *depth-r total interval number* $I_r(G)$ is the minimum of $\sum |f(v)|$ over all representations of G with depth at most r. An interval in $f(v)$ is *displayed* if some portion of it intersects no other interval of f.

A *vertex cover* of G is a set of vertices that contains an endpoint of every edge of G. A collection of pairwise edge-disjoint trails whose vertices together form a vertex cover is a *trail cover*. The *trail cover number* $t(G)$ is the minimum number of trails in a trail cover of G. Traversed from left to right, a depth-2 representation can establish at most one edge for each interval after the first, so $I_2(G) \geq m + 1$, and this bound can be achieved only if there are no "gaps" in the representation. Lemma 2.1 extends this observation. This lemma appears in [7], but we repeat its proof here because the transformation to trail cover number is essential for computing $I(G)$ for trees. Algorithm 3.1 constructs a minimum trail cover.

LEMMA 2.1. *For a graph G with m edges, $I_2(G) = m + t(G)$, and hence $I(G) = m + t(G)$ if G is triangle-free.*

Proof. For triangle-free graphs, $I(G) = I_2(G)$. First we prove $I_2(G) \leq m + t(G)$. Let $\{Z_j\}$ be an optimal trail cover. For each trail $Z_j = (v_1, \ldots, v_r)$, choose r intervals in $(j-1, j)$ such that the ith interval intersects only the $i-1$th and $i+1$th intervals (for $2 \leq i \leq r-1$), and add the ith of these intervals to $f(v_i)$. Vertices may appear repeatedly in trails, and all these intervals are displayed. For each edge not in these trails, assign an interval for one endpoint within the displayed portion of its neighbor in $\cup V(Z_j)$. For each trail, the number of intervals used is one more than the number of intervals represented, so we have represented G with $m + t(G)$ intervals.

Conversely, given an optimal depth-2 representation, we obtain a trail cover consisting of $I_2(G) - m$ edge-disjoint trails. Because no more than two intervals intersect at any point, we can eliminate any intersection of intervals by shortening or deleting one interval without affecting any other intersection. Therefore, we may assume that every edge is represented exactly once. Removal of each nondisplayed interval from an optimal representation leaves a representation of edge-disjoint trails as described

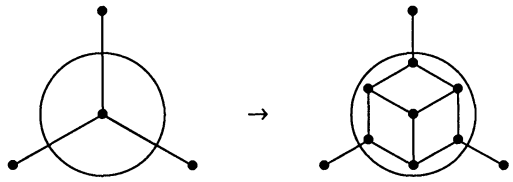


FIG. 1. *Transformation at each vertex.*

above, having deleted one edge for each interval deleted. Furthermore, the vertices of the resulting trails touch all edges of the original graph. If we now shrink each trail to a single vertex by deleting one interval and edge at a time, we have deleted every edge of G and one interval for each edge. There remain $I_2(G) - m$ intervals, one from each trail in the trail cover. □

COROLLARY 2.2. *The decision problem $I(G) \leq m+1$ is NP-complete, even when restricted to the class of planar, 3-regular, triangle-free graphs.*

Proof. The problem is in NP, because it is easy to check whether an assignment of $m+1$ intervals is a representation. For triangle-free graphs, the problem is equivalent to testing whether G has a single covering trail. It is well known that testing for a Hamiltonian path in a 3-regular planar graph is NP-complete [3]. Given an arbitrary 3-regular planar graph G, we replace each vertex by a 7-vertex subgraph as indicated in Figure 1. The resulting graph G' is 3-regular, planar, and triangle-free. It suffices to show that G has a Hamiltonian path if and only if G' has a covering trail. (This transformation was used in [9] to prove the NP-completeness of testing $i(G) \leq 2$.)

Given $v \in V(G)$, let $H(v)$ be the subgraph induced by the seven vertices that replace v in G'. Because $H(v)$ contains edges not incident to vertices of any other $H(w)$, a covering trail must enter every $H(v)$. Since only three edges enter each $H(v)$, a trail can enter and/or exit $H(v)$ only once. Therefore, contracting each $H(v)$ to a single vertex v turns G' into G and a covering trail of G' into a Hamiltonian path or cycle in G.

Conversely, if G has a Hamiltonian path P, then we can replace each internal vertex v of the path by a Hamiltonian path in $H(v)$ between the two-valent vertices of $H(v)$ that correspond to the edges incident to v in P. If v is an end of P, we use a Hamiltonian path in $H(v)$ ending at the central vertex of $H(v)$. The result is a Hamiltonian path of G', which is certainly a covering trail. □

From HAMILTONIAN CYCLE in 3-regular planar graphs, the same transformation proves that it is NP-hard to test whether a graph has a single *closed* covering trail. By combining this with known results about line graphs, we obtain a short alternative proof of the known result that testing for Hamiltonian cycles is NP-hard even when the input is restricted to line graphs. The existence of a Hamiltonian cycle in a line graph $L(H)$ is not equivalent to the existence of an Eulerian circuit in H.

COROLLARY 2.3 (see Bertossi [2]). *HAMILTONIAN CYCLE is NP-hard on line graphs.*

Proof. Given a line graph G with at least four vertices, we can retrieve the unique graph H such that $G = L(H)$ in linear time (Lehot [8]). We also know that G is Hamiltonian if and only if H has a closed covering trail (Harary and Nash-Williams [5]). As observed above, this is NP-hard. □

3. Trail covers of trees. Our recursive algorithm for computing the trail cover number of a tree computes additional information about the tree. We use (T, x) to denote a tree T with a vertex x distinguished as its *root*. Suppose $v \in V(T)$ and $\mathbf{C}$ is a trail cover of T. We say that $\mathbf{C}$ *visits* v if v is a vertex of a trail in $\mathbf{C}$, that $\mathbf{C}$ *ends at*

v if v is an endpoint of a trail in $\mathbf{C}$, and that $\mathbf{C}$ *isolates* v if v is a trail of length 0 in $\mathbf{C}$ (a *degenerate* trail). Isolating v implies ending at v, which in turn implies visiting v. Given a tree T rooted at x, the *code* $c(T,x)$ indicates the most restrictive of these conditions at the root that can be satisfied by a minimum trail cover. See Table 1.

TABLE 1.

$c(T,x)$	Condition
0	no minimum cover visits x
1	some minimum cover visits x but none ends at x
2	some minimum cover ends at x but none isolates x
3	some minimum cover isolates x

We will prove that the following recursive algorithm computes the trail cover number and code of a rooted tree.

ALGORITHM 3.1. Input (T,x). Output trail cover number t, code c, and trail cover $\mathbf{C}$ establishing t and c.

If $n(T) = 1$, set $t = 0$, $c = 0$, and $\mathbf{C} = \emptyset$. Otherwise, let $x_1, \ldots, x_k$ be the neighbors of x, designated as roots of the components $T_1, \ldots, T_k$ of $T - x$. Let $t_i, c_i, \mathbf{C}_i$ be the output of the algorithm for T_i rooted at x_i. For $0 \le j \le 3$, let $k_j = |\{i\colon\ c_i = j\}|$. Note that $\sum k_i = k > 0$.

If $k_2 + k_3 = 0$ and $k_1 = k$, then set $\mathbf{C} = \cup\mathbf{C}_i$, $t = \sum t_i$, and $c = 0$.

If $k_2 + k_3 = 0$ and $k_1 < k$, then set $\mathbf{C} = (\cup\mathbf{C}_i) \cup \{(x)\}$, $t = 1 + \sum t_i$, and $c = 3$.

If $k_2 + k_3 > 0$, then form $\mathbf{C}$ by beginning with $\cup\mathbf{C}_i$ and iteratively joining pairs of trails that end in $\{x_i\colon\ c_i \ge 2\}$ by edges from those roots to x. Set $t = \sum t_i - \lfloor (k_2 + k_3)/2 \rfloor$, and set $c = 1$ if $k_2 + k_3$ is even and $c = 2$ if $k_2 + k_3$ is odd.

Since the computation of c and t uses only $\{c_i\}$ and $\{t_i\}$, the algorithm can be used to compute the trail cover number without storing trails. It can be implemented to build the computation up from leaves and thus run in linear time and space.

The intuition behind the algorithm is that deleting a vertex from a minimum trail cover should leave minimum trail covers of the resulting subtrees. The code $c(T,x)$ computed in the algorithm takes care of the fact that this is not true for arbitrary minimum trail covers. This is illustrated by the tree T on the left in Figure 2.

There are several ways to cover the edges of this tree with three trails; Figure 2 shows one of them in solid edges. Deleting x leaves two subtrees that can be covered with one trail each. The only minimum covering of T that turns into minimum coverings of the subtrees when x is deleted is the one in which x is a degenerate trail. In the algorithm, this corresponds to the case $k_1 = k_2 = k_3 = 0$. This example suggests that a minimum covering in which the root is a degenerate trail is desirable if one exists. We say that a minimum trail cover $\mathbf{C}$ of a tree T with root x is *weakly optimal* if $\mathbf{C}$ ends at x or no minimum cover ends at x. Furthermore, $\mathbf{C}$ is *strongly optimal* if (a) $\mathbf{C}$ is weakly optimal and (b) $\mathbf{C}$ isolates x or no minimum cover isolates x. The statements "no minimum cover ends at x" and "no minimum cover isolates x" are equivalent to $c(T,x) \le 1$ and $c(T,x) \le 2$, respectively. The empty trail cover is a strongly optimal trail cover of the 1-vertex tree. The next theorem is a precise version of the intuition suggested above and is the main result needed to prove the correctness of the algorithm.

THEOREM 3.2. *If* $\mathbf{C}$ *is a strongly optimal trail cover of* (T,x)*, then the trails in* $T - x$ *obtained by deleting* x *from any trail containing it in* $\mathbf{C}$ *form weakly optimal trail covers of the components of* $T - x$ *rooted at the neighbors of* x.

FIG. 2. *Insufficient variants of optimality.*

Proof. Let $\{x_i\}$ be the neighbors of x, $\{T_i\}$ the subtrees, and $\{\mathbf{C}_i\}$ the resulting sets of trails. Suppose $\mathbf{C}_i$ is not weakly optimal, and let $\mathbf{D}_i$ be a weakly optimal trail cover of (T_i, x_i). In all cases, we construct a trail cover of (T, x) that contradicts the strong optimality of $\mathbf{C}$.

Case 1. $\mathbf{C}$ does not use the edge xx_i. In this case, all the trails of $\mathbf{C}_i$ are trails of $\mathbf{C}$. If $|\mathbf{D}_i| < |\mathbf{C}_i|$, let $\mathbf{D}' = \mathbf{C} - \mathbf{C}_i + \mathbf{D}_i$. Since $|\mathbf{D}'| < |\mathbf{C}|$ and $\mathbf{C}$ is strongly optimal, $\mathbf{D}'$ is not a trail cover, so it fails to touch some edge. The only edge that $\mathbf{D}'$ can miss is xx_i; if $\mathbf{D}'$ misses xx_i, then $\mathbf{C}$ does not visit x. If this happens, then $\mathbf{D} = \mathbf{D}' \cup \{(x)\}$ is a trail cover with $|\mathbf{D}| \leq |\mathbf{C}|$ that isolates x. This contradicts the strong optimality of $\mathbf{C}$, because $\mathbf{C}$ does not visit x.

Hence we may assume that $|\mathbf{D}_i| = |\mathbf{C}_i|$. Since we assumed that $\mathbf{C}_i$ is not weakly optimal, the weak optimality of $\mathbf{D}_i$ implies that $\mathbf{D}_i$ ends at x_i and $\mathbf{C}_i$ does not. Let D be the trail in $\mathbf{D}_i$ that ends at x_i, and let $\mathbf{D}_i'$ be the collection of trails obtained from $\mathbf{D}_i$ by extending D to include $x_i x$. Let $\mathbf{D} = \mathbf{C} - \mathbf{C}_i + \mathbf{D}_i'$; note that $|\mathbf{D}| = |\mathbf{C}|$. If $\mathbf{C}$ ends at x, then $\mathbf{D}$ has two trails ending at x that can be concatenated to obtain a trail cover smaller than $\mathbf{C}$. If $\mathbf{C}$ does not end at x, then $\mathbf{D}$ is a trail cover of the same size that ends at x. Each possibility contradicts the strong optimality of $\mathbf{C}$.

Case 2. $\mathbf{C}$ uses the edge xx_i on some trail C. In this case $\mathbf{C}_i$ ends at x_i. Since $\mathbf{C}_i$ is not weakly optimal, this implies $|\mathbf{D}_i| < |\mathbf{C}_i|$. Let $\mathbf{D}$ be the collection of trails obtained from $\mathbf{C}$ by deleting xx_i from C and replacing the resulting $\mathbf{C}_i$ by $\mathbf{D}_i$. Note that $\mathbf{D}$ ends at x, that $|\mathbf{D}| \leq |\mathbf{C}|$, and that $\mathbf{D}$ is a trail cover of T.

If C ends at x, then $\mathbf{D}$ isolates x but $\mathbf{C}$ does not, which contradicts the strong optimality of $\mathbf{C}$. If C does not end at x, then we consider two possibilities, as in the second half of Case 1. If $\mathbf{C}$ ends at x, then $\mathbf{D}$ has two trails ending at x that can be concatenated to obtain a smaller trail cover than $\mathbf{C}$. If $\mathbf{C}$ does not end at x, then $\mathbf{D}$ is a trail cover of at most the same size that ends at x. Each possibility contradicts the strong optimality of $\mathbf{C}$. □

The trees in Figure 2 show that both types of optimality are needed. The trail cover given by solid edges for the tree on the left shows that deletion of the root from a weakly optimal trail cover need not leave minimum trail covers for the subtrees. The trail cover given for the tree on the right shows that deletion of the root from a strongly optimal trail cover need not leave strongly optimal covers for the subtrees.

If $\mathbf{C}$ is a strongly optimal trail cover of (T, x), then applying Algorithm 3.1 to the resulting covers $\{\mathbf{C}_i\}$ of the subtrees reconstructs $\mathbf{C}$ or constructs another trail cover with the same number and placement of trail ends as $\mathbf{C}$. To prove the correctness of the algorithm, we will show that applying the algorithm to arbitrary weakly optimal trail covers of the rooted subtrees generates a strongly optimal trail cover of (T, x).

LEMMA 3.3. *If some weakly optimal trail cover* $\mathbf{C}$ *of a rooted tree* (T, x) *visits* x *but does not end at* x*, then every minimum trail cover of* (T, x) *is strongly optimal.*

Proof. In this situation the definition of weakly optimal implies that $c(T, x) = 1$ and that no minimum trail cover ends at x. Hence every weakly optimal trail cover is strongly optimal. A minimum trail cover $\mathbf{C}'$ can fail to be strongly optimal only if it does not visit x. Since x does not appear, $\mathbf{C}'$ consists of $|\mathbf{C}'| = |\mathbf{C}|$ trails covering $T - x$. If we delete x from $\mathbf{C}$, we obtain at least $|\mathbf{C}| + 1$ trails, since $\mathbf{C}$ does not end at x. This implies that at least one of the components of $T - x$ does not receive a

minimum trail cover, which contradicts Theorem 3.2. □

In the application of the recursive step, Algorithm 3.1 treats subtrees with code 2 or 3 exactly the same. Hence it will behave the same and produce the same code and size of trail cover for the root as long as the trail covers of the subtrees are any weakly optimal trail covers. Phrasing the recursive step of the algorithm in terms of the trail cover alone, it (1) forms the union of $\{\mathbf{C}_i\}$, (2) extends to x any trail ending at a neighbor x_i, (3) concatenates pairs of trails extended to x until at most one remains, and (4) adds (x) as a trail of length 0 if no trail extended to x and some neighbor x_i is not visited by its weakly optimal $\mathbf{C}_i$. To complete the proof that the algorithm works, it suffices to show that the resulting $\mathbf{C}$ is strongly optimal.

THEOREM 3.4. *If (T, x) is a rooted tree with neighbors $\{x_i\}$ of the root, and $\{\mathbf{C}_i\}$ are weakly optimal trail covers of the rooted subtrees $\{(T_i, x_i)\}$ of $T - x$, then application of the recursive step of Algorithm* 3.1 *to $\{\mathbf{C}_i\}$ produces a strongly optimal trail cover $\mathbf{C}$ of (T, x).*

Proof. Suppose that $\mathbf{D}$ is a strongly optimal trail cover of (T, x) and that $\{\mathbf{D}_i\}$ for $\{(T_i, x_i)\}$ are obtained from $\mathbf{D}$ by deleting all appearances of x. By Theorem 3.2, $\mathbf{D}_i$ is a weakly optimal trail cover of (T_i, x_i). By the definition of weak optimality, $\mathbf{D}_i$ ends at x_i if and only if $\mathbf{C}_i$ ends at x_i. Hence applying the algorithm to $\{\mathbf{C}_i\}$ produces the same number of extensions to x as applying the algorithm to $\{\mathbf{D}_i\}$ to obtain $\mathbf{D}$. As a result, $\mathbf{C}$ and $\mathbf{D}$ have the same size and associated code, except possibly when there are no extensions to x.

In this final case, for each i neither $\mathbf{D}_i$ nor $\mathbf{C}_i$ ends at x_i. We still have identical size and code for $\mathbf{C}$ and $\mathbf{D}$ unless one of them visits all $\{x_i\}$ and the other does not. This possibility is forbidden by Lemma 3.3. □

4. Critical trees. To characterize the trees with interval number $m + t$, we characterize the trees that just barely require t trails in a trail cover. The graph obtained by contracting an edge e of G is denoted $G \cdot e$. An edge e is *contractible* if $t(G \cdot e) = t(G)$, and a tree is *critical* if it has no contractible edge. We seek to characterize the *k-critical* trees, which are the critical trees with trail cover number k.

By applying Algorithm 3.1 to (G, x) for arbitrary $x \in V(G)$, we obtain a unique code for any vertex of a tree G. We say that a tree vertex v is *essential* if $c(G, v) = 3$, *useful* if $c(G, v) = 2$, *not useful* if $c(G, v) < 2$, and *at least useful* if $c(G, v) \geq 2$. The terminology is chosen to suggest that the critical trees are those in which every vertex is at least useful. A *penultimate* vertex of a tree is a nonleaf vertex that has at most one nonleaf neighbor. We refer to a vertex of degree 2 as a *bivalent* vertex and an edge incident to a leaf as a *pendant edge.*

LEMMA 4.1. *In a tree G,* (1) *every penultimate vertex is at least useful,* (2) *the nonleaf neighbor of a penultimate vertex is not essential, and* (3) *if G is critical, then every penultimate vertex is bivalent.*

Proof. Let u be a penultimate vertex, and let v be its nonleaf neighbor. (1) We can modify any minimum trail cover so that it touches the pendant edges incident to u via a trail ending at u by moving a degenerate trail from a leaf to u or deleting the leaf from a trail containing u. (2) If $\mathbf{C}$ is a minimum trail cover isolating v, then some other trail must touch the edges from u to its leaf neighbors. As above, we may assume this trail ends at u. Extending this trail to v produces a smaller trail cover. (3) If u is not bivalent, let G' be the tree obtained by contracting all but one pendant edge incident to u. Since u is penultimate in G', G' has a minimum trail cover that ends at u. This is also a trail cover of G, so G was not critical. □

Next we describe how to grow the critical trees. Suppose that G is a tree with a vertex partition into two sets U and W. Let P be a 5-vertex path with central vertex

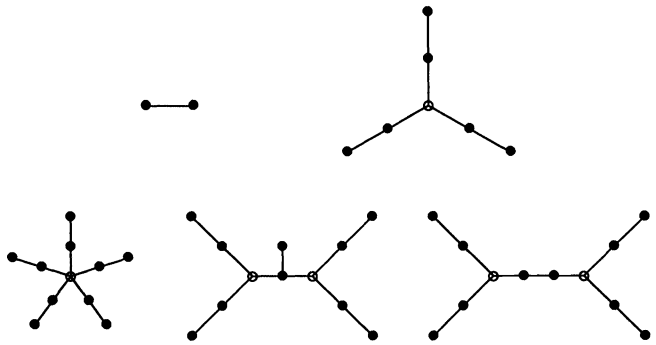


FIG. 3. *The sets* $\mathbf{H}_1$, $\mathbf{H}_2$, *and* $\mathbf{H}_3$.

u partitioned so that $U = \{u\}$ and $W = V(P) - \{u\}$. Then the *augmentation of* G *at* v is the tree G' obtained from $G \cup P$ by (1) adding the edge uv if $v \in W$ or (2) identifying u, v if $v \in U$ (the combined vertex remains in U). Let $\mathbf{H}_1 = \{K_2\}$ with both vertices in W, and let $\mathbf{H}_k$ be the collection of all augmentations of trees in $\mathbf{H}_{k-1}$. The graphs in $\mathbf{H}_1$, $\mathbf{H}_2$, and $\mathbf{H}_3$ appear in Figure 3, with the vertices of U indicated by open dots and those of W by closed dots. The graph in $\mathbf{H}_2$ is the unique forbidden subtree for trees that are interval graphs [10]. The three graphs of $\mathbf{H}_3$ correspond to augmentations at the three isomorphism classes of vertices of the graph in $\mathbf{H}_2$.

We need one more concept describing the relationship between vertices and trail covers: we say that a trail cover $\mathbf{C}$ *swallows* v if v is an internal vertex of some trail in $\mathbf{C}$. Note that a minimum trail cover may end at v and swallow v, but it cannot isolate v and swallow v.

THEOREM 4.2. *If* $G \in \mathbf{H}_k$, *with* $V(G) = U \cup W$ *as described in the construction of* $\mathbf{H}_k$, *then* G *is* k*-critical, the set of essential vertices in* G *is* W, *and the set of useful vertices in* G *is* U. *Furthermore, every minimum trail cover of* G *swallows every vertex of* U.

Proof. This is proved by induction on k. The claim holds by inspection for $k = 1$. Suppose that $k > 1$ and that G is the augmentation of $G' \in \mathbf{H}_{k-1}$ at v. We will apply Algorithm 3.1 to (G, u), where $u = v$ if $v \in U$ and u is the new neighbor of v (on P) if $v \in W$.

Case 1. $v \in W$. In this case, u has exactly three neighbors. The subtrees for the algorithm are (G', v) and two copies of K_2. By the induction hypothesis, v is essential in G', so $c(G', v) = 3$. The code for any vertex of K_2 is also 3. The algorithm therefore yields $t(G) = 1 + t(G') = k$ and $c(G, u) = 2$. Hence $u \in U$, and trails in the subtrees can be extended or shifted in such a way that a degenerate trail is left at any other vertex of P to prove $V(P) - \{u\} \subset W$. Furthermore, if some minimum trail cover does not swallow u, then the pendant edges of P require two trails wholly contained in P, forcing an impossible trail cover of G' with size $k - 2$.

The induction hypothesis guarantees that vertices of U in G' are at least useful in G' and vertices of W in G' are essential in G'. A minimum trail cover establishing this for a particular vertex of G' can be combined with P to obtain a minimum trail cover establishing the same condition for this vertex in G. Similarly, for any edge e in G', we can combine a minimum trail cover of $G' \cdot e$ with P to show that e is not contractible in G. If we contract any edge of P, then the algorithm applied at u will use only $k - 1$ trails, since one of the subtrees becomes K_1. If we contract the edge uv, then we can replace the degenerate trail (v) by the trail P in some minimum trail cover of G' to obtain $k - 1$ disjoint trails covering $G \cdot uv$.

We have shown that G is k-critical; it remains only to show that a vertex $w \in U \cap V(G')$ is not essential in G and is swallowed by any minimum trail cover. By the induction hypothesis, a trail cover $\mathbf{C}'$ of G' that isolates w or does not swallow w has at least k trails. If $\mathbf{C}$ is a minimum trail cover of G that isolates w or does not swallow w, then restricting these trails to G' creates such a trail cover $\mathbf{C}'$ of G'. However, the pendant edges of P require $\mathbf{C}$ to have a trail that contains no vertex of G'. Hence $|\mathbf{C}| \geq k+1$, and the assumption on $\mathbf{C}$ and w was impossible.

Case 2. $v \in U$. In this case $u = v$. The subtrees for applying the algorithm to (G, u) are the same as for applying it to (G', v), plus two copies of K_2. If β is the value of $k_2 + k_3$ for (G, u) and β' is its value for (G', u), then $\beta = \beta' + 2$, since vertices of K_2 have code 3. By the induction hypothesis, v is useful in G', so $c(G', v) = 2$. The algorithm therefore yields $t(G) = 1 + t(G') = k$ and $c(G, u) = 2$. Hence $u \in U$, and again trails in the subtrees can be extended or shifted in such a way that a degenerate trail is left at any other vertex of P to prove $V(P) - \{u\} \subset W$. Furthermore, if some minimum trail cover does not swallow u, then the pendant edges of P require two trails wholly contained in P. This leaves a trail cover of G' of size $k-2$ unless one of the trails in P ends at u, in which case we have a trail cover of G' of size $k-1$ that isolates v; both cases are forbidden by the induction hypothesis.

The remainder of the proof for this case is identical to the last two paragraphs of the proof for Case 1, except that the last sentence of the first paragraph (on contracting the edge uv) is unnecessary and should be deleted, the second paragraph applies only to $w \in (U \cap V(G') - \{v\}$, and the last two sentences of the second paragraph should be replaced by the following: "However, the pendant edges of P require $\mathbf{C}$ to have a trail that contains no vertex of G' except possibly v. This forces $|\mathbf{C}| > |\mathbf{C}'|$ unless v is a degenerate trail in $|\mathbf{C}'|$, in which case $|\mathbf{C}'| \geq k$ because $c(G', v) = 2$. In either case we have $|\mathbf{C}| > k$, and the assumption on $\mathbf{C}$ and w was impossible." □

To complete the characterization of the critical trees, we need only show that every critical tree arises in this way.

THEOREM 4.3. *For each $k \geq 1$, the set of k-critical trees is $\mathbf{H}_k$.*

Proof. We need only show that every k-critical tree is in $\mathbf{H}_k$; we use induction on k. The claim is immediate for $k = 1$; suppose $k > 1$ and G is k-critical. Let P be a longest path in G. Since penultimate vertices in critical trees are bivalent, a critical tree with longest path having fewer than five vertices is only a path. Hence we may assume $P = (u, v, w, x, \ldots, z)$. We have $d(v) = 2$. If $d(w) = 2$, then uv is contractible. If w is incident to a pendant edge e (not in P), then e is contractible. Hence $d(w) \geq 3$ and every neighbor of w except (possibly) x is penultimate (since P is a longest path).

Let $r = d(w) - 1$, let $v_1, \ldots, v_r$ be the neighbors of w other than x, and let $u_1, \ldots, u_r$ be the leaves adjacent to them. When using Algorithm 3.1 on (G, x), we build a minimum covering of the subtree rooted at w by concatenating trails from the v_i in pairs. In particular, the path $Q = (v_1, w, v_2)$ is a trail in some minimum covering $\mathbf{C}$ of G. Let $G' = G - \{u_1, v_1, u_2, v_2\}$ if $r > 2$, and let $G' = G - \{u_1, v_1, u_2, v_2, w\}$ if $r = 2$. Since Q does not extend to x when the algorithm is applied to x, $\mathbf{C} - \{Q\}$ is a minimum covering of G'. The k-criticality of G then implies that G' is $(k-1)$-critical, which by the induction hypothesis implies $G' \in \mathbf{H}_{k-1}$. To show that G is an augmentation of G' and thus that $G \in \mathbf{H}_k$, we consider three cases for the value of r.

First, r cannot be odd. In this case, the algorithm for (G, x) builds a trail rising from w to x. In $(G' \cdot wx, x')$, where x' denotes the combined vertex, the same trail results, with the edge wx contracted. Hence the trail cover number computed by the algorithm is the same for G and $G \cdot wx$, contradicting the criticality of G.

If $r = 2$, we show that $x \in V(G')$ is in W in the canonical partition of G', and hence G is the augmentation of G' at x. If $x \in U$, then by Theorem 4.2, G' has no

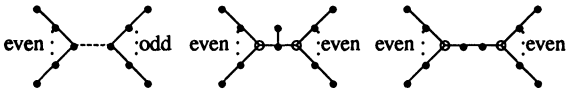


FIG 4. *Trees with* $f = 1$.

trail cover of size $k-1$ that isolates x. This means that the trail ending at x in a minimum trail cover of G' cannot be extended to touch both u_1v_1 and u_2v_2 in $G \cdot wx$. As a result, $t(G \cdot wx) = k$ and wx is contractible, contradicting the criticality of G.

If r is even and $r > 2$, then w has penultimate neighbors in G'. By Lemma 4.1, w cannot be essential in G'. By Theorem 4.2, this implies $w \in U$ in the canonical partition of G', and hence G is the augmentation of G' at w. □

5. Applications. We close this paper by using our results for trees to give alternative proofs of two results that appeared earlier. These are the maximum value of $I(G)$ when G is an n-vertex tree (Aigner and Andreae [1]) and the maximum value of $I(G)$ when G is a connected graph with m edges (Kratzke and West [7]). We use the notation and concepts of the earlier sections together with the auxiliary function $f(G) = n(G) - 4t(G) + 1$.

COROLLARY 5.1. *If G is a tree with $n \geq 3$ vertices, then $I(G) \leq (5n-3)/4$.*

Proof. It suffices to show that $f(G) \geq 0$ for every tree G. If not, let G be a smallest tree such that $f(G) < 0$. If the contraction of some edge does not decrease the trail cover number, then it decreases f by one. Hence G must be critical, which implies $G \in \cup\mathbf{H}_k$. Each such graph is obtained by a sequence of augmentations from K_2. An augmentation at a vertex of U increases f by one, and an augmentation at a vertex of W leaves f unchanged. Although $f(K_2) = -1$, the tree K_2 has no vertex in W, so the first augmentation changes f to 0, and thereafter $f \geq 0$ for each graph in $\cup\mathbf{H}_k$. □

A closer look at this yields a "procedure" for computing f and a description of the n-vertex trees with maximum total interval number equal to $\lfloor (5n-3)/4 \rfloor$. Aigner and Andreae [1] presented trees achieving the bound. A tree G achieves this value if and only if $f(G) \leq 3$. For $G \in \mathbf{H}_k$, the size of U in the canonical partition of G is the number of augmentations at black vertices in the construction of G from K_2. Hence we can determine $f(G)$ for an arbitrary tree G as follows: (0) Initialize $f = -1$. (1) While the remaining graph has a contractible edge, contract it and increase f by 1. (2) When the remaining graph has no contractible edge, determine its canonical partition, increase f by $|U|$, and terminate.

Reworded, this means that the trees with $f(G) = k$ are (1) the trees having a contractible edge e such that $f(G \cdot e) = k-1$ and (2) the critical trees having a canonical partition U, W with $k+1$ vertices in U. Using this and the fact that K_2 is the only tree with $f(G) = -1$, it is not hard to describe explicitly the trees with $f \leq 1$. The trees $\mathbf{F}$ with $f = 0$ are obtained by subdividing every edge of a star that has an odd number of edges. (The first of these is P_3 and is not critical.) The trees with $f = 1$ consist of those obtained by adding a pendant (contractible) edge to a tree with $f = 0$ and those in the three families in Figure 4, where the dashed central edge on the left is contractible and the open circles indicate vertices of U.

In [7], we proved by ad hoc inductive methods that $I(G) \leq (5m+2)/4$ for every connected graph with m edges. Our results for trees yield the bound more cleanly. Since $n(G) = m(G) + 1$ when G is a tree, we have $I(G) = (5m+2)/4$ when G is a tree with $f(G) = 0$. Here we prove that all other connected graphs with at least two edges have a smaller total interval number. In fact, we bound the depth-2 total

interval number, again by using trail covers.

THEOREM 5.2. *The extremal trees are the unique connected graphs having the maximum total interval number in terms of size. In particular, if G is a connected graph with $m \geq 2$ edges and $G \notin \mathbf{F}$, then $I(G) \leq I_2(G) < (5m+2)/4$.*

Proof. We use induction on the number of cycles in G. If G is a tree, then the result follows from the characterization of the trees with $f = 0$. If G has a cycle, we alter G in a way that reduces the number of cycles without changing the number of edges or reducing the trail cover number.

Given distinct edges uv and vw in G, define *snipping* uv to mean deleting uv and subdividing vw. To see that snipping uv does not reduce the trail cover number, let x be the new vertex in the resulting graph G'. A trail cover of G' can be turned into a trail cover of G with the same size by contracting vx.

If G has an edge e that belongs to a cycle of G but not to every cycle of G, then snipping e leaves a graph G' that still has a cycle, and the induction hypothesis yields the result. Otherwise, G has exactly one cycle, and snipping any edge of this cycle yields a tree G'. If $f(G') > 0$, then again we are finished. Hence we may assume that every snip of any edge on the unique cycle in G yields the unique m-edge tree G' with $f(G') = 0$. However, this is impossible; if snipping by deleting uv and subdividing vw yields the extremal graph, then snipping by deleting vw and subdividing uv does not. □

REFERENCES

[1] M. AIGNER AND T. ANDREAE, *The total interval number of a graph*, J. Combin. Theory Ser. B, 46 (1989), pp. 7–21.

[2] A. A. BERTOSSI, *The edge Hamiltonian path problem is NP-complete*, Inform. Process. Lett., 13 (1981), pp. 157–159.

[3] M. R. GAREY, D. S. JOHNSON, AND R. E. TARJAN, *The planar Hamiltonian circuit problem is NP-complete*, SIAM J. Comput., 5 (1976), pp. 704–714.

[4] J. R. GRIGGS AND D. B. WEST, *Extremal values of the interval number of a graph*, SIAM J. Alg. Disc. Meth., 1 (1980), pp. 1–7.

[5] F. HARARY AND C. ST. J. A. NASH-WILLIAMS, *On eulerian and hamiltonian graphs and line graphs*, Canad. Math. Bull., 8 (1965), pp. 701–710.

[6] T. M. KRATZKE, *The Total Interval Number of a Graph*, Ph.D. thesis, University of Illinois, Urbana, IL, 1987; Coordinated Science Laboratory Research report UILU-ENG-88-2202, 1988.

[7] T. M. KRATZKE AND D. B. WEST, *The total interval number of a graph* I: *Fundamental classes*, Discrete Math., 118 (1993), pp. 145–156.

[8] P. G. H. LEHOT, *An optimal algorithm to detect a line-graph and output its root graph*, J. Assoc. Comput. Mach., 21 (1974), pp. 569–575.

[9] D. B. SHMOYS AND D. B. WEST, *Recognizing graphs with fixed interval number is NP-complete*, Discrete Appl. Math., 8 (1984), pp. 295–305.

[10] W. T. TROTTER AND F. HARARY, *On double and multiple interval graphs*, J. Graph Theory, 2 (1978), pp. 137–142.

SIAM J. DISCRETE MATH.
Vol. 9, No. 3, pp. 349–359, August 1996

ANGLES OF PLANAR TRIANGULAR GRAPHS*

GIUSEPPE DI BATTISTA† AND LUCA VISMARA‡

Abstract. We give a characterization of all the planar drawings of a triangular graph through a system of equations and inequalities relating its angles; we also discuss minimality properties of the characterization. The characterization can be used: (1) to decide in linear time whether a given distribution of angles between the edges of a planar triangular graph can result in a planar drawing; (2) to reduce the problem of maximizing the minimum angle in a planar straight-line drawing of a planar triangular graph to a nonlinear optimization problem purely on a space of angles; (3) to give a characterization of the planar drawings of a triconnected graph through a system of equations and inequalities relating its angles; (4) to give a characterization of Delaunay triangulations through a system of equations and inequalities relating its angles; (5) to give a characterization of all the planar drawings of a triangular graph through a system of equations and inequalities relating the lengths of its edges; in turn, this result allows us to give a new characterization of the disc-packing representations of planar triangular graphs.

Key words. graph drawing, planar graph, Delaunay triangulation, angle

AMS subject classifications. 05C10, 05C40, 05C85, 68R10, 68U05

1. Introduction.

Planar straight-line drawings of planar graphs are a classical topic of the graph drawing field (a survey on graph drawing can be found in [5]).

A classical result independently established by Steinitz and Rademacher [23], Wagner [27], Fary [9], and Stein [22] shows that every planar graph has a planar straight-line drawing.

A grid drawing is a drawing in which the vertices have integer coordinates. Independently, de Fraysseix, Pach, and Pollack [3] and Schnyder [20, 21] have shown that every n-vertex planar graph has a planar straight-line grid drawing with $O(n^2)$ area.

Planar straight-line drawings have also been studied with the constraint that all faces be represented by convex polygons (convex drawings). Tutte [24, 25] has shown that, for a triconnected graph, convex drawings can be constructed by solving a system of linear equations. Recently, Kant has shown an algorithm for constructing grid convex drawings with quadratic area [15, 16].

In the research on planar straight-line drawings a very special role is played by the angles between the segments that compose the drawing. In particular, Vijayan [26] has studied angle graphs. An angle graph is a planar embedded graph in which the angles between successive edges incident with each vertex is given. The problem of the existence of a planar straight-line drawing of an angle graph that preserves the angles is tackled and partial characterization results are shown.

* Received by the editors March 2, 1994; accepted for publication (in revised form) August 11, 1995. This research was supported in part by ESPRIT Basic Research Action 7141 (ALCOM II) and by Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the Consiglio Nazionale delle Ricerche. An extended abstract of this paper appears in the *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, San Diego, CA, 1993.

† Dipartimento di Discipline Scientifiche, Sezione Informatica, Terza Università degli Studi di Roma, Via della Vasca Navale 84, 00146 Roma, Italy (dibattista@iasi.rm.cnr.it). This research was performed in part while this author was with the Dipartimento di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza" and with the Dipartimento di Ingegneria e Fisica dell'Ambiente, Università degli Studi della Basilicata.

‡ Dipartimento di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza," Via Salaria 113, 00198 Roma, Italy (vismara@iasi.rm.cnr.it). This research was performed in part while this author was with the Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche.

In [10], Formann et al. have studied the problem of constructing straight-line drawings of graphs with large angles. It is shown that it is always possible to construct a drawing whose smallest angle between the edges incident with a vertex is $\Omega(1/d^2)$, where d is the maximum vertex degree of the graph. Other results are given for particular classes of graphs. For planar graphs the bound is improved to $\Omega(1/d)$; however, in general, the obtained drawing is nonplanar.

Malitz and Papakostas [17] have shown that it is always possible to construct a planar straight-line drawing of a planar graph whose smallest angle is $\Omega(\alpha^d)$, where $0 < \alpha < 1$ is a constant. They exploit a disc-packing representation of the graph. In a disc-packing representation (1) each vertex is a disc, (2) two discs are tangent if and only if the vertices they represent are adjacent, and (3) the interiors of the discs are pairwise disjoint. No polynomial algorithm is known to construct a disc-packing representation. However, Brightwell and Scheinerman [2] have shown that, in general, it is not possible to construct a disc-packing representation with radii whose lengths are rational numbers. Recently, Mohar [18] has provided a polynomial time algorithm to approximate a disc-packing representation to any specified accuracy.

Garg and Tamassia [12] have shown that there exist planar graphs such that in any of their planar straight-line drawings the smallest angle is $O(\sqrt{\log d/d^3})$.

Garg [11] has proved that testing the planarity of a consistent angle graph (an angle graph for which there exists a straight-line drawing preserving the angles) is an NP-complete problem.

An important tool for several algorithms and characterizations described above are planar triangular graphs. For instance the algorithm by de Fraysseix, Pach, and Pollack and the algorithm by Schnyder have an intermediate step in which the given planar graph is triangulated. Also, planar triangular graphs play a very special role in a number of problems arising in computational geometry. However, as far as we know, characterizing angles of planar triangular graphs has been an elusive goal for a long time. This paper can be summarized as follows.

We give a characterization of all the planar drawings of a triangular graph through a system of equations and inequalities relating its angles. The problem is mentioned as open in [1, 17, 26]. We also discuss minimality properties of the characterization.

The characterization above has several applications.

(i) It can be used to decide in linear time whether a given distribution of angles between the edges of a planar triangular graph can result in a planar drawing.

(ii) It reduces the problem of maximizing the minimum angle in a planar straight-line drawing of a planar triangular graph to a nonlinear optimization problem purely on a space of angles.

(iii) It gives a characterization of the planar drawings of a triconnected graph through a system of equations and inequalities relating its angles.

(iv) It gives a characterization of Delaunay triangulations through a system of equations and inequalities relating its angles, solving a problem stated in [6]. Recently, the problem of deciding whether a plane triangular graph G is Delaunay realizable, i.e., it is combinatorially equivalent to a Delaunay triangulation (DT), has been tackled by Dillencourt and Rivin [8]. They have shown that if a linear system of equations and inequalities relating the angles of G has a feasible solution, then G is Delaunay realizable; in general, however, the angles of the DT differ from the angles obtained by solving the system.

(v) It can be exploited to give a characterization of all the planar drawings of a triangular graph through a system of equations and inequalities relating the lengths of its edges; in turn, this result allows us to give a new characterization of the disc-

packing representations of planar triangular graphs.

The rest of the paper is organized as follows. Preliminaries are in §2. Section 3 gives some basic results. Section 4 contains the characterization and some minimality results. Section 5 highlights some applications. Section 6 contains a discussion of open problems.

2. Preliminaries. We assume familiarity with planar graphs and with graph connectivity [19].

A *drawing* of a graph maps each vertex to a distinct point of the plane and each edge (u, v) to a simple Jordan curve with end-points u and v. A drawing is *planar* if no two edges intersect, except, possibly, at common end-points. A graph is planar if it has a planar drawing. A *straight-line* drawing is a drawing such that each edge is mapped to a straight-line segment.

Two planar drawings of a planar graph G are *equivalent* if, for each vertex v, they have the same circular clockwise sequence of edges incident with v. Hence, the planar drawings of G are partitioned into equivalence classes. Each of those classes is called an *embedding* of G. An *embedded* planar graph (also *plane* graph) is a planar graph with a prescribed embedding. A planar triconnected graph has a unique embedding, up to a reflection. A planar drawing divides the plane into topologically connected regions delimited by cycles; such cycles are called *faces*. The *external* face is the cycle delimiting the unbounded region. Two drawings with the same embedding have the same faces. Hence, one can refer to the faces of an embedding. We call *internal angles* of a drawing the angles formed by consecutive edges of internal faces.

A *plane triangular* graph is a plane triconnected graph with at least four vertices in which one face is chosen as external and all faces (except possibly the external one) consist of three edges.

A *Delaunay drawing* of a plane triangular graph G is a planar straight-line drawing of G that is a DT. Not all plane triangular graphs have a Delaunay drawing (see, e.g., [6, 7, 13, 14]).

3. How to draw a wheel. A *wheel* W_d is a plane triangular graph with $d+1$ ($d \geq 3$) vertices $v, v_1, \ldots, v_d$ and $2d$ edges: v is connected to $v_1, \ldots, v_d$ and v_i is connected to $v_{(i \bmod d)+1}$. Vertices $v_1, \ldots, v_d$ identify the external face. Vertex v is called *center* of the wheel. A *centered* drawing of W_d is a planar straight-line drawing in which v is drawn in the region bounded by the polygon representing cycle $v_1, \ldots, v_d$. In a drawing of W_d the angles of an internal face are denoted as follows: the angle at vertex v is called the *central* angle and is followed, in clockwise order, by the *left* angle and by the *right* angle. See Fig. 1.

LEMMA 3.1. *Let W_d be a wheel with center v and suppose a set of positive values for the internal angles between the edges of W_d is given. Wheel W_d has a centered drawing Γ with the given angles if and only if the following conditions hold (see Fig. 1(a)).*

1. *Let $\gamma_1, \ldots, \gamma_d$ be the central angles of W_d:*

$$\sum_{i=1}^{d} \gamma_i = 2\pi.$$

2. *For each internal face f of W_d, let α, β, γ be the angles of f:*

$$\alpha + \beta + \gamma = \pi.$$

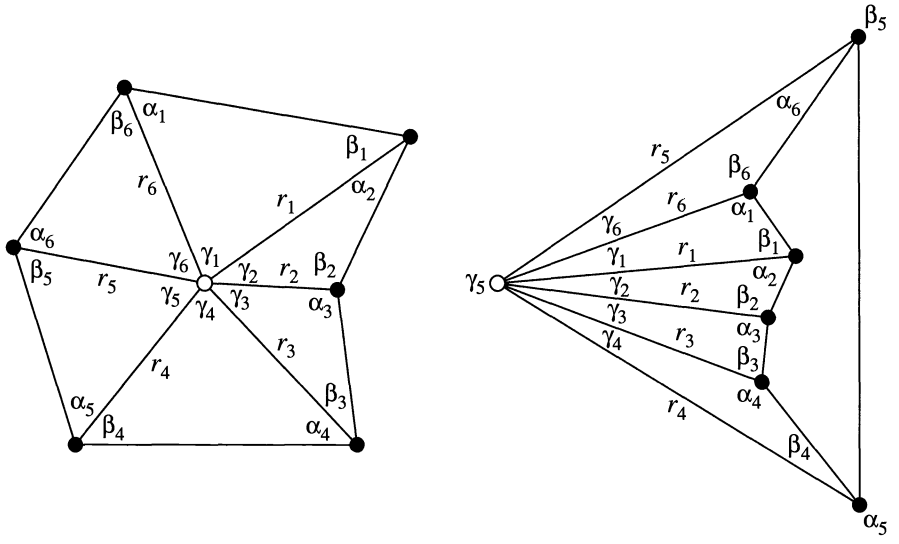


FIG. 1. (a) *a centered drawing and* (b) *a noncentered drawing of a wheel.*

3. *Let* $\alpha_1, \ldots, \alpha_d$ *and* $\beta_1, \ldots, \beta_d$ *be the left and right angles of* W_d*, respectively:*

$$\prod_{i=1}^{d} \frac{\sin \alpha_i}{\sin \beta_i} = 1.$$

Proof. This is proved only if conditions 1 and 2 are trivial. Let Γ be a centered drawing of W_d and let r_i be the length of the edge e_i between angles β_i and $\alpha_{(i \bmod d)+1}$ in Γ. We have $\prod_{i=1}^{d} \frac{r_{(i \bmod d)+1}}{r_i} = 1$. By the triangle sine law, $\frac{r_{(i \bmod d)+1}}{r_i} = \frac{\sin \alpha_{(i \bmod d)+1}}{\sin \beta_{(i \bmod d)+1}}$ $(i = 1, \ldots, d)$; condition 3 follows.

If. In the proof we construct Γ. Let r_i be the length of the edge e_i between angles β_i and $\alpha_{(i \bmod d)+1}$. First, we choose arbitrarily r_1; then, we compute r_{i+1} from r_i $(i = 1, \ldots, d-1)$ by using the triangle sine law with the prescribed angles; namely, $r_{i+1} = r_i \cdot \frac{\sin \alpha_{i+1}}{\sin \beta_{i+1}}$. At each step, condition 2 allows the construction of a new triangle having edge e_i in common with the previous one, and condition 1 guarantees that triangles do not overlap. We have $r_d = r_1 \cdot \prod_{i=2}^{d} \frac{\sin \alpha_i}{\sin \beta_i}$. By using condition 3 we obtain $\frac{r_1}{r_d} = \frac{\sin \alpha_1}{\sin \beta_1}$; that is lengths r_1 and r_d satisfy the sine law for the triangle with angles α_1, β_1, and γ_1. Thus, we can complete Γ by adding a segment between the end-vertices of e_1 and e_d different from v.

From the construction above, it follows that v is drawn inside the polygon identified by the other vertices. Further, the prescribed angles being positive, there is no overlapping between vertices and/or edges. □

4. Characterizing the planar drawings of plane triangular graphs. In this section we give the main theorem (Theorem 4.6) of the paper. In order to prove this theorem, we give some preliminary lemmas.

In the following lemma, an *external path* of a graph G is a proper subpath p of the external face of G such that (a) it contains at least one edge; (b) all its vertices are adjacent to a common internal vertex of G; (c) its vertices different from the end-vertices are adjacent to no other internal vertex of G. To *remove* an external path p

from a graph G means to remove from G all the edges of p, all the vertices of p except the end-vertices, and all the remaining dangling edges.

In Lemmas 4.1 and 4.4 our definition of plane triangular graph as a plane triconnected graph plays a crucial role.

LEMMA 4.1. *Let G be a plane triangular graph. There exists an external path p of G, such that the graph obtained by removing p from G is a plane triangular graph or the triangle graph.*

Proof. Let c be the external face of G. If G is a wheel, then p can be chosen as any subpath of c containing all the edges of c except one. The graph obtained by removing p from G is the triangle graph.

If G is not a wheel, then we proceed as follows: we look for p in a subpath (called *candidate path*) of c with at least one edge. At the beginning of the search the candidate path is suitably chosen. At each step two cases are possible: (1) p is immediately found as a subpath of the candidate path; (2) the candidate path is restricted to a proper subpath of itself.

For the candidate path the following invariant holds: let u and w be the end-vertices of the candidate path; u and w are adjacent to a common internal vertex v, while all the other vertices of the candidate path, if any, are not adjacent to v; further, the graph delimited by (v, u), by (v, w), and by the candidate path has at least one internal vertex.

At the first step there exists at least one subpath of c with the invariant property defined above, otherwise G would be a wheel.

Let e be an edge of the candidate path and let v' be the internal vertex opposite to e. Vertex v' is an internal vertex of the graph delimited by (v, u), by (v, w), and by the candidate path. As explained above, two cases are possible.

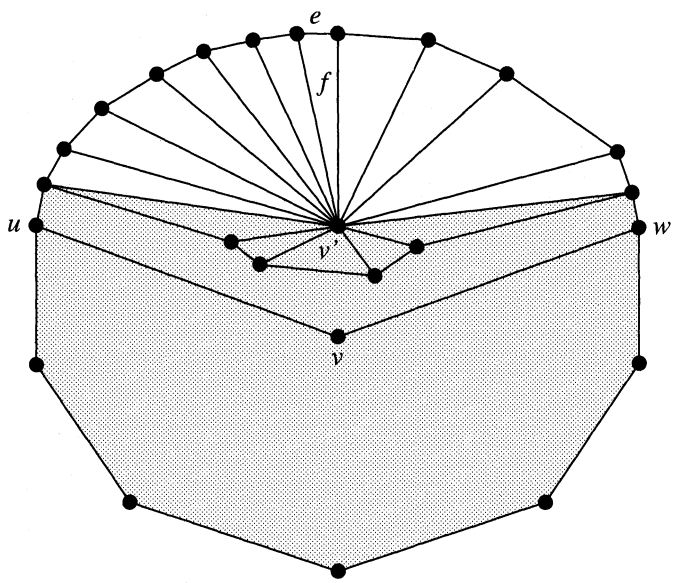


FIG. 2. *Illustration of Case* 1 *in the proof of Lemma* 4.1.

Case 1. The edges that connect v' to the vertices of the candidate path are consecutive in the adjacency list of v' (see Fig. 2). Then, p is the subpath of the candidate path identified by the end-vertices, distinct from v', of such edges. Path

p can be removed from G without losing the triconnectivity property: in fact, no separation pair can be generated by the removal of p, because, by planarity, the only external vertices of G adjacent to v' are those contained in p. Note that p may consist of just u, w, and $e = (u, w)$.

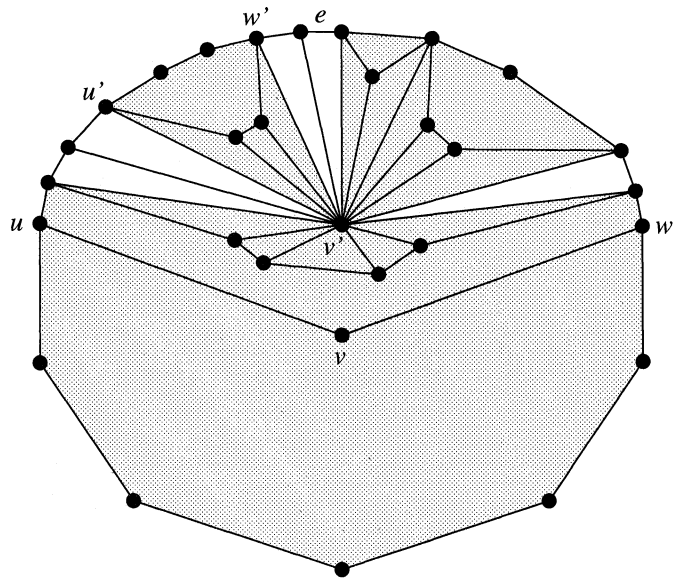


FIG. 3. *Illustration of Case* 2 *in the proof of Lemma* 4.1.

Case 2. The edges that connect v' to the vertices of the candidate path are non-consecutive in the adjacency list of v' (see Fig. 3). Let A be the set of the edges connecting v' to the vertices of the candidate path. Consider an ordering for the vertices of the candidate path, say from u to w; it induces an ordering for the edges of A. Let (v', u') and (v', w') be the first two consecutive edges in A such that there exists at least one edge (v', x) between them in the adjacency list of v'. We consider, as the new candidate path, the proper subpath of the current one between u' and w'. In fact, it satisfies the invariant of a candidate path: u' and w' are adjacent to the internal vertex v', the other vertices, if any, are not adjacent to v', and there exists at least one internal vertex x in the graph delimited by (v', u'), by (v', w'), and by the new candidate path.

Note that if the candidate path contains just one edge, Case 1 applies. □

Operation *close-wheel* with *center* v is defined as follows. Let u, v, and w be three consecutive vertices of the external face of a plane triangular graph G. Operation close-wheel adds k new vertices and $2k + 1$ new edges ($k \geq 0$) to G. If $k = 0$ then close-wheel adds to G edge (u, w). If $k > 0$ then close-wheel adds to G vertices $v_1, \ldots, v_k$, edges $(u, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k), (v_k, w)$, and edges $(v, v_1), \ldots, (v, v_k)$.

LEMMA 4.2. *Let G be a plane triangular graph with n vertices. G can be constructed, starting from the triangle graph, by a sequence of $O(n)$ close-wheel operations. Each intermediate graph is a plane triangular graph.*

Proof. The lemma easily descends from the definition of operation close-wheel, from Lemma 4.1, and from the definition of external path. Further, each close-wheel adds at least one edge. □

In the following corollary we define a new ordering of the vertices of a maximal

plane graph, called *wheel ordering.*

COROLLARY 4.3. *Let G be an n-vertex maximal plane graph, with external vertices x, y, and z. There exists an ordering of the vertices of G such that*

1. $v_1 = x, v_2 = y$;
2. v_3 *is the internal vertex opposite to* (x, y);
3. *for* $k = 4, \ldots, n-1$,

(i) *subgraph* $G_{k-1} \subset G$ *formed by the union of the wheels with centers* $v_3, \ldots, v_{k-1}$ *is a plane triangular graph, and its exterior face is a cycle* C_{k-1} *containing* (x, y);

(ii) *vertex* v_k *is the vertex of* C_{k-1} *center of the* k*th close-wheel operation*;

4. $v_n = z$.

A drawing Γ of a plane graph G is called *embedding-preserving* if, for each vertex v of G, the edges incident with v appear in Γ in the same circular order they have in the adjacency list of v in G. The external face of G is *drawn convex* if it is represented as a (not necessarily simple) polygon p such that, for each vertex v that is found while walking around p, the angle on, say, the left side is greater than π and all the edges incident with v are on the right side.

LEMMA 4.4. *Let G be a plane triangular graph and let Γ be an embedding-preserving straight-line drawing of G such that the external face of G is drawn convex. There exists an internal vertex v of G such that the wheel with center v is drawn centered in Γ.*

Proof. Let $v_1, \ldots, v_d$ be the vertices adjacent to v in this circular order around v.

If v is the only internal vertex (i.e., G is a wheel) or v is not the only internal vertex but the wheel with center v is drawn centered in Γ, then the lemma is trivial.

Suppose now that G is not a wheel and that the wheel with center v is drawn noncentered in Γ; it follows that one of the angles around v, say the one between (v, v_1) and (v, v_d), is greater than π; see, e.g., Fig. 1(b).

If for each vertex v_i $(i = 2, \ldots, d-1)$ the sum of the angle between (v_i, v_{i-1}) and (v_i, v) and the angle between (v_i, v) and (v_i, v_{i+1}) is less than π, then, by easy geometric considerations, vertex v and vertices $v_2, \ldots, v_{d-1}$ are drawn in Γ on different sides of edge (v_1, v_d). This violates the embedding at vertices v_1 and v_d; thus, for at least one vertex v_i $(i = 2, \ldots, d-1)$ the sum of the angle between (v_i, v_{i-1}) and (v_i, v) and the angle between (v_i, v) and (v_i, v_{i+1}) is greater than π. It follows that vertex v_i is drawn in the (open) triangular area with vertices v, v_{i-1}, v_{i+1}. Hence, no angle around v_i can be greater than π; it follows, by the convexity of the external face, that v_i cannot be an external vertex and that the wheel with center v_i is drawn centered in Γ. □

LEMMA 4.5. *Let G be a plane triangular graph and let Γ be a straight-line drawing of G such that the external face of G is drawn convex. If Γ is embedding-preserving then it is planar.*

Proof. The proof is by induction on the number n_i of internal vertices of G. If $n_i = 1$, i.e., G is a wheel, by Lemma 4.4 Γ is a centered drawing of G and the planarity of Γ easily follows from the embedding-preserving property. Assume that the lemma holds for graphs with $n_i - 1$ internal vertices. Let v be an internal vertex of G such that the wheel W_d with center v is drawn centered in Γ (see Lemma 4.4). We remove v and its incident edges from G. Let G' be the new graph, and let Γ' be the straight-line drawing obtained from Γ by removing the point and the segments representing v and its incident edges. If $d \geq 4$, we triangulate the inside of the star-shaped polygon representing the external cycle of W_d by adding new segments to Γ'; let Γ'' be the new straight-line drawing. By adding the corresponding edges to G', we obtain a plane triangular graph G'' with $n_i - 1$ internal vertices; Γ'' is its

embedding-preserving straight-line drawing (with the same external face of Γ). By using the inductive hypothesis, we argue that Γ'' is planar. It follows that Γ' is planar too and so is the union of Γ' and the centered drawing of W_d. □

THEOREM 4.6. *Let G be a plane triangular graph. Suppose a set of positive values for the internal angles between the edges of G is given. G has a planar straight-line drawing Γ with the given angles and with the given external face drawn convex if and only if the following conditions hold.*

1. *For each internal vertex v of G, let $\gamma_1, \ldots, \gamma_d$ be the angles around v, where d is the degree of v:*

$$\sum_{i=1}^{d} \gamma_i = 2\pi.$$

2. *For each internal face f of G, let α, β, γ be the angles of f:*

$$\alpha + \beta + \gamma = \pi.$$

3. *For each internal vertex v of G, let W_d be the wheel with center v and let $\alpha_1, \ldots, \alpha_d$ and $\beta_1, \ldots, \beta_d$ be the left and right angles of W_d, respectively:*

$$\prod_{i=1}^{d} \frac{\sin \alpha_i}{\sin \beta_i} = 1.$$

4. *For each external vertex v of G, the sum of the internal angles of v is less than π.*

Proof. This is proved only if condition 4 is trivial. Each internal vertex of G forms a wheel with its adjacent vertices; hence, necessity of conditions 1, 2, and 3 follows from Lemma 3.1.

If. We give a constructive proof. Namely, starting from a triangle, we construct Γ by means of a sequence of close-wheel operations, each one adding to Γ at least one edge. The completeness of the approach is guaranteed by Lemma 4.2. Consider the operation close-wheel performed with center v; let W_d be the wheel with center v; let u and w be the neighbors of v on the external face of the part of G drawn so far.

At each close-wheel operation we complete the drawing of W_d as in the proof of Lemma 3.1.

Once G has been completely drawn with the described above strategy, Γ is a straight-line drawing with the prescribed angles. It remains to be proved that Γ is planar. This is done by observing that

1. Each close-wheel operation preserves the circular orderings of the adjacency lists of u, v, and w, given by the embedding; this follows from conditions 1 and 4.
2. The external face of G is drawn convex; this follows from condition 4.

Hence, to prove planarity, we use Lemma 4.5. □

COROLLARY 4.7. *Let G be a plane triangular graph with n vertices. Suppose a set of positive values for the internal angles between the edges of G is given. There exists an $O(n)$ time and space algorithm to test if G has a planar straight-line drawing with the given angles and with the given external face drawn convex.*

We now study the minimality of the characterization. Namely, we answer the question if some of the equations of condition 3 of Theorem 4.6 can be omitted without losing the correctness of the characterization. As shown in the following theorem, the answer is negative. In general, all the equations of condition 3 are needed.

THEOREM 4.8. *There exists a plane triangular graph $\bar{G}$, an internal vertex v of $\bar{G}$, and a set of positive values for the internal angles between the edges of $\bar{G}$ such that* (1) *conditions* 1, 2, *and* 4 *of Theorem* 4.6 *hold and condition* 3 *holds for each internal vertex different from v;* (2) *no planar straight-line drawing of $\bar{G}$ exists with the given angles.*

Proof. Let $\bar{G}$ be the graph of Fig. 4, where v is the white vertex and the angles are those specified in the figure. Observe that conditions 1, 2, and 4 are satisfied, and that condition 3 does not hold for v while it holds for any other internal vertex. Hence, the wheel with center v cannot be drawn with the given angles. □

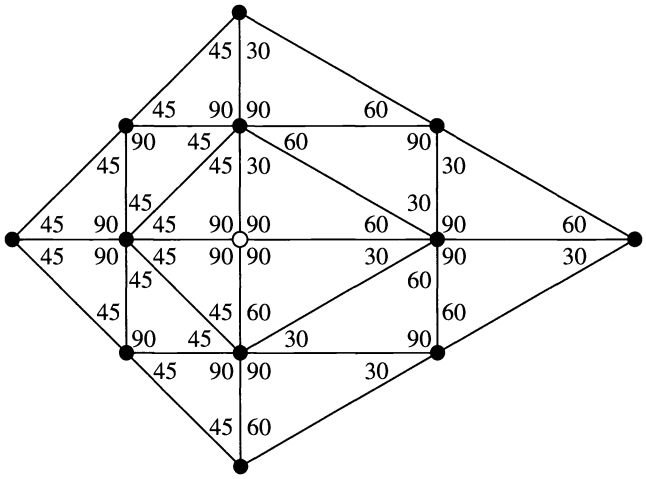


FIG. 4. *Graph $\bar{G}$ in the proof of Theorem* 4.8.

5. Some applications. By applying Theorem 4.6, it is possible to set up a nonlinear optimization problem in order to maximize the minimum angle in a planar straight-line drawing of a given plane triangular graph. This can be done by declaring each variable representing an angle greater than or equal to a new variable α_0 and by maximizing α_0 [1].

Theorem 4.6 can be used to characterize the convex drawings of a triconnected graph. Namely, let G be a plane triconnected graph; let G' be the graph obtained by inserting a dummy vertex v inside each nontriangular internal face f of G and by connecting v to the vertices of f. A characterization of the angles of the convex drawings of G is easily obtained by applying the conditions of Theorem 4.6 on G' and by supplementing them with conditions that guarantee the convexity of the nontriangular faces.

Theorem 4.6 can also be used to characterize DTs.

THEOREM 5.1. *Let G be a plane triangular graph. Suppose a set of positive values for the internal angles between the edges of G is given. G has a Delaunay drawing Γ with the given angles and with the given external face if and only if conditions* 1, 2, 3, *and* 4 *of Theorem* 4.6 *and the following condition hold:*

5. *For each internal edge e of G, let ϵ_1 and ϵ_2 be the angles opposite to e in the two triangular faces of G incident with e:*

$$\epsilon_1 + \epsilon_2 < \pi.$$

Proof. The theorem descends from Fact 2.1 of [7] (or from [4]) and from Theorem 4.6. □

Another interesting aspect of Theorem 4.6 is that it can be exploited to characterize the planar drawings of plane triangular graphs through the lengths of their edges. The proof of the following theorem is analogous to the proof of Theorem 4.6 and makes use of the triangle cosine law.

THEOREM 5.2. *Let G be a plane triangular graph. Suppose a set of positive values for the lengths of the edges of G is given. G has a planar straight-line drawing Γ with the given lengths for the edges and with the given external face drawn convex if and only if the following conditions hold.*

1. *For each internal face f of G, let a, b, c be the lengths of the edges of f: $a < b + c$, $b < c + a$, and $c < a + b$.*

2. *For each internal vertex v of G, let W_d be the wheel with center v. Let $c_1, \ldots, c_d$ be the lengths of the edges of the external cycle of W_d and let $r_1, \ldots, r_d$ be the lengths of the other edges of W_d:*

$$0 < \arccos \frac{{r_i}^2 + {r_{(i \bmod d)+1}}^2 - {c_i}^2}{2 r_i r_{(i \bmod d)+1}} < \pi, \qquad i = 1, \ldots, d,$$

$$\sum_{i=1}^{d} \arccos \frac{{r_i}^2 + {r_{(i \bmod d)+1}}^2 - {c_i}^2}{2 r_i r_{(i \bmod d)+1}} = 2\pi.$$

3. *For each external vertex v of G, let $f_1, \ldots, f_{d-1}$ be the internal faces of G sharing v. Consider face f_i, let c_i be the length of the edge of f_i opposite to v, and let r_i, r_{i+1} be the lengths of the other two edges:*

$$\sum_{i=1}^{d-1} \arccos \frac{{r_i}^2 + {r_{i+1}}^2 - {c_i}^2}{2 r_i r_{i+1}} < \pi.$$

The disc-packing representations of a plane triangular graph G are easily characterized by using a variable x_v for each vertex v of G, representing the length of the radius of the disc centered at v, and by adding, for each edge (u, v) with length l, a new condition $x_u + x_v = l$ to the conditions of Theorem 5.2.

6. Open problems. The conditions of Theorem 4.6 are nonlinear. Is it possible to simplify some of them for particular classes of drawings and/or for particular classes of graphs?

Concerning the first possibility, there is a negative result. Namely, consider straight-line drawings such that for each internal edge the two opposite angles are equal. In such drawings condition 3 of Theorem 4.6 is always satisfied and hence can be omitted. Unfortunately, not all the plane triangular graphs admit one of such drawings.

Theorem 4.8 shows that, in general, it is necessary to use condition 3 of Theorem 4.6 for each internal vertex. Are there special classes of graphs that allow us to use condition 3 of Theorem 4.6 on a subset of internal vertices? For example, are there classes of graphs for which such subset is a point cover?

Another issue is to consider how to generalize Theorem 4.6 to nonplanar graphs or to nonconvex drawings of planar graphs (see also [26]).

Acknowledgments. We thank Pierluigi Crescenzi and Adolfo Piperno for useful discussions on the disc-packing problem.

REFERENCES

[1] F. J. Brandenburg, P. Kleinschmidt, and U. Schnieders, *Drawing planar graphs with wide angles*, Manuscript, 1991.

[2] G. R. Brightwell and E. R. Scheinerman, *Representation of planar graphs*, SIAM J. Discrete Math., 6 (1993), pp. 214–229.

[3] H. de Fraysseix, J. Pach, and R. Pollack, *How to draw a planar graph on a grid*, Combinatorica, 10 (1990), pp. 41–51.

[4] B. Delaunay, *Sur la sphère vide*, Izv. Akad. Nauk SSSR, VII Seria, Otdel. Mat. i Estestvennyka Nauk, 7 (1934), pp. 793–800.

[5] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Algorithms for drawing graphs: An annotated bibliography*, Comput. Geom. Theory Appl., 4 (1994), pp. 235–282.

[6] M. B. Dillencourt, *Graph-Theoretical Properties of Algorithms Involving Delaunay Triangulations*, Tech. Report CS-TR-2059, Institute for Advanced Computer Studies, University of Maryland, College Park, MD, 1988.

[7] ———, *Toughness and Delaunay triangulations*, Discrete Comput. Geom., 5 (1990), pp. 575–601.

[8] M. B. Dillencourt and I. Rivin, Personal communication, 1992.

[9] I. Fary, *On straight lines representation of planar graphs*, Acta Sci. Math. (Szeged), 11 (1948), pp. 229–233.

[10] M. Formann, T. Hagerup, J. Haralambides, M. Kaufmann, F. T. Leighton, A. Simvonis, E. Welzl, and G. Woeginger, *Drawing graphs in the plane with high resolution*, SIAM J. Comput., 22 (1993), pp. 1035–1052.

[11] A. Garg, *On drawing angle graphs*, in Graph Drawing, in Proc. DIMACS International Workshop on Graph Drawing, Princeton, NJ, Lecture Notes in Computer Science 894, R. Tamassia and I. G. Tollis, eds., Springer-Verlag, Berlin, New York, 1995, pp. 84–95.

[12] A. Garg and R. Tamassia, *Planar drawings and angular resolution: Algorithms and bounds*, in Algorithms, Proceedings of the 2nd Annual European Symposium on Algorithms, Utrecht, the Netherlands, Lecture Notes in Computer Science 855, J. van Leeuwen, ed., Springer-Verlag, Berlin, New York, 1994, pp. 12–23.

[13] C. D. Hodgson, I. Rivin, and W. D. Smith, *A characterization of convex hyperbolic polyhedra and of convex polyhedra inscribed in the sphere*, Bull. Amer. Math. Soc., 27 (1992), pp. 246–251.

[14] ———, *Erratum (A characterization of convex hyperbolic polyhedra and of convex polyhedra inscribed in the sphere)*, Bull. Amer. Math. Soc., 28 (1993), p. 213.

[15] G. Kant, *Drawing planar graphs using the lmc-ordering*, in Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, PA, 1992, pp. 101–110.

[16] ———, *Drawing planar graphs using the canonical ordering*, Algorithmica (special issue on Graph Drawing, G. Di Battista and R. Tamassia, eds.), to appear.

[17] S. Malitz and A. Papakostas, *On the angular resolution of planar graphs*, SIAM J. Discrete Math., 7 (1994), pp. 172–183.

[18] B. Mohar, *A polynomial time circle packing algorithm*, Discrete Math., 117 (1993), pp. 257–263.

[19] T. Nishizeki and N. Chiba, *Planar Graphs: Theory and Algorithms*, Annals of Discrete Mathematics, Vol. 32, North-Holland, Amsterdam, 1988.

[20] W. Schnyder, *Planar graphs and poset dimension*, Order, 5 (1989), pp. 323–343.

[21] ———, *Embedding planar graphs on the grid*, in Proc. 1st ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1990, pp. 138–148.

[22] S. K. Stein, *Convex maps*, Proc. Amer. Math. Soc., 2 (1951), pp. 464–466.

[23] E. Steinitz and H. Rademacher, *Vorlesungen über die Theorie der Polyeder*, Julius Springer, Berlin, Germany, 1934.

[24] W. T. Tutte, *Convex representations of graphs*, Proc. London Math. Soc., 10 (1960), pp. 304–320.

[25] W. T. Tutte, *How to draw a graph*, Proc. London Math. Soc., 13 (1963), pp. 743–768.

[26] G. Vijayan, *Geometry of planar graphs with angles*, in Proc. 2nd Annual ACM Symposium on Computational Geometry, Yorktown Heights, NY, 1986, pp. 116–124.

[27] K. Wagner, *Bemerkungen zum vierfarbenproblem*, Jahresber. Deutsch. Math.-Verein., 46 (1936), pp. 26–32.

SIAM J. DISCRETE MATH.
Vol. 9, No. 3, pp. 360–364, August 1996

002

CONSTRAINED EMBEDDING PROBABILITY FOR TWO BINARY STRINGS*

JOVAN DJ. GOLIĆ†

Abstract. An exponentially small upper bound on the probability that a given binary string of length n can be embedded into a uniformly distributed random binary string of length $2n$ by inserting at most one bit between any two successive bits and an arbitrary number of bits at the end is analytically derived. This probability is important for a cryptanalytic problem of the initial state reconstruction of a binary clock-controlled shift register that is clocked either once or twice per each output symbol, given a segment of its output sequence. The developed approach may also be interesting for other problems of sequence comparison as well, especially for the codes for correcting synchronization errors.

Key words. sequence comparison, constrained embedding, recurrences, cryptanalysis, synchronization error-correcting codes

AMS subject classifications. 05A15, 11B37, 94A60, 94B50

1. Introduction.

A cryptanalytic problem of the initial state reconstruction of a binary clock-controlled shift register that is clocked either once or twice per each output symbol is considered in [5], assuming that its output sequence is known. A more general case when the register is additively noised and the maximum number of consecutive clocks at a time is an arbitrary positive integer is examined in [3] using a generalization of the Levenshtein distance. In the zero-noise case the reconstruction is successful if the probability that a given binary string can be embedded into a random binary string of appropriate length decreases sufficiently fast with the string length. An exponentially decreasing upper bound on the embedding probability derived in [5] shows that the required length of the output sequence is linear in the shift register length. The bound is obtained by direct counting based on some theoretical considerations. In this paper, the underlying combinatorial problem is solved analytically and an upper bound on the probability that a given binary string of length n can be embedded into a random binary string of length $2n$ by inserting at most one bit between any two successive bits and as many bits as needed at the end is thus derived. The bound is the tightest in the observed class and hence sharper than the one from [5]. The combinatorial problems of this kind are also relevant for the codes capable of correcting synchronization errors; see [1], [2], for example.

2. Preliminaries.

Consider two binary strings $X = \{x_i\}_{i=1}^n$ and $Y = \{y_i\}_{i=1}^m$ of lengths n and m, respectively. For an arbitrary binary string X of length n, let X_i denote its prefix of length i, $1 \le i \le n$. Say that X can be 1-embedded into Y if there exists a decimation sequence of integers $D = \{d_i\}_{i=1}^n$ such that $x_i = y_{d_i}$ for $1 \le i \le n$, with $d_1 = 1$ and $d_i - d_{i-1} \in \{1, 2\}$ for $2 \le i \le n$. Equivalently, X can be 1-embedded into Y if Y can be obtained from X by inserting at most one bit between any two successive bits of X and an arbitrary number of bits at the end. If in addition $m - d_n \in \{0, 1\}$, then X is said to strictly 1-embed into Y, and in particular if $d_n = m$, then X is said to 1-embed onto Y.

* Received by the editors January 13, 1992; accepted for publication (in revised form) August 11, 1995. A preliminary version was presented at 36.ETAN, Yugoslavia, 1992.

† Information Security Research Centre, Queensland University of Technology, GPO Box 2434, Brisbane, Queensland 4001, Australia, and School of Electrical Engineering, University of Belgrade, Yugoslavia (golic@fit.qut.edu.au).

Our objective here is to determine the probability that a given binary string X of length n can be 1-embedded into a uniformly distributed random binary string Y of length $2n$. As is shown in [5], this probability proves to be important for the cryptanalytic problem of reconstructing the initial state of a shift register given a segment of its decimated output sequence. Let $A_{n,k}(X)$, $0 \leq k \leq n$, denote the number of binary strings of length $n+k$ into which a binary string X of length n can be strictly 1-embedded. Also, let $A_{n,k}$ denote the maximum of $A_{n,k}(X)$ over all X of length n and let

$$A_n = \sum_{k=0}^{n} 2^{n-k} A_{n,k}. \tag{1}$$

Clearly, A_n represents an upper bound on the number of binary strings of length $2n$ into which any binary string of length n can be 1-embedded. This is just an upper bound because it is in general possible that a given binary string can be strictly 1-embedded into two binary strings of different lengths one of which is the prefix of the other. Therefore, the probability that an arbitrary binary string of length n can be 1-embedded into a uniformly distributed random binary string of length $2n$ is upper-bounded by

$$P_n = 2^{-2n} A_n. \tag{2}$$

In the next section, we will derive and solve a linear recurrence for A_n and thus determine the upper bound P_n. To this end, we also introduce $A_{n,k,i}(X)$, $i = 0, 1$, as the number of binary strings Y of length $n+k$ such that a binary string X of length n can be 1-embedded onto the prefix Y_{n+k-i} and not onto $Y_{n+k} = Y$ for $i = 1$. Similarly, let $A_{n,k,i}$ denote the maximum of $A_{n,k,i}(X)$ over all X of length n. Clearly, for every X,

$$A_{n,k}(X) = A_{n,k,0}(X) + A_{n,k,1}(X). \tag{3}$$

3. Embedding probability. Our basic result is a recursive property of $A_{n,k,i}(X)$ established by the following theorem.

THEOREM 3.1. *For an arbitrary binary string X of length N and every $2 \leq n \leq N$ and $0 \leq k \leq n$ we have*

$$A_{n,k,0}(X_n) = A_{n-1,k,0}(X_{n-1}) + A_{n-1,k,1}(X_{n-1}), \tag{4}$$

$$A_{n,k,1}(X_n) \leq A_{n,k-1,0}(X_n) + A_{n-1,k-1,1}(X_{n-1}), \tag{5}$$

with equality in (5) if $x_{n-1} \neq x_n$ and with the initial values, independent of X_1, $A_{1,0,0}(X_1) = 1$, $A_{1,0,1}(X_1) = 0$, $A_{1,1,0}(X_1) = 0$, and $A_{1,1,1}(X_1) = 2$. It is assumed that $A_{n,k,i}(X_n) = 0$ if $k < 0$ or $k > n$.

Proof. The initial conditions follow directly. By the definition of $A_{n,k,0}(X_n)$, (4) is true because X_n can be 1-embedded onto Y_{n+k} if and only if $y_{n+k} = x_n$ and X_{n-1} can be strictly 1-embedded into Y_{n+k-1}.

On the other hand, a closer examination of $A_{n,k,1}(X_n)$ reveals the following. If $y_{n+k} \neq x_n$, then X_n can be 1-embedded onto Y_{n+k-1} and not onto Y_{n+k} if and only if X_n can be 1-embedded onto Y_{n+k-1}, which accounts for the first term on the right-hand side of (5). If $y_{n+k} = x_n$, then X_n can be 1-embedded onto Y_{n+k-1} and not

onto Y_{n+k} *only if* $y_{n+k-1} = x_n$ and X_{n-1} can be 1-embedded onto Y_{n+k-3} and not onto Y_{n+k-2}. Namely, if X_{n-1} could be 1-embedded onto Y_{n+k-2}, then in view of $y_{n+k} = x_n$ it follows that X_n could be 1-embedded onto Y_{n+k}. This accounts for the second term on the right-hand side of (5), whereas the inequality is a consequence of the fact that the condition is necessary but in general not sufficient. Precisely, if $x_{n-1} = x_n$ and X_{n-2} can be strictly 1-embedded into Y_{n+k-2}, then X_n can be 1-embedded onto Y_{n+k}, because $y_{n+k} = x_n$ and $y_{n+k-1} = x_n = x_{n-1}$. However, if $x_{n-1} \neq x_n$, this is not possible and the condition becomes sufficient, meaning that if $y_{n+k} = x_n$, then X_n can be 1-embedded onto Y_{n+k-1} and not onto Y_{n+k} *if and only if* $y_{n+k-1} = x_n$ and X_{n-1} can be 1-embedded onto Y_{n+k-3} and not onto Y_{n+k-2}. Therefore, if $x_{n-1} \neq x_n$, then (5) holds with equality. □

COROLLARY 3.1. *If X is a binary alternating string of length N, then* (5) *holds with equality for every $2 \leq n \leq N$ and $0 \leq k \leq n$.* □

Consequently, for alternating strings the system of recursive inequalities for $A_{n,k,i}(X_n)$ given in Theorem 3.1 becomes a system of two linear recurrences. Moreover, since the coefficients in (3)–(5) are positive and the initial values $A_{1,k,i}(X_1)$ are independent of X_1, we directly obtain the following corollary.

COROLLARY 3.2. *For any $n \geq 1$, $0 \leq k \leq n$, and $i = 0, 1$, the maximum values $A_{n,k,i}$ and $A_{n,k}$ of $A_{n,k,i}(X_n)$ and $A_{n,k}(X_n)$ over all X_n of length n are both achieved if X_n is alternating and are determined by the linear recurrences*

$$A_{n,k,0} = A_{n-1,k,0} + A_{n-1,k,1}, \tag{6}$$

$$A_{n,k,1} = A_{n,k-1,0} + A_{n-1,k-1,1}, \tag{7}$$

for $n \geq 2$ and $0 \leq k \leq n$, with the initial values $A_{1,0,0} = 1$, $A_{1,0,1} = 0$, $A_{1,1,0} = 0$, and $A_{1,1,1} = 2$, assuming that $A_{n,k,i} = 0$ if $k < 0$ or $k > n$, along with the linear equation

$$A_{n,k} = A_{n,k,0} + A_{n,k,1}. \tag{8}$$ □

After certain manipulations with (6)–(8), it is not difficult to obtain a linear recurrence for $A_{n,k}$.

COROLLARY 3.3. *For every $n \geq 3$ and $0 \leq k \leq n$ we have*

$$A_{n,k} = A_{n-1,k} + 2A_{n-1,k-1} - A_{n-2,k-1}, \tag{9}$$

with the initial values $A_{1,0} = 1$, $A_{1,1} = 2$, $A_{2,0} = 1$, $A_{2,1} = 3$, and $A_{2,2} = 4$, assuming that $A_{n,k} = 0$ if $k < 0$ or $k > n$. □

Corollary 3.3 together with (1) results in a linear recurrence for A_n which can be solved easily.

COROLLARY 3.4. *For every $n \geq 3$ we have*

$$A_n = 4A_{n-1} - 2A_{n-2}, \tag{10}$$

with the initial values $A_1 = 4$ and $A_2 = 14$. The solution to (10) *is given by*

$$A_n = \frac{\sqrt{2}+1}{2}(2+\sqrt{2})^n - \frac{\sqrt{2}-1}{2}(2-\sqrt{2})^n, \quad n \geq 1. \tag{11}$$ □

Corollary 3.4 and (2) determine the desired upper bound P_n on the constrained embedding probability. The bound is exponentially small for large n.

COROLLARY 3.5. *We have that*

$$P_n = \frac{\sqrt{2}+1}{2}\left(\frac{2+\sqrt{2}}{4}\right)^n - \frac{\sqrt{2}-1}{2}\left(\frac{2-\sqrt{2}}{4}\right)^n, \quad n \geq 1. \qquad \square \tag{12}$$

We proceed now by discussing the upper bound from [5]. There it is shown that for any $1 \leq m \leq n$,

$$P_n \leq P_m^{\lfloor n/m \rfloor} \leq \left(\frac{A_m^{1/m}}{4}\right)^{n-m+1}, \tag{13}$$

where $\lfloor x \rfloor$ denotes the integer part of x. Note that (13) is a simple consequence of (1) and the fact that any strict 1-embedding of a concatenation of binary strings can be represented as a concatenation of strict 1-embeddings of individual strings and vice versa. Interestingly enough, it follows that any $P_m < 1$ results in an exponentially small upper bound on the constrained embedding probability. As is shown in [5] by direct counting, it turns out that P_m is smaller than one for every $2 \leq m \leq 7$. The sharpest bound is obtained for $m = 7$. Clearly our bound (12), which is based on the analytical expression (11) for A_n, is sharper.

Finally, it is not difficult to see that the upper bound A_n can be further improved if instead of (1) we use

$$A'_n = \sum_{k=0}^{n} 2^{n-k} A_{n,k,1}, \tag{14}$$

which is equal to $A_n - 2A_{n-1}$, for $n \geq 2$, and $A'_1 = 2$. In view of (11), the corresponding upper bound on the constrained embedding probability $P'_n = 2^{-2n} A'_n$ is then given by the following corollary.

COROLLARY 3.6. *We have that*

$$P'_n = \frac{1}{2}\left(\frac{2+\sqrt{2}}{4}\right)^n + \frac{1}{2}\left(\frac{2-\sqrt{2}}{4}\right)^n, \quad n \geq 1. \qquad \square \tag{15}$$

4. Conclusion. In this paper, a counting combinatorial problem raised in [5] is analytically solved and an upper bound on the probability that a given binary string of length n can be embedded into a uniformly distributed random binary string of length $2n$ by inserting at most one bit between any two successive bits and an arbitrary number of bits at the end is thus determined. The solution is obtained by deriving suitable recursive characteristics of certain numbers of binary strings of a given length that satisfy appropriate embedding properties. The probability is important for the cryptanalytic problem [5], [3] of the initial state reconstruction of a binary clock-controlled shift register that is clocked at least once and at most twice per each output symbol, given a segment of its output sequence. The obtained bound is sharper than the one from [5] where the underlying combinatorial problem is treated by direct counting for small string lengths. Both the bounds are exponentially small for large string lengths which shows that the required length of the output sequence for the successful reconstruction is linear in the shift register length. The developed theoretical approach may also be useful for solving a more general constrained embedding problem with an arbitrary maximum number of consecutive insertions or a

modified embedding problem with permitted substitutions, both relevant for the general cryptanalytic problem pointed out in [3]. Apart from that, the result might also be interesting for other problems of sequence comparison [4] as well, especially for synchronization error-correcting codes; see [1], [2], for example.

REFERENCES

[1] L. Calabi and W. E. Hartnett, *Some general results of coding theory with applications to the study of codes for the correction of synchronization errors*, Inform. and Control, 15 (1969), pp. 235–249.

[2] A. S. Dolgopolov, *Capacity bounds for a channel with synchronization errors*, Prob. Peredachi Inform., 26 (1990), pp. 27–37. (In Russian.)

[3] J. Dj. Golić and M. J. Mihaljević, *A generalized correlation attack on a class of stream ciphers based on the Levenshtein distance*, J. Cryptology, 3 (1991), pp. 201–212.

[4] D. Sankoff and J. B. Kruskal, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison–Wesley, Reading, MA, 1983.

[5] M. V. Živković, *An algorithm for the initial state reconstruction of the clock-controlled shift register*, IEEE Trans. Inform. Theory, 37 (1991), pp. 1488–1490.

SIAM J. DISCRETE MATH.
Vol. 9, No. 3, pp. 365–376, August 1996

003

ON THE COMPLEXITY OF A CUTTING PLANE ALGORITHM FOR SOLVING COMBINATORIAL LINEAR PROGRAMS*

E. ANDREW BOYD†

Abstract. A cutting plane algorithm is presented for solving combinatorial linear programs—integer programs with 0/1 vertices represented by a separation oracle. The algorithm is a standard cutting plane method but uses a prescribed dual criterion for choosing a cut at each iteration. As a result, it is possible to demonstrate that for problems containing a ball whose size is polynomially bounded from below, there exists an algorithm polynomial in the running time of the separation oracle and pseudopolynomial in the size of the objective function. In particular, the cardinality versions of many combinatorial optimization problems are shown to be solvable in polynomial time using this generic algorithm.

Key words. cutting planes, integer programming

AMS subject classifications. 90C10, 90C11

1. Introduction.

Cutting plane algorithms for solving integer programs have received considerable attention in recent years. Much of the work has been performed on specific combinatorial optimization problems such as the traveling salesman problem (see, for example, [19] and [20]), but a great deal of work has also been devoted to general integer programs (see, for example, [7], [13], and [21]). Padberg coined the term "branch-and-cut" for branch-and-bound algorithms making extensive use of cutting planes at nodes of the search tree, and branch-and-cut algorithms form the basis for almost all of the recent work in exact algorithms for integer programs.

Yet, while cutting plane algorithms have been used with great frequency and success in practice, very little is known about their convergence properties apart from the fact that in many instances they are finite. In the case of Gomory cutting planes [10], [11], and Fenchel cutting planes [5], finiteness was at one time an issue in and of itself. Lift and project cutting plane procedures using ideas such as those found in [2], [17], and [23] are finite but depend upon an exponential function of the problem dimension. Standard polyhedral cutting plane algorithms commonly employ separation oracles for finding violated inequalities that generate facets of the underlying polyhedron, and in these instances the running time of the cutting plane algorithm is bounded by the number of facets. Unfortunately, in most instances where cutting plane algorithms are actually employed, the number of facets is exponential in the natural problem size. Even for problems that are known to have polynomial algorithms that are not based on cutting planes, such as the matching problem, the theoretical behavior of cutting plane algorithms remains unexplored. The importance of convergence properties is far from purely theoretical, since it is well recognized that cutting plane algorithms almost uniformly demonstrate significant "tailing off"—substantial progress in early iterations with very little progress in latter iterations. While general observations can be made explaining this phenomenon, a better theoretical understanding of the convergence properties of cutting plane algorithms may lead to variants which at least partially overcome this difficulty.

* Received by the editors November 23, 1993; accepted for publication (in revised form) August 14, 1995. This work was sponsored in part by the National Science Foundation and the Office of Naval Research under NSF grant number DDM-9396105.

† Department of Industrial Engineering, Texas A&M University, College Station, TX 77843–3131 (boyd@marvin.tamu.edu).

An alternative to more standard cutting plane methods is the ellipsoid algorithm, the convergence properties of which are well studied and theoretically quite satisfactory. (See [4].) Like cutting plane algorithms, the ellipsoid algorithm does not require an explicit representation of the linear program being solved, instead requiring only the existence of a separation oracle. The polynomiality of the ellipsoid algorithm in the presence of a polynomial separation oracle was recognized early in the development of the ellipsoid algorithm, and is extensively elaborated upon in [12]. In fact, the unique theoretical properties of the ellipsoid algorithm are responsible for its continued significance in combinatorial optimization. Nonetheless, the algorithm performs so poorly in practice that it has been all but abandoned, and in problems where the underlying linear program is represented by a separation oracle more standard cutting plane algorithms continue to be used.

In light of the favored status of standard cutting plane algorithms in practice, their convergence properties remain compelling. In this paper we develop a cutting plane algorithm whose complexity is unrelated to the number of facets defining the underlying optimization polyhedron. The algorithm is a standard cutting plane method but uses a prescribed dual criterion for choosing a cut at each iteration. As a result, it is possible to demonstrate that for problems containing a ball whose size is polynomially bounded from below, there exists an algorithm polynomial in the running time of the separation oracle and pseudopolynomial in the size of the objective function. In particular, the cardinality versions of many combinatorial optimization problems are shown to be solvable in polynomial time using this generic cutting plane algorithm.

The optimization problem to be considered is

$$\text{(P)} \qquad \begin{array}{ll} \max & cx \\ \text{s.t.} & x \in \mathcal{P} \end{array}$$

where $\mathcal{P}$ is a polyhedron. Rather than assuming $\mathcal{P}$ is represented by an explicit set of linear inequalities, we assume instead that it is represented by a separation algorithm, SEPARATE, defined as follows.

SEPARATE. Given a point $\bar{x} \in \mathbf{R}^n$, find an inequality $\pi x \leq \delta$ satisfying $\pi \bar{x} > \delta$ and $\pi x \leq \delta$ for all $x \in \mathcal{P}$, or prove that no such inequality exists.

Polyhedra represented in this form commonly arise in the context of integer and mixed-integer programming, where $\mathcal{P}$ corresponds to the convex hull of feasible integer solutions for the underlying problem and an explicit inequality description is not typically known. We make the following assumptions.

Assumptions. 1. $\mathcal{P} \subseteq \mathbf{R}^n$ is full dimensional and contains the origin in its interior.

2. There exists a ball of radius $r \leq 1$ centered at the origin and contained in $\mathcal{P}$ and a ball of radius $R \geq 1$ centered at the origin which contains $\mathcal{P}$.

3. c is integral and the absolute value of each element of c is bounded by C.

4. The extreme points $\bar{x}$ of $\mathcal{P}$ can be expressed as $\bar{x}/M$, where $\bar{x}$ is integral and the absolute value of each element of $\bar{x}$ is bounded by N.

Translating $\mathcal{P}$ so that the origin is contained in its interior is primarily for expository purposes, since it allows 1-polar results to be applied without constant reference to a virtual origin. The 1-*polar* Π_1 of $\mathcal{P}$ is defined by

$$\Pi_1 = \{\pi \in \mathbf{R}^n : \pi x \leq 1 \text{ is valid for } \mathcal{P}\}.$$

With the origin contained in the interior of $\mathcal{P}$, every valid inequality for $\mathcal{P}$ is represented in Π_1 (up to scalar multiplication). The following proposition is used extensively and without explicit reference in the results to be presented [18].

PROPOSITION 1.1. *Suppose $\mathcal{P}$ is bounded and contains the origin in its interior. Then Π_1 is a polyhedron, and $x\pi \leq 1$ is a facet defining inequality of Π_1 if and only if x is an extreme point of $\mathcal{P}$. Further, $\mathcal{P}$ is the 1-polar of Π_1.*

The following results are easily verified and will prove useful in the exposition to follow. Throughout the paper, $\|\cdot\|$ will be used to denote the ℓ_2 norm of a vector, and log will denote the base 2 logarithm unless otherwise indicated.

PROPOSITION 1.2. *Given a problem* (P) *satisfying assumptions* 1 *through* 4, *the following conditions are true.*

1. *Every extreme point solution of $\mathcal{P}$ has objective function value p/M in* (P), *where p is an integer in the interval $[-NCn, NCn]$. In particular, the optimal value of* (P) *is p/M, where $p \in [0, NCn]$.*
2. *For any $\pi \in \Pi_1$, $\|\pi\| \leq 1/r$.*

2. Algorithm. Given a problem (P) as described in the previous section, a generic implementation of standard cutting plane methods proceeds as follows. The polyhedron $\mathcal{P}$ is first approximated by at least n linearly independent constraints $\pi_i x \leq 1$ with the property that maximizing cx subject to these constraints yields a finite solution $\hat{x}$. A separation oracle is then invoked to determine if $\hat{x} \in \mathcal{P}$, and if so the algorithm terminates with $\hat{x}$ optimal for (P). Otherwise, the separating hyperplane $\tilde{\pi}x \leq 1$ is included in the set of constraints approximating $\mathcal{P}$ and the process continues. An idealized version of this algorithm maintains exactly n linearly independent constraints at each iteration, replacing one incumbent constraint with the cutting plane generated at each iteration.

The proposed algorithm varies from the standard framework in two regards. First, it maintains $n+1$ affinely independent constraints rather than n constraints, and this is essential in proving the convergence properties of the algorithm. Second, the algorithm is based on solving a collection of validity problems rather than a single optimization problem, where each validity problem is solved using cutting planes. More specifically, the proposed algorithm proceeds in a sequence of major iterations indexed by s where the optimal objective function value is assumed to be v^s. The algorithm then verifies or refutes this condition by determining whether or not the constraint $c^s x \leq 1$, $c^s = c/v^s$, is valid for $\mathcal{P}$. The algorithm begins with the procedure MAIN shown in Fig. 1, which is primarily a binary search procedure. It calls VALIDITY to determine if the constraint $c^s x \leq 1$ is valid for $\mathcal{P}$ or not, and OPTIMAL to construct an optimal solution from a near optimal solution at the termination of the algorithm.

Input: A problem (P) satisfying assumptions 1 through 4.
Output: An optimal extreme point solution x^* to (P) and its optimal value v^*.

0. *Initialize.* Let $v^0_{LB} = 0$, $v^0_{UB} = NCn/M$, and $\tilde{x} = 0$. Let $s = 0$.

1. Let $s = s + 1$, $v^s = \lfloor M(v^s_{LB} + v^s_{UB})/2 \rfloor / M$, and $c^s = c/v^s$. If $v^s = v^s_{LB}$ call OPTIMAL($\tilde{x}$), let x^* be the value returned by this algorithm, let $v^* = v^s_{UB}$, and stop. Otherwise, continue with step 2.

2. If $\|c^s\| > 1/r$, $c^s x \leq 1$ is not valid for $\mathcal{P}$; go to step 3. Otherwise, call VALIDITY(c^s).

3. If VALIDITY determines that $c^s x \leq 1$ is valid for $\mathcal{P}$, let $v^s_{UB} = v^s$. Otherwise, let $v^s_{LB} = v^s$ and let $\tilde{x}$ be the value of x returned by VALIDITY. Continue with step 1.

FIG. 1. *Procedure MAIN.*

Key to the operation of the overall algorithm is procedure VALIDITY. (See Fig. 2.) At iteration t, VALIDITY maintains a set Q^t of $n+1$ affinely independent valid inequalities $\pi_i x \leq 1$ for $\mathcal{P}$ and makes use of procedures CLOSE and SEPSTEP. In each iteration, the procedure either demonstrates the validity or invalidity of the constraint $c^s x \leq 1$ or it determines a new set of valid inequalities Q^t differing from the previous set by a single constraint. Validity is demonstrated by showing that c^s is contained in or almost contained in $conv(Q^t)$, and determining this condition is the purpose of CLOSE. Invalidity is determined by finding a point $\tilde{x} \in \mathcal{P}$ such that $c^s \tilde{x} > 1$, which is the function of SEPSTEP. If neither validity nor invalidity of the constraint $c^s x \leq 1$ is determined during a particular iteration t, Q^t is modified by excluding a constraint determined by CLOSE and including a constraint determined by SEPSTEP. Invalidity can also be determined by performing a fixed number of iterations of VALIDITY, and this is an extremely important consideration in bounding the work performed by the algorithm.

Input: A problem (P) satisfying assumptions 1 through 4 and a constraint $c^s x \leq 1$.

Output: A point $\tilde{x} \in \mathcal{P}$ satisfying $c^s \tilde{x} > 1$ or a proof that no such $\tilde{x}$ exists.

0. *Initialize.* Let $t = 0$. Determine a set of $n+1$ affinely independent valid inequalities $\pi_i^t x \leq 1$ for $\mathcal{P}$ and let $Q^t = \{\pi_1^t, \ldots, \pi_{n+1}^t\}$.

1. Call CLOSE(c^s, Q^t). If the returned value $\pi_0^t \in conv(Q^t)$ satisfies $\|c^s - \pi_0^t\| < (RNCn)^{-1}$, return; $c^s x \leq 1$ is valid for $\mathcal{P}$. Otherwise, let π_k^t be the element of Q^t returned by CLOSE.

2. Call SEPSTEP(c^s, π_0^t). If a value $\tilde{x}^t \in \mathcal{P}$ satisfying $c^s \tilde{x}^t > 1$ is returned, return; $c^s x \leq 1$ is invalid for $\mathcal{P}$. Otherwise, let $Q^{t+1} = Q^t \cup \{\tilde{\pi}^t\} - \{\pi_k^t\}$, where $\tilde{\pi}^t x \leq 1$ is the separating hyperplane returned by SEPSTEP.

3. Let $t = t + 1$. If

$$t = 2\left(\frac{4RNCn}{r}\right)^2 \left|\log_e\left(\frac{r}{2RNCn}\right)\right|$$

return; $c^s x \leq 1$ is valid for $\mathcal{P}$. Otherwise, go to step 1.

FIG. 2. *Procedure VALIDITY.*

CLOSE (see Fig. 3) takes as input the collection Q^t of $n+1$ valid inequalities for $\mathcal{P}$ maintained by VALIDITY and seeks to determine if c^s can be expressed as a convex combination of these vectors. It accomplishes this task by seeking the unique point $\pi_0^t \in conv(Q^t)$ closest to c^s. If π_0^t is sufficiently close to c^s, indicating that c^s is either contained in or nearly contained in $conv(Q^t)$, CLOSE returns only π_0^t as an indicator of this fact. Otherwise, CLOSE returns both π_0^t and a $\pi_k^t \in Q^t$ such that $\alpha_k = 0$ in the expression $\pi_0^t = \sum_{Q^t} \alpha_i \pi_i^t$; that is, π_0^t can be expressed as a convex combination of the vectors in $Q^t - \{\pi_k^t\}$.

SEPSTEP (see Fig. 4) takes as input the direction $d = (c^s - \pi_0^t)/\|c^s - \pi_0^t\|$ and seeks to determine how far it is possible to move in this direction from the origin while remaining in the polyhedron $\mathcal{P}$. More to the point, it conceptually seeks $\tilde{x}_0^t = \theta d$, $\theta > 0$, on the boundary of $\mathcal{P}$ and a constraint $\tilde{\pi}_0^t x \leq 1$ defining the face of $\mathcal{P}$ in which $\tilde{x}_0^t$ resides. In actuality, it uses binary search to determine approximations $\tilde{x}^t$ of $\tilde{x}_0^t$ and $\tilde{\pi}^t$ of $\tilde{\pi}_0^t$ with $\tilde{x}^t \in \mathcal{P}$ and $\tilde{\pi}^t x \leq 1$ valid for $\mathcal{P}$. If $c^s \tilde{x}^t > 1$ the procedure returns $\tilde{x}^t$ as proof that $c^s x \leq 1$ is not valid for $\mathcal{P}$; otherwise, it returns $\tilde{\pi}^t x \leq 1$ as a new valid inequality for $\mathcal{P}$, and this inequality is used to replace the inequality $\pi_k^t x \leq 1$

Input: A set of $n+1$ affinely independent vectors $\pi_1, \ldots, \pi_{n+1}$ and a distinguished point c^s.
Output: The unique point π_0 satisfying $\|c^s - \pi_0\| \leq \|c^s - \pi\|$ for all $\pi \in conv(\pi_1, \ldots, \pi_{n+1})$, and if $c^s \notin conv(\pi_1, \ldots, \pi_{n+1})$ a value π_k such that $\pi_0 \in conv(\pi_1, \ldots, \pi_{k-1}, \pi_{k+1}, \ldots, \pi_{n+1})$.
Procedure: Apply the algorithm of Kojima, Mizuno, and Yoshise [14] to solve the linear complementary problem

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} - \begin{bmatrix} & & & & \Pi & -I & I \\ & & & & -\Pi & I & -I \\ & & & & -e^T & & \\ & & & & e^T & & \\ -\Pi^T & \Pi^T & e & -e & & & \\ I & -I & & & & I & -I \\ -I & I & & & & -I & I \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ -c^s \\ c^s \end{bmatrix}$$

$$x_i v_i = 0 \qquad y_i u_i = 0$$

$$x_i, y_i, u_i, v_i \geq 0$$

where Π is a matrix with columns $\pi_1, \ldots, \pi_{n+1}$ and e^T is a row vector of 1's. Return $\pi_0 = x_2 - x_3$ and any π_k such that element k of x_1 is equal to 0, if such a π_k exists.

FIG. 3. *Procedure CLOSE.*

Input: A problem (P) satisfying assumptions 1 through 4 and vectors c^s and π_0.
Output: A point $\tilde{x} \in \mathcal{P}$ satisfying $c^s\tilde{x} > 1$ or a valid inequality $\tilde{\pi}x \leq 1$ for $\mathcal{P}$.

0. *Initialize.* Let $h = 0$, $d = (c^s - \pi_0)/\|c^s - \pi_0\|$, $u^h_{LB} = r$, and $u^h_{UB} = R + r$. Call SEPARATE($u^h_{UB}d$) and let $\tilde{\pi}x \leq 1$ be the separating hyperplane that is returned.
1. Let $u^h = \frac{1}{2}(u^h_{LB} + u^h_{UB})$.
2. Call SEPARATE($u^h d$). If $u^h d \in \mathcal{P}$ let $u^h_{LB} = u^h$, otherwise let $u^h_{UB} = u^h$ and let $\tilde{\pi}x \leq 1$ be the separating hyperplane returned by SEPARATE.
3. Let $h = h + 1$. If $c^s(u^h_{LB}d) > 1$ return $\tilde{x} = u^h_{LB}d$. Otherwise, if $h \geq \log 2R^2r^{-2}NCn$ return the valid inequality for $\mathcal{P}$, $\tilde{\pi}x \leq 1$. Otherwise, go to step 1.

FIG. 4. *Procedure SEPSTEP.*

in Q^t determined by CLOSE. It is often the case in practice that when a separation oracle exists a direct "step-length" oracle exists as well so that binary search need not be used.

The remaining procedure, OPTIMAL (see Fig. 5), is a purification procedure that generates an optimal solution to (P) from a near optimal solution. It is discussed in greater detail in §4.

It is important to realize that while the algorithm is expressed using concepts

Input: A problem (P) satisfying assumptions 1 through 4, a vector c, and a point $\tilde{x} \in \mathcal{P}$ satisfying $c\tilde{x} > v^* - 1/M$, where v^* is the optimal value of (P).
Output: An optimal solution for (P).

0. *Initialize.* Let $q = 0$, $\tilde{x}^q = \tilde{x}$, and let B be the null matrix. Let d^q be any non-descent direction with respect to c.

1. Use SEPARATE in a binary search algorithm to determine the largest value $\tilde{\theta}$ of θ such that $\tilde{x}^q + \theta d^q \in \mathcal{P}$ and a valid inequality $\tilde{\pi}^q x \leq 1$ for $\mathcal{P}$ such that $\tilde{\pi}^q(\tilde{x}^q + \tilde{\theta} d^q) = 1$. Let $\tilde{x}^{q+1} = \tilde{x}^q + \tilde{\theta} d^q$ and append $\tilde{\pi}^q$ to B as a new row. If $q + 1 = n$, return the value $\tilde{x}^{q+1}$.

2. Let d^{q+1} be any non-descent direction with respect to c contained in the nullspace of B. Let $q = q + 1$ and go to step 1.

FIG. 5. *Procedure OPTIMAL.*

related to the primal space in which $\mathcal{P}$ resides, motivation for the algorithm comes from the dual space in which Π_1 resides. In this space, the direction $d = (c^s - \pi_0^t)/\|c^s - \pi_0^t\|$ passed to SEPSTEP by VALIDITY can be seen as defining an objective function that when maximized on Π_1 yields an optimal solution $\tilde{\pi}^t$ with the properties required to achieve the convergence criteria embodied in the statement of Theorem 3.2. The dual interpretation of SEPSTEP is that it yields a solution $\tilde{\pi}^t$ to this optimization problem of high accuracy using binary search and SEPARATE as a validity oracle for Π_1.

Apart from the binary search aspect of MAIN dictated by solving a sequence of validity problems, the major difference between the proposed algorithm and standard cutting plane algorithms is the way in which the cutting plane is chosen. Rather than being driven by seeking to separate the incumbent linear programming solution from $\mathcal{P}$, the algorithm is instead driven by seeking to express c^s as a convex combination of valid inequalities for $\mathcal{P}$, thus demonstrating that $c^s x \leq 1$ is itself valid for $\mathcal{P}$. It is this dual approach which leads to a complexity independent of the number of facets of $\mathcal{P}$.

3. Proof of correctness. A number of issues remain to be addressed in order to demonstrate the correctness of the algorithm. We begin by making some observations about the algorithm that do not require explicit proof but are fundamental to understanding the algorithm.

Proposition 1.2 guarantees that v_{LB}^0 and v_{UB}^0 as defined in MAIN represent lower and upper bounds on the optimal value of (P), and that every extreme point of (P) has an objective function value that is a multiple of $1/M$. The operation of MAIN guarantees that v_{LB}^0 and v_{UB}^0 remain lower and upper bounds on the optimal value of (P). Thus, when MAIN terminates, v^* is the optimal value for (P), and while $\tilde{x}$ is not optimal for (P) it is feasible and $c\tilde{x} > v^* - 1/M$. The point $\tilde{x}$ is therefore near optimal in the sense of having objective function value better than any nonoptimal extreme points of $\mathcal{P}$. These facts are important to the operation of OPTIMAL. The claim made in step 2 of MAIN follows directly from Proposition 1.2.1.

CLOSE is a direct application of the algorithm of Kojima, Mizuno, and Yoshise [14] to the linear complementarity problem derived from the Karush-Kuhn-Tucker conditions associated with finding the closest point π_0 to c^s contained in $conv(\pi_1, \ldots, \pi_{n+1})$. The solution of this problem is such that $\pi_0 = x_2 - x_3$ and x_1 is the vector of weights expressing π_0 as a convex combination of the points $\pi_1, \ldots, \pi_{n+1}$; that is, $\pi_0 = \Pi x_1$.

Note that if $c^s \notin conv(\pi_1, \ldots, \pi_{n+1})$, at least one of the entries in x_1 must be 0, as implied in the statement of the algorithm.

The definitions of R and r ensure that $u^0_{LB}d \in \mathcal{P}$ and $u^0_{UB}d \notin \mathcal{P}$ in SEPSTEP. The values $u^h_{LB}d$ and $u^h_{UB}d$ remain contained in and not contained in $\mathcal{P}$, respectively, by virtue of the binary search nature of the procedure.

Two issues require explicit verification, both found in VALIDITY. First, it must be demonstrated that when VALIDITY terminates in step 1, it is correct to conclude that $c^s x \leq 1$ is a valid inequality for $\mathcal{P}$. Second, it must be demonstrated that when VALIDITY terminates in step 3 after the stated number of iterations it is correct to reach this same conclusion. The following theorem addresses the first question.

THEOREM 3.1. *Let $c^s = \frac{M}{p}c$, where $p \in [1, NCn]$ is an integer, and suppose $c^s \notin \Pi_1$. Then for any $\pi \in \Pi_1$, $\|c^s - \pi\| \geq (RNCn)^{-1}$.*

Proof. Let $q > p$ be the smallest integer such that $\frac{M}{q}cx \leq 1$ is valid for $\mathcal{P}$. By Proposition 1.2.1, $q \leq NCn$ and there exists an $\bar{x} \in \mathcal{P}$ such that $\frac{M}{q}c\bar{x} = 1$. Since $\bar{x}\pi \leq 1$ is valid for Π_1, it follows that for any $\pi \in \Pi_1$,

$$\|c^s - \pi\| \geq \frac{1}{\|\bar{x}\|}|\bar{x}c^s - \bar{x}\pi| \geq \frac{1}{\|\bar{x}\|}\left(\bar{x}\frac{M}{p}c - \bar{x}\frac{M}{q}c\right) = \frac{1}{\|\bar{x}\|}M\bar{x}c\frac{q-p}{pq}.$$

With $q > p$ we have $(q-p)/pq \geq 1/pq$, and with $\bar{x}c = q/M$ and $\|\bar{x}\| \leq R$ the last expression is greater than or equal to $1/pR$. Together with the fact that $p \leq NCn$, the desired result follows. □

Since CLOSE returns a value $\pi_0 \in \Pi_1$, Theorem 3.1 verifies the termination condition in step 1 of procedure VALIDITY.

The termination condition in step 3 of procedure VALIDITY also depends on Theorem 3.1. It can be shown that $\|c^s - \pi^t_0\|$ strictly decreases at each iteration t at a rate such that after the number of iterations stated in step 3 of procedure VALIDITY, $\|c^s - \pi^t_0\| < (RNCn)^{-1}$. Thus, by Theorem 3.1, if $c^s x \leq 1$ is not valid for $\mathcal{P}$, procedure VALIDITY must terminate within this number of iterations. The following theorem demonstrates the rate at which $\|c^s - \pi^t_0\|$ decreases.

THEOREM 3.2. *If $c^s \notin \Pi_1$, then at iteration t of VALIDITY,*

$$\|c^s - \pi^{t+1}_0\| \leq \left[1 - \left(\frac{r}{4RNCn}\right)^2\right]^{1/2} \|c^s - \pi^t_0\|. \tag{1}$$

Proof. In order to demonstrate (1) it is shown that some point $\bar{\pi}^t$ on the line segment connecting π^t_0 and $\tilde{\pi}^t$ defined in VALIDITY satisfies

$$\|c^s - \bar{\pi}^t\| \leq \left[1 - \left(\frac{r}{4RNCn}\right)^2\right]^{1/2} \|c^s - \pi^t_0\|. \tag{2}$$

Since π^t_0 and $\tilde{\pi}^t \in conv(Q^{t+1})$ by the operation of VALIDITY and since $\|c^s - \pi^{t+1}_0\| \leq \|c^s - \bar{\pi}^t\|$ by the definition of π^{t+1}_0, (1) follows.

It is shown in Lemma 3.4 that $\tilde{\pi}^t$ returned by SEPSTEP satisfies

$$\frac{(c^s - \pi^t_0)}{\|c^s - \pi^t_0\|}c^s - \frac{(c^s - \pi^t_0)}{\|c^s - \pi^t_0\|}\tilde{\pi}^t \leq \frac{1}{2RNCn}, \tag{3}$$

and since $c^s \notin \Pi_1$ it follows by Theorem 3.1 that

$$\|c^s - \pi^t_0\| = \frac{(c^s - \pi^t_0)}{\|c^s - \pi^t_0\|}c^s - \frac{(c^s - \pi^t_0)}{\|c^s - \pi^t_0\|}\pi^t_0 \geq \frac{1}{RNCn}. \tag{4}$$

Thus,

$$\frac{(c^s-\pi_0^t)}{\|c^s-\pi_0^t\|}\tilde{\pi}^t-\frac{(c^s-\pi_0^t)}{\|c^s-\pi_0^t\|}\pi_0^t\geq\frac{1}{2RNCn}. \tag{5}$$

Let $\bar{\pi}^t$ be the projection of c^s onto the line defined by π_0^t and $\tilde{\pi}^t$. If $\bar{\pi}^t=c^s$ then π_0^t, $\tilde{\pi}^t$, and c^s are colinear, in which case (3) and (4) imply $\|c^s-\tilde{\pi}^t\|\leq\|c^s-\pi_0^t\|/2$. With $\tilde{\pi}^t\in Q^{t+1}$ by the operation of VALIDITY, this would complete the proof.

Thus, suppose $\bar{\pi}^t\neq c^s$ and consider the right triangle formed by π_0^t, $\bar{\pi}^t$, and c^s. By (5), we can restrict attention to the two cases $\bar{\pi}^t\in conv(\pi_0^t,\tilde{\pi}^t)$ and $\tilde{\pi}^t\in conv(\pi_0^t,\bar{\pi}^t)$. If $\bar{\pi}^t\in conv(\pi_0^t,\tilde{\pi}^t)$ then $\bar{\pi}^t\in conv(Q^{t+1})$ since $\pi_0^t,\tilde{\pi}^t\in Q^{t+1}$ by the operation of VALIDITY. Letting θ be the angle between the vectors $(c^s-\pi_0^t)$ and $(\tilde{\pi}^t-\pi_0^t)$, we have by the definition of $\bar{\pi}^t$ that $\|c^s-\bar{\pi}^t\|/\|c^s-\pi_0^t\|=\sin\theta$. Letting $\tilde{\pi}_p^t$ be the projection of $\tilde{\pi}^t$ onto the line defined by π_0^t and c^s, we also have $\cos\theta=\|\tilde{\pi}_p^t-\pi_0^t\|/\|\tilde{\pi}^t-\pi_0^t\|$. By Proposition 1.2.2, $\|\tilde{\pi}^t-\pi_0^t\|\leq 2/r$, and by (5),

$$\|\tilde{\pi}_p^t-\pi_0^t\|=\frac{(c^s-\pi_0^t)}{\|c^s-\pi_0^t\|}\tilde{\pi}^t-\frac{(c^s-\pi_0^t)}{\|c^s-\pi_0^t\|}\pi_0^t\geq\frac{1}{2RNCn}.$$

Thus, $\cos\theta\geq r/4RNCn$ and

$$\frac{\|c^s-\bar{\pi}^t\|}{\|c^s-\pi_0^t\|}=\sin\theta=(1-\cos^2\theta)^{1/2}\leq\left[1-\left(\frac{r}{4RNCn}\right)^2\right]^{1/2}.$$

Finally, consider the case $\tilde{\pi}^t\in conv(\pi_0^t,\bar{\pi}^t)$. By the definition of $\bar{\pi}^t$, this can only occur if $\tilde{\pi}^t$ is contained in the sphere of radius $\|c^s-\pi_0^t\|/2$ centered between c^s and π_0^t. By (3) and (4), it follows that $\|c^s-\tilde{\pi}^t\|\leq\frac{1}{\sqrt{2}}\|c^s-\pi_0^t\|$. With $\tilde{\pi}^t\in\Pi_1$ by the operation of SEPSTEP, the proof is complete. □

It remains only to show that condition (3) assumed in the proof of Theorem 3.2 is guaranteed by the operation of the algorithm, and this is demonstrated in Lemma 3.4. However, before proving this lemma we first make note of the following corollary to Theorem 3.2, which is the main result of this section.

THEOREM 3.3. *If $c^s\notin\Pi_1$, then after at most*

$$t=2\left(\frac{4RNCn}{r}\right)^2\left|\log_e\left(\frac{r}{2RNCn}\right)\right| \tag{6}$$

iterations, VALIDITY will return an $\tilde{x}\in\mathcal{P}$ satisfying $c^s\tilde{x}>1$.

Proof. By the operation of MAIN, VALIDITY is not called if $\|c^s\|>1/r$, and since $\|\pi_0^0\|\leq 1/r$ by Proposition 1.2.2, $\|c^s-\pi_0^0\|\leq 2/r$. Using this fact and the bound $-\log_e a\geq 1-a$ we find that if

$$t=2\left(\frac{4RNCn}{r}\right)^2\left|\log_e\left(\frac{r}{2RNCn}\right)\right|>2\log_e\left(\frac{r}{2RNCn}\right)\log_e^{-1}\left[1-\left(\frac{r}{4RNCn}\right)^2\right],$$

then $\|c^s-\pi_0^{t+1}\|<(RNCn)^{-1}$, causing VALIDITY to return as specified. □

LEMMA 3.4. *If SEPSTEP returns a valid inequality $\tilde{\pi}x\leq 1$ for $\mathcal{P}$ then*

$$dc^s-d\tilde{\pi}=\frac{(c^s-\pi_0)c^s-(c^s-\pi_0)\tilde{\pi}}{\|c^s-\pi_0\|}\leq\frac{1}{2RNCn}.$$

Proof. Since $\|u^0_{LB}d\| \leq r$ and $\|u^0_{UB}d\| > R$ it follows by assumption 2 that $u^0_{LB}d \in \mathcal{P}$ while $u^0_{UB}d \notin \mathcal{P}$, and by the operation of SEPSTEP $u^h_{LB}d \in \mathcal{P}$ and $u^h_{UB}d \notin \mathcal{P}$ are maintained throughout.

Let $H = \log 2R^2r^{-2}NCn$, the number of iterations performed by the procedure when a separating hyperplane is returned. Since $\tilde{\pi}x \leq 1$ separates $u^H_{UB}d$ from $\mathcal{P}$, $\tilde{\pi}(u^H_{UB}d) > 1$. Further, $c^s(u^H_{LB}d) \leq 1$, for otherwise the procedure would return $\tilde{x}$ instead of $\tilde{\pi}$. Thus,

$$dc^s - d\tilde{\pi} < \frac{1}{u^H_{LB}} - \frac{1}{u^H_{UB}} = \frac{u^H_{UB} - u^H_{LB}}{u^H_{LB}u^H_{UB}} \leq \frac{u^H_{UB} - u^H_{LB}}{(u^H_{LB})^2}.$$

Given that $u^0_{UB} - u^0_{LB} = R$, it follows that after H iterations of SEPSTEP $u^H_{UB} - u^H_{LB} \leq R/2^H$; and since u^h_{LB} can only increase, $u^H_{LB} \geq r$. Together, these facts imply $dc^s - d\tilde{\pi} \leq R/2^Hr^2$, which in turn implies the desired result. □

4. An optimal solution from MAIN. At the conclusion of MAIN, v^* is the optimal solution value of (P), but the point $\tilde{x}$ maintained by the algorithm is only nearly optimal. A common way of purifying a near optimal solution is to begin by perturbing the objective function by a small amount so that an optimal solution can be obtained simply by rounding the near optimal solution appropriately. This perturbation poses no difficulty for algorithms having a complexity polynomial in the size of the objective function, but since the algorithm presented here is pseudopolynomial in the size of the objective function an alternative purification method is required.

As mentioned in §3, when MAIN terminates $\tilde{x}$ is not only feasible for (P) but $c\tilde{x} > v^* - 1/M$; that is, $\tilde{x}$ has objective function value greater than any extreme point solution of (P) other than any optimal solutions. All OPTIMAL must do, therefore, is find an extreme point solution of (P) with objective function value at least as large as that of $\tilde{x}$, and this extreme point is guaranteed to be optimal for (P). Conceptually the process is quite straightforward, consisting of modifying the point $\tilde{x}$ through a sequence of nondescent moves until after n such moves an extreme point is found. This is the basis of procedure OPTIMAL.

5. Analysis. It remains to consider the complexity of the overall algorithm. Here we follow the development of recent interior point methods and focus on the number of arithmetic operations performed in the course of the algorithm without addressing the bit complexity of these operations. A bitwise development would not fundamentally impact the stated results.

CLOSE inherits the complexity of the algorithm of Kojima, Mizuno, and Yoshise, which is $O(n^3L)$, where L is the size of the data defining the linear complementarity problem that must be solved. The entries in c^s and Π predominate so that L is $O(n \log C + n^2 \log g)$, where g is an upper bound on the absolute value of the numerators and denominators in the coefficients of the π^t_k found in Q^t. Equivalently, g is an upper bound on the absolute value of the numerators and denominators found in the $\tilde{\pi}$ returned by SEPARATE. Using this bound on L we have

$$O(n^3L) = O(n^4 \log C + n^5 \log g). \tag{7}$$

The complexity of SEPSTEP is also intimately related to the complexity of SEPARATE but directly on the running time G of this algorithm rather than the size $\log g$ of numbers occurring in its output. The amount of work per iteration of SEPSTEP is constant other than the work performed in SEPARATE, and so the

overall complexity is

$$O(G \log 2R^2 r^{-2} NCn). \tag{8}$$

VALIDITY performs constant work other than the work involved in initialization and the work performed in calls to the procedures CLOSE and SEPSTEP, and by Theorem 3.3 the maximum number of iterations of VALIDITY during any given call is given by (6).

OPTIMAL is called only once at the conclusion of MAIN and performs n iterations. The dual interpretation of step 1 is that of maximizing d^q on Π_1 using SEPARATE as a validity oracle. Maximizing a linear function on a bounded polyhedron in R^n contained in a ball of radius $1/r$ (see Proposition 1.2.2) with constraints of encoding length bounded by $2n \log \max(M, N)$ (see assumption 4) and objective function d^q can be performed in time polynomial in $\log 1/r$, n, $\log \max(M, N)$, and the encoding length of d^q via binary search; see, for example, [12]. The process generates an optimal extreme point of Π_1 with encoding length polynomial in n and $2n \log \max(M, N)$. Step 2 of OPTIMAL consists of finding a solution d^q to a $q \times n$ system of equations, each of which has encoding length bounded by a polynomial function of n and $2n \log \max(M, N)$ by the operation of step 1. A well-known result of Edmonds (see [12]) implies that the direction d^q generated by solving this system of equations therefore has encoding length polynomial in n and $2n \log \max(M, N)$.

Finally, MAIN performs constant work except for calls to VALIDITY and a single call to OPTIMAL, and as a binary search algorithm it performs $\log NCn$ iterations. The overall complexity of the proposed algorithm is given by the product of $\log NCn$, (6), and the sum of (7) and (8), plus the work done by OPTIMAL. The algorithm is therefore a polynomial function of R, r, N, C, n, $\log 1/r$, $\log \max(M, N)$, $\log g$, and G, where standard assumptions on the separation oracle imply $\log g$ is polynomially related to G. For integer programs with 0/1 vertices, it is trivially verified that N, M, and R are polynomial in n. Thus, for 0/1 integer programs the proposed algorithm is polynomial in r, C, n, and the running time of the separation oracle.

In many 0/1 integer programs the existence of a polynomially bounded ball contained within the feasible set $\mathcal{P}$ is easily verified. One particular class of such problems is that of *independence system* polyhedra, which are defined by the property that if $x \in \mathcal{P}$ and $y \leq x$ then $y \in \mathcal{P}$. It is easily seen that full dimensional 0/1 independence system polyhedra contain the origin and all of the unit coordinate vectors, and therefore they contain a ball of radius $r = (n+1)^{-3/2}$ centered at the point $(\frac{1}{n+1}, \ldots, \frac{1}{n+1})$. As a consequence, the proposed algorithm is polynomial in C and the running time of the separation oracle for optimization problems on these polyhedra. In particular, the proposed algorithm represents a generic polynomial cutting plane procedure for the cardinality versions of such problems as matching, T-join, and matroid intersection, among others.

6. Conclusions. A cutting plane algorithm has been presented and analyzed for solving linear programs where the feasible region is represented by a separation oracle, a condition common to many integer and mixed-integer programs. It was demonstrated that for problems containing a ball of polynomially bounded radius the algorithm is polynomial in the running time of the separation oracle and pseudopolynomial in the size of the objective function, and consequences of the algorithm were discussed.

A number of obstacles preclude the use of the algorithm in practice. First, a linear complementarity problem is solved during every iteration of VALIDITY. While

the linear complementarity problems that are encountered are relatively small and each problem in the sequence varies only slightly, they remain a significant bottleneck to the operation of the algorithm. Second, the algorithm is primarily a constraint validity algorithm molded into an optimization algorithm through the use of binary search. While the use of binary search can be mitigated through problem-specific knowledge of an optimal or near optimal solution, as a rule, algorithms based on binary search do not perform as well as more direct methods.

What the algorithm does show, however, is that it is possible to construct a cutting plane algorithm whose complexity does not depend on the number of facets of the underlying polyhedron. Instead, the complexity of the algorithm is fundamentally interrelated with the numerical properties of the extreme points and the objective function. This alternative perspective makes it possible to show that the proposed algorithm is polynomial in some instances where alternative cutting plane algorithms are not. Given the predominance of cutting plane algorithms in applied integer programming, it is hoped that further results along these lines can be developed.

Acknowledgments. The author gratefully acknowledges the Department of Mathematics at the University of Houston for providing facilities which aided immensely in the development and preparation of the present work.

REFERENCES

[1] D. Applegate and W. Cook (1991), *A computational study of the job-shop scheduling problem*, ORSA J. Comput., 3, pp. 149–156.

[2] E. Balas, S. Ceria, and G. Cornuéjols (1993), *A lift-and-project cutting plane algorithm for mixed* 0–1 *programs*, Math. Programming, 58, pp. 295–324.

[3] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty (1993), *Nonlinear Programming: Theory and Algorithms*, Wiley and Sons, New York.

[4] R. G. Bland, D. Goldfarb, and M. J. Todd (1981), *The ellipsoid method: A survey*, Oper. Res., 29, pp. 1039-1091.

[5] E. A. Boyd (1995), *On the convergence of Fenchel cutting planes in mixed-integer programming*, SIAM J. Optim., 5, pp. 421–435.

[6] V. Chvátal (1973), *Edmonds polytopes and a hierarchy of combinatorial problems*, Discrete Math., 4, pp. 305–337.

[7] H. Crowder, E. L. Johnson, and M. W. Padberg (1983), *Solving large-scale zero-one linear programming problems*, Oper. Res., 31, pp. 803–834.

[8] A. Frank and E. Tardos (1987), *An application of simultaneous diophantine approximation in combinatorial optimization*, Combinatorica, 7, pp. 49–65.

[9] D. Goldfarb and M. J. Todd (1989), *Linear Programming*, in Optimization, G. L. Nemhauser et al., eds., Elsevier Science Publishers, New York.

[10] R. E. Gomory (1958), *Outline of an algorithm for integer solutions to linear programs*, Bull. Amer. Math. Soc., 64, pp. 275–278.

[11] ——— (1963), *An algorithm for integer solutions to linear programs*, in Recent Advances in Mathematical Programming, R. L. Graves and P. Wolfe, eds., McGraw Hill, New York, pp. 269–302.

[12] M. Grötschel, L. Lovász, and A. Schrijver (1988), *Geometric Algorithms and Combinatorial Optimization*, Springer–Verlag, New York.

[13] K. L. Hoffman and M. W. Padberg (1991), *Improving the LP-representation of zero-one linear programs for branch-and-cut*, ORSA J. Comput., 3, pp. 121–134.

[14] M. Kojima, S. Mizuno, and A. Yoshise (1989), *A polynomial-time algorithm for a class of linear complementarity problems*, Math. Programming, 44, pp. 1–26.

[15] ——— (1991), *An $O(\sqrt{n}L)$ iteration potential reduction algorithm for linear complementarity problems*, Math. Programming, 50, pp. 331–342.

[16] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds. (1985), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley and Sons, New York.

[17] L. Lovász and A. Schrijver (1991), *Cones of matrices and set-functions and* 0–1 *optimization*, SIAM J. Optim., 1, pp. 166–190.

[18] G. L. Nemhauser and L. A. Wolsey (1988), *Integer and Combinatorial Optimization*, Wiley and Sons, New York.

[19] M. Padberg and M. Grötschel (1985), *Polyhedral computations*, in The Traveling Salesman Problem, E. L. Lawler et al., eds., Wiley and Sons, New York.

[20] M. Padberg and G. Rinaldi (1991), *A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, SIAM Rev., 33, pp. 60–100.

[21] W. P. Savelsbergh, G. C. Sigismondi, and G. L. Nemhauser (1991), *MINTO, A Mixed Integer Optimizer*, Tech. report COC-91-04, Computational Optimization Center, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

[22] J. F. Shapiro (1979), *Mathematical Programming: Structures and Algorithms*, Wiley and Sons, New York.

[23] H. Sherali and W. Adams (1990), *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM J. Discrete Math., 3, pp. 411–430.

[24] E. Tardos (1986), *A strongly polynomial algorithm to solve combinatorial linear programs*, Oper. Res., 34, pp. 250–256.

SIAM J. DISCRETE MATH.
Vol. 9, No. 3, pp. 377–392, August 1996

004

COHEN–MACAULAY RINGS IN NETWORK RELIABILITY*

JASON I. BROWN†, CHARLES J. COLBOURN‡, AND DAVID G. WAGNER‡

Abstract. For any simplicial complex Δ and field K, one can associate a graded K-algebra $K[\Delta]$ (the *Stanley–Reisner ring*). For certain Δ and K, the Stanley–Reisner rings have a *homogeneous system of parameters*, Θ, such that $K[\Delta]/\langle\Theta\rangle$ is finite-dimensional, and coefficients of its Hilbert series are the h-vector of Δ. The previous constructions of Θ were noncombinatorial. In the special case of cographic matroids, we give (for *any* field K) a combinatorial description of a homogeneous system of parameters in terms of the graph structure, as well as an explicit basis for the resulting quotient algebra. The results have applications to a central problem of reliability, namely the association of a multicomplex to a connected graph, such that the reliability is a simple function of the rank numbers.

Key words. reliability, graph, Cohen–Macaulay ring, homogeneous system of parameters, Gröbner basis

AMS subject classifications. 05C30, 05E99, 68M15

1. Introduction.

Let G be a finite undirected connected graph of order n (i.e., with exactly n vertices). Assume that each edge of G is independently operational with probability $p \in [0,1]$ (we denote the failure probability of an edge by $q = 1-p$). The *reliability* of G, $\mathrm{Rel}(G,p)$, is the probability that the graph is connected, i.e., the probability that the operational edges form a spanning connected subgraph of G. This model of reliability (sometimes called *all-terminal reliability* in the literature) has been well studied [12], with the preponderance of work carried out on techniques for efficiently bounding the reliability function. One of the best techniques for bounding reliability is due to Ball and Provan [1] and relies on Stanley's [21] inequalities for the h-vectors of shellable complexes. The essence is that there is a Cohen–Macaulay standard graded algebra $A = A(G)$ whose Hilbert series, $\mathrm{Hilbert}(A,x)$, has the property that

$$\mathrm{Rel}(G,p) = p^{n-1}\mathrm{Hilbert}(A, 1-p).$$

What is not explicit is how the construction can be carried out combinatorially to find A. It is this problem that we consider here.

2. Combinatorial background: Complexes, matroids, and order sets of monomials.

A *complex* Δ on a finite set $X = \{x_1, \ldots, x_m\}$ is simply a collection of subsets (often called *faces* or *independent sets*) of X closed under containment. For an independent set σ of complex Δ, we let $\overline{\sigma}$ denote the set of subsets of σ (of course, $\overline{\sigma} \subseteq \Delta$). If σ_1 and σ_2 are any two independent sets of Δ with $\sigma_1 \subseteq \sigma_2$, the *interval* $[\sigma_1, \sigma_2] = \{\sigma : \sigma_1 \subseteq \sigma \subseteq \sigma_2\}$. The *dimension* of a complex Δ is equal to the cardinality of the largest set in Δ (this is one more than the definition of dimension usually found in the literature, but will be convenient for our combinatorial applications). We say that Δ is *purely d-dimensional* if Δ is d-dimensional and every maximal independent set (or *facet*) of Δ has cardinality d. A purely d-dimensional complex Δ is *shellable*

* Received by the editors July 6, 1994; accepted for publication (in revised form) August 14, 1995.
† Department of Mathematics, Statistics, and Computing Science, Dalhousie University, Halifax, Nova Scotia, Canada B3H 3J5 (brown@cs.dal.ca).
‡ Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1 (cjcolbou@math.uwaterloo.ca and dgwagner@math.uwaterloo.ca).

if its facets can be ordered as $\sigma_1, \ldots, \sigma_t$ with the property that for $i = 2, \ldots, t$, $\overline{\sigma_i} \cap (\bigcup_{j=1}^{i-1} \overline{\sigma_j})$ is purely $(d-1)$-dimensional. For any shelling $\sigma_1, \ldots, \sigma_t$,

$$\overline{\sigma_i} - \bigcup_{j<i} \overline{\sigma_j}$$

is an interval $[\tau_i, \sigma_i]$, and these intervals partition Δ (cf. [4]); this interval partition is called the *interval partition corresponding to the shelling* $\sigma_1, \ldots, \sigma_t$.

A complex Δ is a *matroid* if the well-known exchange axiom holds (cf. [24]). One well-known class of matroids is that of *cographic matroids*. Given a graph G, the complex on the edges of G whose independent sets consist of those sets of edges whose removal does not increase the number of components is in fact a matroid, the *cographic matroid* of G (which we usually denote by Cog_G).

The *f-vector* of a d-dimensional complex Δ is $(f_0, \ldots, f_d)$, where f_i is the number of independent sets of cardinality i. The *h-vector* of Δ is $(h_0, \ldots, h_d)$, where h_i is given by

$$h_i = \sum_{j=0}^{i} (-1)^{i-j} \binom{d-j}{d-i} f_j.$$

(h-vectors have played an important role in polytope theory, primarily with regards to the Upper Bound Theorem.) The h-vector of a shellable complex is always non-negative and has no internal zeros (the reasons for this will soon become apparent). Let

$$f(\Delta, x) = \sum_{i=0}^{d} f_i x^i$$

and

$$h(\Delta, x) = \sum_{i=0}^{d} h_i x^i$$

(these are the generating functions for the f-vector and h-vector, respectively). It is not hard to verify that

$$h(\Delta, x) = (1-x)^d f\left(\Delta, \frac{x}{1-x}\right), \tag{1}$$

and hence

$$\sum_{i=0}^{d} h_i = f_d.$$

When $\Delta = \mathrm{Cog}_G$, the cographic matroid of a connected graph G, one sees that

$$\mathrm{Rel}(G, p) = p^m f\left(\mathrm{Cog}_G, \frac{1-p}{p}\right) = p^{n-1} h(\Delta, 1-p),$$

and the sum of the terms in the h-vector is simply the number of spanning trees of G.

Finally, we can extend complexes to multicomplexes as follows. Let $x_1, \ldots, x_l$ be commuting indeterminates. We let $\mathrm{Mon}(x_1, \ldots, x_l)$ denote the set of monomials in $x_1, \ldots, x_m$. An *order set of monomials* in variables $x_1, \ldots, x_l$ is a subset M of $\mathrm{Mon}(x_1, \ldots, x_l)$ closed under division ('|'), i.e., if $m_1,\ m_2 \in \mathrm{Mon}(x_1, \ldots, x_l)$, $m_1 \in M$ and $m_2 | m_1$, then $m_2 \in M$. In an obvious way, any order set of square-free monomials corresponds to a complex, and hence order sets of monomials are sometimes called *multicomplexes.*

If we are given $N \subseteq \mathrm{Mon}(x_1, \ldots, x_l)$, we can define an order set of monomials $-N$ by "chopping out" from $\mathrm{Mon}(x_1, \ldots, x_l)$ all monomials that are divisible by any monomial in N, that is,

$$-N = \{m \in \mathrm{Mon}(x_1, \ldots, x_l) : (\forall m' \in N)(m' \nmid m)\}.$$

We call N a set of *choppers* on the variables $x_1, \ldots, x_l$ for the order set of monomials $-N$.

3. Algebraic background: Stanley–Reisner rings. We turn now to commutative algebra, and assume that the reader has a basic knowledge of rings. Let K be any field. A commutative ring A containing K is a *standard graded K-algebra* if A is a vector space direct sum

$$A = \bigoplus_{i \geq 0} A_i$$

of subspaces $A_0, \ldots$, such that

(i) $A_0 = K$,

(ii) $A_i A_j \subseteq A_{i+j}$ for all $i, j \geq 0$, and

(iii) there are finitely many elements $z_1, \ldots, z_t$ in A_1 such that every element of A can be written as a polynomial in $z_1, \ldots, z_t$ with coefficients in K; for such an A, we call the elements of A_i *homogeneous of degree i.* The *Hilbert series* for the standard graded K-algebra A is the generating series for the dimensions of the vector spaces A_i, i.e.,

$$\mathrm{Hilbert}(A, x) = \sum_{i \geq 0} \mathrm{Dim}_K(A_i) x^i.$$

If I is an ideal of ring A, $z \in A$, and $\phi : A \to A/I$ is the natural homomorphism, we often write simply z for $\phi(z)$. If J' is an ideal of A/I and $J = \phi^{-1}(J')$, then clearly $(A/I)/J' \equiv A/(I + J) = A/\langle I \cup J \rangle$, and so we don't distinguish between these. We also write A (modulo $(I + J)$) for the ring $(A/I)/J'$.

We now associate a standard graded K-algebra to any complex (further details can be found, for example, in [3]). Let Δ be a complex of dimension d on a set $X = \{x_1, \ldots, x_m\}$. $K[\underline{x}] = K[x_1, \ldots, x_m]$ denotes the (commutative) polynomial ring over K in indeterminates $x_1, \ldots, x_m$; recall that $\mathrm{Mon}(x_1, \ldots, x_m)$ denotes the set of monomials in $x_1, \ldots, x_m$. For a subset Y of X, we denote

$$\sum Y = \sum_{x_i \in Y} x_i$$

and

$$\prod Y = \prod_{x_i \in Y} x_i.$$

The *Stanley–Reisner ring* of the complex Δ over K is

$$K[\Delta] = K[\underline{x}]/\mathrm{Circ}(\Delta),$$

where $\mathrm{Circ}(\Delta) = \langle\{\prod Y : Y \subseteq X,\ Y \notin \Delta\}\rangle$ is the ideal generated by all (minimal) subsets of X not in Δ (such minimal subsets are called *circuits* of Δ). Since $K[x_1, \ldots, x_m]$ is a standard graded K-algebra and $\mathrm{Circ}(\Delta)$ is generated by homogeneous polynomials, $K[\Delta]$ is also a standard graded K-algebra, and can therefore be expressed as a vector space direct sum

$$K[\Delta] = \bigoplus_{i \geq 0} R_i,$$

where R_i is generated by all monomials of degree i. It is easy to verify that the vector space dimension of R_i is $\sum_{j=0}^{d} \binom{i-1}{j-1} f_j$. This is also the coefficient of x^i in $f(\Delta, \frac{x}{1-x})$. From

$$\mathrm{Hilbert}(K[\Delta], x) = \sum_{i \geq 0} \mathrm{Dim}_K(R_i) x^i,$$

the preceding fact, and (1), we see that

$$h(\Delta, x) = (1-x)^d \mathrm{Hilbert}(K[\Delta], x). \tag{2}$$

This connects the h-vector of a complex to the Hilbert series of the associated Stanley–Reisner ring.

Now any quotient of a polynomial ring has a basis that is an order set of monomials. Stanley's argument [20] of this fact (which was first proved by Macaulay [18]) uses a greedy algorithm on a term ordering on all monomials to choose a maximal linearly independent set of monomials. A *term ordering* $<$ on all monomials in a finite set of variables is any linear ordering of them with the property that

(i) $1 \leq m$ for any monomial m, and

(ii) if $m_1 < m_2$, then $m_1 \cdot u < m_2 \cdot u$ for any monomials m_1, m_2, and u.

For example, a lexicographic (or reverse lexicographic) ordering on each set of monomials of the same degree, with monomials of lower degree preceding those of higher degree, is a term ordering. Another viewpoint for finding an order set of monomials that is a basis is afforded via Gröbner bases, and we will return to this later.

We want now to associate an order set of monomials such that the number of monomials of degree i equals h_i, the ith term in the h-vector of the complex Δ. From (2) and the above, if we can show that $(1-x)^d \mathrm{Hilbert}(K[\Delta], x)$ is again the Hilbert series of a quotient of the polynomial ring, then we will derive a monomial basis with the required properties. However, it is not necessarily true that for any standard graded K-algebra A, $(1-x)^d \mathrm{Hilbert}(A, x)$ is again a Hilbert series. Fortunately, for certain complexes and fields, the Stanley–Reisner rings do in fact satisfy this condition.

A set of homogeneous elements $\Theta = \{\theta_1, \ldots, \theta_d\}$ of $K[\underline{x}]$ is called a *homogeneous system of parameters (h.s.o.p.)* for $K[\Delta]$ if $K[\Delta]/\langle\theta_1, \ldots, \theta_d\rangle$ is a **finite**-dimensional vector space over K, that is,

$$K[\underline{x}]/\langle \mathrm{Circ}(\Delta) \cup \{\theta_1, \ldots, \theta_d\}\rangle = \bigoplus_{i=0}^{d} R'_i,$$

where for $i = 0, \ldots, d$, R'_i is generated by all monomials of degree i; if, furthermore, each θ_i has degree 1, then Θ is a h.s.o.p. of degree 1. If Δ is any d-dimensional

complex then, provided K is infinite, Stanley noted that $K[\Delta]$ does in fact have a h.s.o.p. of degree 1, though his argument [22] does not explicitly construct one (we shall return to this in the next section).

The Stanley–Reisner rings of shellable complexes (over any field) are known to be *Cohen–Macaulay*, and it can be shown in these cases that for any h.s.o.p. $\{\theta_1, \ldots, \theta_d\}$ of degree 1,

$$\begin{aligned}\text{Hilbert}(K[\Delta]/\langle\{\theta_1, \ldots, \theta_d\}\rangle, x) &= (1-x)^d \text{Hilbert}(K[\Delta], x) \\ &= h(\Delta, x) \qquad\qquad \text{(from (2))}\end{aligned}$$

(see [3] for a definition of Cohen–Macaulay rings and their relation to Stanley–Reisner rings). Now

$$K[\Delta]/\langle\{\theta_1, \ldots, \theta_d\}\rangle \cong K[\underline{x}]/\langle \text{Circ}(\Delta) \cup \{\theta_1, \ldots, \theta_d\}\rangle$$

is clearly the quotient of the polynomial ring $K[\underline{x}]$, and thus we can, for any shellable complexes Δ, associate an order set of monomials, such that the number of monomials of degree i in the set equals the ith term of the h-vector of the complex (Stanley [19] used this result to derive certain inequalities on the terms in the h-vector of a shellable complex, and thereby prove the upper bound conjecture for convex polytopes). This association has applications as well to network reliability (cf. [1], [5]–[8]) since, if G is a connected graph of order n and size m, then its reliability can be expressed as

$$\text{Rel}(G, p) = p^{n-1} \sum_{i=0}^{m-n+1} h_i (1-p)^i,$$

where $\langle h_0, \ldots, h_{m-n+1}\rangle$ is the h-vector of the cographic matroid of G. Ball and Provan [1] used Stanley's inequalities for h-vectors with this polynomial expansion to derive strong upper and lower bounds for network reliability.

What is fascinating is that this association between two types of combinatorial objects (shellable complexes and order sets of monomials) goes through commutative ring theory as described above, and it is not known how to carry out the algebraic steps in a purely combinatorial setting. Stanley also showed that for shellable complexes it is not necessarily true that for finite K such a h.s.o.p. of degree 1 exists. In the next section we turn to cographic matroids. We show for this case that homogeneous systems of parameters of degree 1 exist for *any* field K, and we give an explicit combinatorial construction for them.

4. H.S.O.P.s for cographic matroids. We begin with a useful result of Stanley's which determines when $K[\Delta]$ has a homogeneous system of parameters of degree 1.

PROPOSITION 4.1 (see [22]). *Let Δ be a simplicial complex of dimension d over $X = \{x_1, \ldots, x_m\}$. Let $\theta_1, \ldots, \theta_d$ be any homogeneous element of degree 1 in $K[\Delta]$, with $\theta_i = \sum_{j=1}^m a_{i,j} x_j$. Then $\theta_1, \ldots, \theta_d$ is a homogeneous system of parameters (of degree 1) if and only if the $d \times m$ matrix M, whose (i,j)th entry is $a_{i,j}$, has the property that for every facet $F \in \Delta$, the $d \times d$ submatrix m_F of M consisting of the columns of M corresponding to the edges in F is nonsingular.*

In particular, if K is infinite, we can clearly choose such a matrix M so that *every* $d \times d$ submatrix is nonsingular, and hence every simplicial complex has a h.s.o.p. of degree 1 over any infinite field.

We need to recall another standard definition from matroid theory (for more details, see [24]). Let Δ be a matroid on a set X of cardinality m. A *representation* over field K of Δ is a function π that maps X into some vector space V over K such that $Y \in \Delta$ if and only if $\pi(Y)$ (considered as a multiset of vectors) is linearly independent in V (that is, π preserves rank). Now if Δ is a matroid that is representable over field K, then there exists a $d \times m$ matrix M such that the submatrix consisting of d columns of M is nonsingular (i.e., of full rank) if and only if the elements of M indexing these columns form a basis of M. We deduce immediately from this and Proposition 4.1 the following.

COROLLARY 4.2. *Let Δ be a matroid of dimension d over $X = \{x_1, \ldots, x_m\}$. Let $\theta_1, \ldots, \theta_d$ be any homogeneous elements of degree 1 in $K[\Delta]$, with $\theta_i = \sum_{i=1}^{m} a_{i,j} x_j$. Then $\theta_1, \ldots, \theta_d$ is a homogeneous system of parameters (of degree 1) for $K[\Delta]$ if the map $\pi : \Delta \to K^d : x_i \mapsto (a_{1,i}, \ldots, a_{d,i})^t$ is a representation of Δ over K.*

Thus for a matroid Δ representable over a field K, $K[\Delta]$ has a homogeneous system of parameters of degree 1. We remark that the converse is not true, as it was observed earlier that any simplicial complex has a homogeneous system of parameters of degree 1 over any infinite field, while there are matroids that are not representable over *any* field.

We turn specifically now to cographic matroids. Let G be a connected graph of order n with edge set $E = \{e_1, \ldots, e_m\}$, and set $d = m-n+1$. Let K be any field. As in the previous section, $K[\mathrm{Cog}_G]$ denotes the graded algebra $K[e_1, \ldots, e_m]/\mathrm{Circ}(\mathrm{Cog}_G)$, where $\mathrm{Circ}(\mathrm{Cog}_G) = \langle \{\prod Y : Y \notin \mathrm{Cog}_G\} \rangle$ is the ideal generated by all (minimal) subsets of E not in Cog_G (i.e., all edge cutsets of G). From the previous section, we know that if K is infinite, $K[\mathrm{Cog}_G]$ has a h.s.o.p. of degree 1. We now proceed to find a h.s.o.p. of degree 1 for *any* field K.

As in [23], we fix an orientation of the edges of graph G, and for every circuit of G, we orient it in one of its two directions. We form the *circuit matrix* $C = C(G)$ whose columns are indexed by the edges $E = \{e_1, \ldots, e_m\}$ of G, whose rows are indexed by the circuits $C_1, \ldots, C_t$ of G, and whose (i,j)th entry is

(i) 1 if $e_j \in C_i$ and the orientation of e_j agrees with its orientation in C_i,
(ii) -1 if $e_j \in C_i$ and the orientation of e_j disagrees with its orientation in C_i, and
(iii) 0 if $e_j \notin C_i$.

Let T be a fixed spanning tree of G, and for $e \notin E - T$, let $\mathcal{C}_e$ be the fundamental cycle of $T + e$. A well-known result (see Theorem 6.11 of [23]) states that if B is a submatrix of the circuit matrix C with d rows and rank d, then a square $d \times d$ submatrix B' of B is nonsingular if and only if the rows of B' correspond to the edges of the complement of some spanning tree of G (actually, the argument there is stated with the field being $\mathbb{Q}$, but it holds as well for any field K). This is equivalent to B being a representation of the cographic matroid Δ of G over K.

Let us now take M_T to be the $d \times m$ submatrix of C whose rows correspond to the fundamental cycles of T. M_T has rank d, since the $d \times d$ submatrix on the columns not in T is a diagonal matrix with all diagonal entries either 1 or -1. Moreover, F is a basis of Δ if and only if it is the complement of a spanning tree of G. Hence M_T is a representation of Δ, and we deduce from Corollary 4.2 the following combinatorial description of a h.s.o.p. of degree 1 for Δ.

THEOREM 4.3. *Let G be a connected graph of order n with edge set E. Fix a*

spanning tree T of G, and for each edge $e \in E - T$, set

$$\theta_e = \sum_{f \in \mathcal{C}_e} c_f^e f,$$

where for $f = x_k \in \mathcal{C}_e$, c_f^e is the entry in matrix C corresponding to edge f and circuit $\mathcal{C}_e$. Then $\{\theta_e : e \in E - T\}$ is a homogeneous system of parameters (of degree 1) for $K[Cog_G]$. □

This result is important in that while Stanley's work states that such a homogeneous system of parameters exists for $K[\mathrm{Cog}_G]$, *provided K is infinite*, we have described an explicit (combinatorial) construction of such a system *for any field K*! In fact, there are matroids M and finite fields K for which $K[M]$ does not have a h.s.o.p. of degree 1 (while it has a h.s.o.p.). For example, Stanley [22] states that for a shellable complex Δ of dimension 2, $K[\Delta]$ has a h.s.o.p. of degree 1 if and only if Δ is $(|K|+1)$-colorable (in the usual graph-theoretic sense). Now if we take for M any complete l-partite graph where $l \geq |K|+2$, then clearly M is not $(|K|+1)$-colorable, and so $K[\Delta]$ has no h.s.o.p. of degree 1. Why does this not contradict our theorem above? The fact is that for $d = 2$, the possible cographic matroids are complete bipartite or tripartite (as they are the K_1-bond of two cycles or a theta graph), and as any field K has at least two elements, such a matroid is always $(|K| + 1 \geq 3)$-colorable.

The net result of Theorem 4.3 is that if we are interested in the h-vector of Cog_G, we can work over $\mathbb{Z}_2$ and use the h.s.o.p.

$$\left\{ \sum_{f \in \mathcal{C}_e} f : e \in E - T \right\}$$

as in $\mathbb{Z}_2$, $-1 = 1$. This simplifies the arithmetic, and in the following sections, we focus in on $K = \mathbb{Z}_2$.

5. A basis for $K[\mathbf{Cog}_G]/\langle\Theta\rangle$. Now once we have the h.s.o.p. Θ of degree 1 (from the previous section) for the cographic matroid Cog_G of G, we can form

$$K[\mathrm{Cog}_G]/\langle\Theta\rangle.$$

It follows (see [3]) that we can find a vector space decomposition

$$K[\mathrm{Cog}_G]/\langle\Theta\rangle = R'_0 \oplus R'_1 \oplus \cdots \oplus R'_d,$$

where $\langle \dim_K(R'_i)\rangle$ is the h-vector of Cog_G. Now

$$K[\mathrm{Cog}_G]/\langle\Theta\rangle = K[e_1, \ldots, e_m]/J,$$

where J is the ideal generated by

$$\mathrm{Circ}(\Delta) \cup \Theta.$$

We can now use Stanley's greedy algorithm (over any field) to find monomial bases for each of the R'_i's. On the other hand, we can attempt to directly find a set of monomials that form a basis for $K[\mathrm{Cog}_G]/\langle\Theta\rangle$, hopefully with a combinatorial interpretation in terms of the graph. We shall do so presently, but first we need some graph-theoretic definitions.

For connected graph G of order n, let $<$ be a fixed linear order on the edge set $E = E(G)$ of G. Let T be a spanning tree of G, and let $e \in E - E(T)$. e is *externally active* if e is the least edge (under the ordering $<$) in the unique cycle in $T + e$, and E is *externally passive* otherwise. $\mathrm{EP}(T)$ denotes the set of externally passive edges of spanning tree T. Let H_i denote the number of spanning trees of G with precisely i externally passive edges. Then [13]

$$\mathrm{Rel}(G, p) = p^{n-1} \sum_{i=0}^{m-n+1} H_i (1-p)^i,$$

i.e., $\langle H_i \rangle$ is the h-vector of the cographic matroid of G. What this states algebraically is that there is a 1–1 correspondence between the spanning trees of G and the elements of a vector space basis for $K[\mathrm{Cog}_G]/\langle \hat{\Theta} \rangle$.

Now, with the help of Theorem 4.3, we can demonstrate a monomial basis. We state the theorem for $K = \mathbb{Z}_2$, but with some minor modifications, the proof holds for any field K.

THEOREM 5.1. *Let G be a connected loopless graph, Δ the cographic matroid of G, and Θ the h.s.o.p. of degree 1 for $\mathbb{Z}_2[Cog_G]$ defined in Theorem 4.3. Then the set of monomials*

$$\left\{ \prod EP(T) : T \text{ is a spanning tree of } G \right\}$$

is a basis for $\mathbb{Z}_2[Cog_G]/\langle \Theta \rangle$.

Proof. It is well known (see [2, 23]) that the subspace generated by Θ is the *circuit subspace*, and contains $\sum C$ for any circuit C of G. It follows that if H is an Eulerian spanning subgraph of G (i.e., every vertex of G has even degree in H), then $\sum H$ also belongs to the subspace generated by Θ. Thus if Euler denotes the ideal generated by $\{\sum H : H \text{ is an Eulerian subgraph of } G\}$, then the ideals Euler and $\langle \Theta \rangle$ of $\mathbb{Z}_2[\mathrm{Cog}_G]$ are identical. Hence

$$\mathbb{Z}_2[\mathrm{Cog}_G]/\langle \Theta \rangle = \mathbb{Z}_2[\mathrm{Cog}_G]/\mathrm{Euler}.$$

Since we know that the total number of monomials in $\{\prod EP(T) : T \text{ is a spanning tree of } G\}$ is at most the total number of spanning trees of G, which is $\sum H_i$, the (vector space) dimension of $\mathbb{Z}_2[\mathrm{Cog}_G]/\langle \Theta \cup \mathrm{Euler} \rangle$, we need only show that $\{\prod EP(T) : T \text{ is a spanning tree of } G\}$ spans $\mathbb{Z}_2[\underline{e}]$ (modulo $(\mathrm{Circ} + \mathrm{Euler})$)(here Circ is an abbreviation for the ideal $\mathrm{Circ}[\mathrm{Cog}_G]$).

We start by showing the following.

LEMMA 5.2.

$$\left\{ \prod S : S \subseteq E \right\}$$

spans $\mathbb{Z}_2[\underline{e}]$ (modulo $(Circ + Euler)$).

Proof. Let $f : E \to \mathbb{N}$; we denote by $\underline{e}^f$ the monomial $\prod_{e \in E} e^{f(e)}$ (so that for $S \subseteq X$, $\prod S$ is simply $\underline{e}^f$, where f is the characteristic vector of S). The *support* of f is the set

$$\mathrm{supp}(f) = \{e \in E : f(e) > 0\}.$$

We shall show that $\underline{e}^f$ is in the span of $\{\prod S : S \subseteq E\}$ by reverse induction on $|\mathrm{supp}(f)|$. If $|\mathrm{supp}(f)| > d$, then $G - \mathrm{supp}(f)$ is not connected, so $\underline{e}^f \in \mathrm{Circ}$, and

therefore $\underline{e}^f = 0$ (modulo (Circ + Euler)). For the inductive step, we can assume that $G - \text{supp}(f)$ is connected. If f is $\{0,1\}$-valued (i.e., $\underline{e}^f \in \{\prod S : S \subseteq E\}$) then we are done, so we can also assume that for some $e \in E$, $f(e) \geq 2$. Let P be a path in $G - \text{supp}(f)$ with the same ends as e. Then

$$e = \sum P(\text{modulo Euler})$$

and so

$$\underline{e}^f = \sum_{p \in P} \underline{e}^f p e^{-1} (\text{modulo Euler}).$$

Each monomial on the right-hand side has larger support than f, so by induction we are done. □

It remains to show that for all $S \subseteq E$, $\prod S$ is a $\mathbb{Z}_2$-linear combination of $\{\prod EP(T) : T \text{ is a spanning tree of } G\}$ (modulo (Circ + Euler)). Again, if $G - S$ is disconnected, $\prod S \in \text{Circ}$, and so $\prod S = 0$ (modulo (Circ + Euler)). Hence we assume that $G - S$ is connected. Let T be a spanning tree of G with $S \cap T = \emptyset$. If $S = EP(T)$, then we are done; otherwise, we make some local changes to "improve" the pair (S,T) (see also [15]).

LEMMA 5.3. *Let T be a spanning tree of G with $EP(T) \neq \emptyset$. Let $e \in EP(T)$ and let $e' = \min_< \mathcal{C}(T,e)$, where $\mathcal{C}(T,e)$ is the unique cycle in $T+e$. Set $T' = T - e' + e$. Then $EP(T') \subseteq EP(T) - e$. Furthermore, if $e = \min_< EP(T)$, then $EP(T') = EP(T) - e$.*

Proof. Consider any edge $a \in EP(T')$. Since $\mathcal{C}(T',e') = \mathcal{C}(T,e)$, we have $e' = \min_< \mathcal{C}(T',e')$, so $e' \notin EP(T')$, and hence $a \neq e'$. If $e \notin \mathcal{C}(T',a)$, then $\mathcal{C}(T',a) = \mathcal{C}(T,a)$, so $a \in EP(T')$ implies $a \in EP(T) - e$. On the other hand, if $e \in \mathcal{C}(T',a)$, then $e' \in \mathcal{C}(T,a)$; hence, if $a \notin EP(T) - e$, then $a = \min_< \mathcal{C}(T,a)$ and $a < e' = \min_< \mathcal{C}(T,e)$, and so we must have $a = \min_< \mathcal{C}(T',a)$ since $\mathcal{C}(T',a)$ is a subset of the symmetric difference of $\mathcal{C}(T,a)$ and $\mathcal{C}(T,e)$. This contradicts $a \in EP(T')$, and thus again $a \in EP(T) - e$. It follows that $EP(T') \subseteq EP(T) - e$.

Now assume $e = \min_< EP(T)$. Let $a \in EP(T) - e$. Then $e < a$. If $e' \notin \mathcal{C}(T,a)$, then $\mathcal{C}(T',a) = \mathcal{C}(T,a)$, so $a \in EP(T')$ as above. If $e' \in \mathcal{C}(T,a)$, then $e \in \mathcal{C}(T',a)$, so $a \in EP(T')$ in this case as well. Hence $EP(T) - e \subseteq EP(T')$, completing the proof of this lemma. □

Returning now to the proof of Theorem 5.1, let Ω_i denote the set of pairs (S,T) where $S \subseteq E$ has cardinality i and T is a spanning tree of G disjoint to S. For $e \in E$ let $l(e) \equiv |\{a \in E : a < e\}|$, and for $S \subseteq E$, let $l(S) = \sum_{e \in S} l(e)$. We define a partial order $\lhd$ on Ω_i by setting $(S,T) \lhd (S',T')$ if and only if either $l(S) < l(S')$, or $l(S) = l(S')$ and $EP(T') - S' \subset EP(T) - S$.

Assume that (S,T) is maximal in $(\Omega_i, \trianglelefteq)$. Then we claim first that $S - EP(T) = \emptyset$. If not, then let $e \in S - EP(T)$. Then $e = \min_< \mathcal{C}(T,e)$. Pick $e' \in \mathcal{C}(T,e) - e$ (we can do so as G is loopless). Then $S' = S - e + e'$ has $l(S') > l(S)$, and thus $(S,T) \lhd (S', T - e' + e)$, contradicting the maximality of (S,T).

Furthermore, the maximality of (S,T) implies that $EP(T) - S = \emptyset$ as well. For if not, then let $e \in EP(T) - S$ and $e' = \min_< \mathcal{C}(T,e)$. Then by Lemma 5.3, $(S,T) \lhd (S,T')$, where $T' = T - e' + e$, which again contradicts the maximality of (S,T).

It follows that the maximality of (S,T) implies that $S = EP(T)$, and clearly in this case $\prod S = \prod EP(T)$ is a $\mathbb{Z}_2$-linear combination of $\{\prod EP(T) : T \text{ is a spanning tree of } G\}$ (modulo (Circ+Euler)), which is what we want to show. We finally describe the local improvements if (S,T) is not maximal.

First, if $EP(T) - S \neq \emptyset$, then let $e \in EP(T) - S$, let $e' = \min_{<} \mathcal{C}(T, e)$, and let $T' = T - e' + e$. Then, as above, $(S, T) \triangleleft (S, T')$, and by downwards induction on Ω_i, $\prod S$ is in the span of $\{\prod EP(T) : T \text{ is a spanning tree of } G\}$ (modulo (Circ + Euler)).

Secondly, if $S - EP(T) \neq \emptyset$, then let $e \in S - EP(T)$. Let $\mathcal{C}(T, e) = \{e, e_1, \ldots, e_k\}$, so $e < e_i$ for all $j \in \{1, \ldots, k\}$. For each such j, let $S_j = S - e + e_j$ and $T_j = T - e_j + e$. Then $(S, T) \triangleleft (S_j, T_j)$ for each j. Furthermore, $e = \sum e_i$ (modulo Euler) and so $\prod S = \prod S_1 + \cdots + \prod S_k$ (modulo Euler). By downwards induction on Ω_i, we see that each $\prod S_i$ is in the span of $\{\prod EP(T) : T \text{ is a spanning tree of } G\}$ (modulo (Circ + Euler)), and hence so is $\prod S$.

This completes the proof of Theorem 5.1. □

What is fascinating is that this basis for $\mathbb{Z}_2[\mathrm{Cog}_G]/\langle\Theta\rangle$ depends on *all* spanning trees of the graph, while the explicit h.s.o.p. Θ (and hence any basis) depends on only *one* spanning tree of G.

We remark finally that in fact Theorem 5.1 can also be deduced in another way. Kind and Kleinschmidt [17] proved that if a d-dimensional complex Δ has a shelling $F_1, \ldots, F_r$ and the associated interval partition $[G_1, F_1], \ldots, [G_r, F_r]$, then the monomials $\prod G_1, \ldots, \prod G_r$ form a basis for $K[\Delta]/\langle\Theta\rangle$, where Θ is any h.s.o.p. of degree 1 for $K[\Delta]$. Björner's explicit shelling of matroids [4, p. 236] yields an interval partition, which for cographic matroids has intervals that are of the form $[EP(T), E - T]$, where T is a spanning tree of G. We have included the proof above since it relies more heavily on the underlying graphical structure (in terms of internal activities, trees, and circuits).

6. Some examples with Gröbner bases. Unfortunately, the monomial basis produced in Theorem 5.1 is not an order set of monomials. In this section, we examine some consequences of our work and a different approach that produces a monomial basis that is an order set of monomials.

A *Gröbner basis* (GB) (cf. [14, 16]) for an ideal I of $K[x_1, \ldots, x_m]$ is a set of polynomials from $K[x_1, \ldots, x_m]$ that is useful for determining membership in the ideal I, as well as for many other algebraic properties. The best known algorithm for producing a Gröbner basis is due to Buchberger [9], and we will briefly outline the algorithm (see [16]).

First we need some definitions. Suppose we have a given term ordering $<$ on $\mathrm{Mon}(x_1, \ldots, x_m)$ and set $Q \subseteq I - \{0\}$. The *head term* of a polynomial p, $\mathrm{ht}(p)$, is the largest monomial (under the given term ordering $<$) that appears with a nonzero coefficient; the *head coefficient*, $\mathrm{hcoeff}(p)$, is the coefficient of $\mathrm{ht}(p)$. For a set of polynomials $Q \subset K[x_1, \ldots, x_m] - \{0\}$, $\mathrm{ht}(\mathrm{GB})$ denotes the set of head terms of all polynomials in S. Let p be any nonzero polynomial. Suppose there exists a polynomial $q \in Q$ such that $\mathrm{ht}(q)$ divides a monomial in p, i.e., $p = km\mathrm{ht}(q) + r$ where $k \in K$, m is a monomial, and r is a polynomial. Then we write

$$p \mapsto_Q p - kmq$$

and say that p is *reducible modulo* Q (otherwise, p is *reduced modulo* Q; we also define the zero polynomial to be reduced as well). If $p \in I$, then the polynomial on the right obviously belongs to the ideal I as well, but has smaller head term. We let $\mapsto^+$ denote the reflexive, transitive closure of $\mapsto_Q$; that is, $p \mapsto_Q^+ r$ if and only if there are polynomials $p_1, \ldots, p_l$ for some l such that

$$p \mapsto_Q q_1 \mapsto_Q q_2 \mapsto_Q \cdots \mapsto_Q q_l \mapsto r.$$

We write $p \mapsto^*_Q r$ if $p \mapsto^+_Q r$ and r is reduced modulo Q. A set $\mathrm{GB} \subseteq I$ is a *Gröbner basis* for I if for every polynomial $p \in K[x_1, \ldots, x_m]$,

$$p \in I \qquad \text{if and only if} \qquad p \mapsto^*_{\mathrm{GB}} 0.$$

Buchberger's algorithm for constructing a Gröbner basis for an ideal I of $K[x_1, \ldots, x_m]$ can be stated as follows (cf. [16]). The *S-polynomial* of polynomials p and q is

$$\mathrm{Spoly}(p, q) = \mathrm{LCM}(\mathrm{ht}(p), \mathrm{ht}(q)) \left(\frac{p}{\mathrm{hcoeff}(p) \cdot \mathrm{ht}(p)} - \frac{q}{\mathrm{hcoeff}(q) \cdot \mathrm{ht}(q)} \right),$$

where LCM denotes the least common multiple (of two monomials). Buchberger [10] proved that a subset GB of ideal I is a Gröbner basis for I if and only if $\mathrm{Spoly}(p, q) \mapsto^+_{\mathrm{GB}} 0$ for all p, $q \in \mathrm{GB}$. To calculate a Gröbner basis for ideal I, take any generating set $Q = \{p_1, \ldots, p_k\}$ for the ideal. We take pairs of elements from Q, and reduce their S-polynomial with respect to Q. If the result is 0, we ignore it, but otherwise we add it into Q, and repeat the process. We stop when the S-polynomial of any two elements of Q reduces to 0 modulo Q (that such a process does terminate is the basis of Buchberger's algorithm). The final set Q is a Gröbner basis for Q.

The following observations hold for Buchberger's algorithm applied to a finitely generated ideal $I = \langle F = \{p_1, \ldots, p_k\} \rangle$ of $K[x_1, \ldots, x_m]$ (see [16] for a proof of the last property):

(i) If $Q' \subseteq Q$ and $p \mapsto^+_{Q'} q$ then $p \mapsto^+_Q q$.

(ii) If p and q are polynomials with $q | p$, then $p \mapsto_q 0$.

(iii) If each p_i is homogeneous, then for any homogeneous polynomial p of degree $l \geq 1$, either p reduces to 0 modulo F, or all reductions of p modulo F are homogeneous of degree l.

(iv) If p and q are homogeneous, then their S-polynomial is either 0 or also homogeneous (of degree $\mathrm{LCM}(\mathrm{ht}(p), \mathrm{ht}(q)) \geq \max(\mathrm{degree}(p), \mathrm{degree}(q))$).

(v) If p and q are monomials, then their S-polynomial is 0.

(vi) If the head terms of p and q are relatively prime (i.e., $\mathrm{LCM}(\mathrm{ht}(p), \mathrm{ht}(q)) = \mathrm{ht}(p) \cdot \mathrm{ht}(q)$) then $\mathrm{Spoly}(p, q) \mapsto^*_{\{p,q\}} 0$.

It follows that in Buchberger's algorithm, we need only consider the S-polynomial pairs of elements whose head terms are not relatively prime.

Another basic fact about Gröbner bases (see [16, p. 452]) is that if GB is a Gröbner basis for ideal I of $K[x_1, \ldots, x_m]$, then a basis for the quotient $K[x_1, \ldots, x_m]/I$ is the set

$$\{m \in \mathrm{Mon}(x_1, \ldots, x_m) : (\forall m' \in \mathrm{ht}(\mathrm{GB}))(m' \nmid m)\}.$$

Thus we can derive a basis of monomials for $K[x_1, \ldots, x_m]/I$ by taking the order set of monomials $-\mathrm{ht}(\mathrm{GB})$ on $x_1, \ldots, x_m$. The head terms of a Gröbner basis yield a set of *choppers* for an order set of monomial that is a basis for $K[x_1, \ldots, x_m]/I$.

Therefore, instead of using Stanley's greedy algorithm, we can calculate Gröbner bases (over any field of our choice) to find monomial bases for each of the homogeneous pieces R'_i's (and if we only mod out by a subset of J, we get upper bounds for the dimensions of the corresponding summands). Again, to simplify the arithmetic, we work over $\mathbb{Z}_2$.

When we talk subsequently about a Gröbner basis for a graph G, we are referring to a Gröbner basis for the quotient $\mathbb{Z}_2[\mathrm{Cog}_G]/\langle \Theta \rangle$, where Θ is the h.s.o.p. found in

§2. In the same vein, an order set of monomials for G is one such that for each i, the number of monomials of degree i is equal to the ith term in the h-vector of the cographic matroid of G. We point out that once we use Buchberger's Gröbner basis algorithm, the order set of monomials constructed appears to have no natural graph-theoretic description.

First we note some observations which follow from the previous sections. Let T be any spanning tree T of G. Suppose first that e is a loop, and let $\Theta(G)$ and $\Theta(G-e)$ denote the h.s.o.p. of G and $G-e$ determined by the tree T. Clearly, $\Theta(G) = \Theta(G-e) \cup \{\{e\}\}$, and G and $G-e$ have the same minimal cutsets. It follows that

$$\mathbb{Z}_2[\mathrm{Cog}_G]/\langle\Theta(G)\rangle \cong \mathbb{Z}_2[\mathrm{Cog}_G]/\langle\Theta(G-e)\rangle,$$

and hence G and $G-e$ have isomorphic order sets of monomials (and of course identical h-vectors).

Suppose now that e is a cut edge of G. Then the minimal cutsets of G are precisely those of $G \cdot e$ together with $\{e\}$. It is easy to verify that G and $G \cdot e$ have the same h.s.o.p. with respect to T. Again it follows that

$$\mathbb{Z}_2[\mathrm{Cog}_G]/\langle\Theta(G)\rangle \cong \mathbb{Z}_2[\mathrm{Cog}_{G\cdot e}]/\langle\Theta(G\cdot e)\rangle,$$

and hence G and $G \cdot e$ have isomorphic order sets of monomials (and identical h-vectors).

PROPOSITION 6.1. *Suppose that G has blocks $G_1, \ldots, G_k$, and a Gröbner basis for the respective rings are $\mathrm{GB}_1, \ldots, \mathrm{GB}_k$ (each of which contains the h.s.o.p. of degree 1 defined earlier and the minimal cutsets as products). Then $\cup_{i=1}^{k} GB_i$ is a Gröbner basis for G. Consequently, one can find an order set of monomials for G by taking the product of order set of monomials for each of $G_1, \ldots, G_k$.*

Proof. This follows from the facts that the union of spanning trees in each G_i yields a spanning tree for G, the minimal cutsets of G are the union of those from each G_i, and Buchberger's algorithm always yields zero reductions when the head terms are relatively prime. The second part is derived from the fact that the set of choppers for G is the union of the set of choppers for each G_i, which are on disjoint sets of variables. □

Now let's turn to a particular class of graphs. For $k \geq 1$, $l_1 \geq 1$, and $l_i \geq 0$ for all $i \in \{2, \ldots, k\}$, let $G(l_1, \ldots, l_k)$ denote the graph consisting of two vertices x and y (called the *terminals*) joined by internally disjoint paths $P_1, \ldots, P_k$ of lengths $l_1, l_2 + 1, \ldots, l_k + 1$, respectively (the length of a path is the number of edges it contains). For $k = 1$, we simply have a path, and for $k = 2$, a cycle. When $k = 3$, we have what is usually called a θ-graph. Clearly, $G(l_1, \ldots, l_k)$ has order $n = (\sum_{i=1}^{k} l_i)+1$, size $m = (\sum_{i=1}^{k} l_i + k - 1$ and dimension $d = k - 1$. We shall produce a Gröbner basis for $G(l_1, \ldots, l_k)$.

PROPOSITION 6.2. *Let $G = G(l_1, \ldots, l_k)$ ($k \geq 2$) have its edges labelled as in Fig. 1. For $i = 1, \ldots, k$, let*

$$l_i' = \begin{cases} l_1 & \text{if } i = 1, \\ l_i + 1 & \text{if } i = 2, \ldots, k. \end{cases}$$

Then a Gröbner basis for G is

$$\{\theta_i = \sum P_1 + \sum P_i : i = 2, \ldots, k\}$$

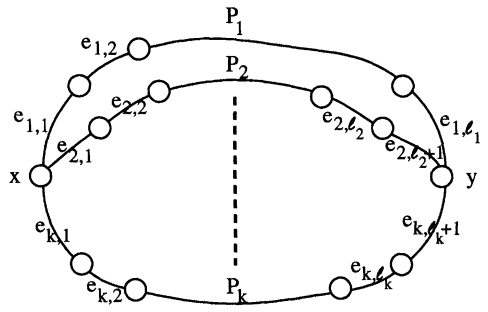


FIG. 1.

$$\cup \{e_{q,i}e_{q,j} : q = 1, \ldots, k,\ 1 \le i < j \le l'_q\}$$
$$\cup \left\{\prod_{q=1}^{k} e_{q,i} : q = 1, \ldots, k,\ 1 \le i \le l'_q\right\}$$
$$\cup \{e_{q,i}^2 + e_{q,i} \cdot \sum P_1 : q = 2, \ldots, k,\ 1 \le i \le l_k\}$$

Proof. Let the edge ordering be $e_{i,j} < e_{i',j'}$ iff $i < i'$, or $i = i'$ and $j < j'$ (i.e., the lexicographic ordering on $\{e_{i,j}\}$), and extend this to any term ordering on the monomials in these variables. Take the spanning tree $T = \{e_{i,j} : i = 1 \text{ or } i \ge 2 \text{ and } j \le l_i\}$.

Now the corresponding h.s.o.p. is $\theta_2, \ldots, \theta_k$, where

$$\theta_i = \sum P_i + \sum P_1.$$

Let

$$\alpha_{r,i,j} = e_{r,i}e_{r,j}$$

for $r = 1$, $1 \le i < j \le l_1$ or $r = 2, \ldots, k$, $1 \le i < j \le l_r + 1$. Let

$$\beta_{i_1,\ldots,i_k} = \prod_{j=1}^{k} e_{j,i_j},$$

where $1 \le i_j \le l'_j$. That is, the sets of α's and β's correspond to the minimal cutsets of G (which are either two edges from some P_i or one edge from each P_i).

Thus a Gröbner basis for G can be obtained by running Buchberger's algorithm on

$$\theta_2, \ldots, \theta_k, \alpha_{1,1,2}, \ldots, \alpha_{k,l'_k-1,l'_k}, \beta_{1,1,\ldots,1}, \ldots, \beta_{l'_1,l'_2,\ldots,l'_k}.$$

Throughout, we do not consider S-polynomials with zero reductions. Recall that zero reductions result from the S-polynomial of any two polynomials whose head terms are relatively disjoint, and for any monomials; also, if a polynomial is a multiple of a member of the Gröbner basis, then it also reduces down to 0. The head term of any monomial is obviously itself, and the head term of θ_i is e_{i,l_i+1}. We will need one additional observation about $\mapsto$:

- If p is any polynomial and θ is homogeneous of degree 1, then

$$p \mapsto_{\{\theta\}} \text{subs}(\text{ht}(\theta) \leftarrow \theta - \text{ht}(\theta), p),$$

where $\text{subs}(x \leftarrow y, p)$ denotes the polynomial formed by substituting y for all occurrences of x in p.

For $r \geq 2$, we have

$$S(\alpha_{r,i,l_r+1}, \theta_r) \mapsto^* \gamma_{r,i} = e_{r,i}^2 + e_{r,i} \cdot (\sum P_1),$$

and these are the only S-polynomials to consider among the θ's and α's. We tack these onto the end of our list.

For the β's, note that if for exactly one j, $i_j = l_j + 1$, then

$$S(\beta_{i_1,\ldots,i_k}, \theta_j) \mapsto^* e_{1,i_1}^2 \prod_{r \neq j} e_{r,i_r},$$

and hence we get all monomials of degree k of the form

$$e_{1,i1}^2 \prod_{r \in J_1} e_{r,i_r},$$

where $J_1 \subseteq \{2, \ldots, k\}$ is of cardinality k-2, and $i_q \leq l_q$. If exactly two j's, say j_1 and j_2, satisfy $i_j = l_j + 1$, then in light of the previous monomials, for $j = j_1$ or j_2,

$$S(\beta_{i_1,\ldots,i_k}, \theta_j) \mapsto^* e_{1,i_1}^3 \prod_{r \neq j_1, j_2} e_{r,i_r},$$

and hence we get all monomials of degree k of the form

$$e_{1,i1}^3 \prod_{r \in J_3} e_{r,i_r},$$

where $J_2 \subseteq \{3, \ldots, k\}$ is of cardinality $k-3$, and $i_q \leq l_q$. Continuing in this manner, we add to our list all monomials of degree k of the form

$$\delta(j, i_1, m) = e_{1,i_1}^j \cdot m,$$

where $j \in \{2, \ldots, k\}$ and m is a square-free monomial formed by taking the product of edges, one taken from each of $k-j$ paths $P_2 - e_{2,l_2'}, \ldots, P_k - e_{k,l_k'}$. These complete the only S-polynomials to consider among the initial sequence:

$$\theta_2, \ldots, \theta_k, \alpha_{1,1,2}, \ldots, \alpha_{k,l_k,l_{k+1}}, \beta_{1,1,\ldots,1}, \ldots, \beta_{l_1,l_2+1,\ldots,l_k+1}.$$

We need only consider S-polynomials involving a $\gamma_{r,i}$ and either an α_{r,j_1,j_2} with $j_1 = i$ or $j_2 = i$, a $\beta_{i_1,\ldots,i_k}$ with $i_r = i$, or a $\delta(j, i_1, m)$ with $e_{r,i}|m$. In the first case, if (without loss) $j_1 = i$, a simple calculation shows that $\text{Spoly}(\gamma_{r,i}, \alpha_{r,j_1,j_2})$ is a multiple of an α_{j_1,j_2}. In the second case, $\text{Spoly}(\gamma_{r,i}, \beta_{i_1,\ldots,i_k})$ is a multiple $\beta_{i_1,\ldots,i_k}$. Similarly, in the third case $\text{Spoly}(\gamma_{r,i}, \delta(j, i_1, m))$ is a multiple of $\delta(j, i_1, m)$. In any case, the S-polynomial reduces down to 0, so Buchberger's algorithm terminates with the sequence

$$\theta_2, \ldots, \theta_k, \alpha_{1,1,2}, \ldots, \alpha_{k,l_k,l_{k+1}}, \beta_{1,1,\ldots,1}, \ldots, \beta_{l_1,l_2+1,\ldots,l_k+1},$$
$$\gamma_{2,1}, \ldots, \gamma_{k,l_k}, \delta(2, 1, e_{2,1} \cdots e_{k,1}), \ldots, \delta(k, 1, e_{l_1,1}) = e_{l_1}^k$$

being a Gröbner basis. □

As a corollary, we derive explicit formulation for an order set of monomials for $G = G(l_1, \ldots, l_k)$.

COROLLARY 6.3. *A set of choppers for an order set of monomials for* $G(l_1, \ldots, l_k)$ *on variables* $\{e_{i,j} : 1 \le i \le k,\ 1 \le j \le l_i\}$ *is*

$$\begin{aligned}&\{e_{i,j}^2 : 2 \le i \le k,\ 1 \le j \le l_i\}\\ &\quad \cup \{e_{i,j}e_{i,j'} : 1 \le i \le k,\ 1 \le j < j' \le l_i\}\\ &\quad \cup \left\{ e_{1,i}^j \prod_{q \in S} e_{q,j} : 2 \le j \le k,\ 1 \le i \le l_1,\ |S| = k - j,\ 1 \le j \le l_q \right\}.\end{aligned}$$

Also, set $N = \sum_{i=1}^{k} l_i$, *and for* $1 \le i \le k-1$ *let* $g_i = \sum l_{j_1} l_{j_2} \cdot l_{j_i}$, *where the sum is taken over all* $2 \le j_1 < j_2 < \cdots < j_i \le k$. *Then the h-vector* $\langle H_0, \ldots, H_{k-1} \rangle$ *satisfies*

$$\begin{aligned}&H_0 = 1,\\ &H_1 = N,\\ &H_i = g_i + l_1(g_{i-2} + g_{i-3} + \cdots + g_1 + 1) \qquad \text{for } i = 2, \ldots, k-1.\end{aligned}$$ □

7. Conclusion. One of the least understood aspects of the use of the Ball–Provan bounds for network reliability is an explicit construction of the order set of monomials for a connected graph G, and what properties hold for such a multicomplex. Corollary 6.3 shows that an order set of monomials for $G(l_1, \ldots, l_k)$ on variables $\{e_{i,j} : 1 \le i \le k,\ 1 \le j \le l_i\}$ consists of all monomials of the form

$$e_{1,i_1}^j \prod_{q \in S} e_{q,i},$$

where $1 \le j < k$, $1 \le i_1 \le l_1$, $S \subseteq \{2, \ldots, k\}$, $|S| < k - j$, and $1 \le i \le l_q$. This order set of monomials is *pure*, i.e., all the maximal monomials (under division) have the same degree. It is conjectured that every graph has an associated pure order set of monomials [20], and the truth of this conjecture would have implications to improving the Ball–Provan bounds for network reliability.

A further understanding of the associated order sets of monomials could have strong bounding implications in the following sense as well. The Ball–Provan bounds for reliability use inequalities [1] that are satisfied by degree sequences of order sets of monomials, and hence h-vectors of shellable complexes. The sharper set of Clements–Lindström inequalities [11] could be used to strengthen the Ball–Provan bounds, provided the maximum degree of each variable can be bounded in the order set of monomials produced in §6. Unfortunately, we only have partial results in this direction. Along these lines, we have been able to prove that if G is a connected graph of order n with an independent set $I = \{v_1, \ldots, v_k\}$ such that $G - I$ is still connected, then there exists for G an order set of monomials M on $n-1$ variables such that for some k of the variables $e_1, \ldots, e_k$, the highest power of each e_i in any term of M is $e_i^{\deg(v_i)}$ (here $\deg(v_i)$ is the degree of vertex v_i in graph G). Much stronger results will be of significant interest.

REFERENCES

[1] M. O. Ball and J. S. Provan, *Bounds on the reliability polynomial for shellable independence systems*, SIAM J. Alg. Disc. Meth., 3 (1982), pp. 166–181.

[2] N. Biggs, *Algebraic Graph Theory*, Cambridge Univ. Press, Cambridge, UK, 1974.

[3] L. J. Billera, *Polyhedral theory and commutative algebra*, in Mathematical Programming: The State of the Art, Springer-Verlag, New York, 1983, pp. 57–77.

[4] A. Björner, *Homology and Shellability*, in Matroid Applications, Cambridge Univ. Press, Cambridge, UK, 1992.

[5] J. I. Brown and C. J. Colbourn, *On the log concavity of reliability and matroidal sequences*, Adv. Appl. Math., 15 (1994), pp. 114–127.

[6] ———, *Roots of the reliability polynomial*, SIAM J. Discrete Math., 5 (1992), pp. 571–585.

[7] J. I. Brown, C. J. Colbourn, and J. S. Devitt, *Network transformations and bounding network reliability*, Networks, 23 (1993), pp. 1–17.

[8] J. I. Brown and C. J. Colbourn, *Non-Stanley bounds for network reliability*, J. Algebraic Combin., 5 (1996), pp. 13–36.

[9] B. Buchberger, *An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal*, Ph.D. Thesis, Univ. of Innsbruck, Austria, 1965.

[10] ———, *A theoretical basis for the reduction of polynomials to canonical forms*, ACM SIGSAM Bull., 10 (1976), pp. 19–29.

[11] G. F. Clements and B. Lindström, *A generalization of a combinatorial theorem of Macaulay*, J. Combin. Theory, 7 (1969), pp. 230–238.

[12] C. J. Colbourn, *The combinatorics of network reliability*, Oxford Univ. Press, New York, 1987.

[13] C. J. Colbourn and W. R. Pulleyblank, *Matroid Steiner problems, the Tutte polynomial and network reliability*, J. Combin. Theory Ser. B, 47 (1989), pp. 20–31.

[14] D. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms*, Springer-Verlag, New York, 1992.

[15] J. E. Dawson, *A Collection of Sets Related to the Tutte Polynomial of a Matroid*, in Lecture Notes in Mathematics 1073, 1984, pp. 193–204.

[16] K. O. Geddes, S. R. Czapor, and G. Labahn, *Algorithms for Computer Algebra*, Kulwer, Boston, 1992.

[17] B. Kind and P. Kleinschmidt, *Schäbare Cohen–Macauley Komplexe und ihre Parametrisierung*, Math. Z., 167 (1979), pp. 173–179.

[18] F. S. Macaulay, *Some properties of enumeration in the theory of modular systems*, J. Lond. Math. Soc., 26 (1927), pp. 531–555.

[19] R. P. Stanley, *The upper bound conjecture and Cohen–Macaulay rings*, Stud. Appl. Math., 54 (1975), pp. 135–142.

[20] ———, *Cohen–Macaulay complexes*, in Higher Combinatorics, Reidel, Boston, 1977, pp. 51–64.

[21] ———, *Hilbert Functions of Graded Algebra*, Adv. Math., 28 (1978), pp. 57–83.

[22] ———, *Balanced Cohen–Macaulay Complexes*, Trans. Amer. Math. Soc., 249 (1979), pp. 139–157; Reidel, Boston, 1977, pp. 51–64.

[23] K. Thulasiraman and M. N. S. Swamy, *Graphs: Theory and Algorithms*, John Wiley, New York, 1992.

[24] D. J. A. Welsh, *Matroid Theory*, Academic Press, London, 1976.

SIAM J. DISCRETE MATH.
Vol. 9, No. 3, pp. 393–412, August 1996

005

TILINGS OF BINARY SPACES*

GERARD COHEN†, SIMON LITSYN‡, ALEXANDER VARDY§, AND GILLES ZÉMOR¶

Abstract. We study partitions of the space $\mathbb{F}_2^n$ of all the binary n-tuples into disjoint sets, where each set is an additive coset of a given set V. Such a partition is called a tiling of $\mathbb{F}_2^n$ and denoted (V, A), where A is the set of coset representatives. We give a sufficient condition for a set V to be a tile in terms of the cardinality of $V+V$. We then employ this condition to classify all tilings with sets of small cardinality. Further, periodicity of tilings in $\mathbb{F}_2^n$ is discussed, and a simple construction of nonperiodic tilings of $\mathbb{F}_2^n$ is presented for all $n \geq 6$. It is also shown that the nonperiodic tiling of $\mathbb{F}_2^6$ is unique. A tiling (V, A) is said to be proper if V generates $\mathbb{F}_2^n$; it is said to be full rank if both V and A generate $\mathbb{F}_2^n$. We show that, in general, the classification of tilings can be reduced to the study of proper tilings. We then prove that any tiling may be decomposed into smaller tilings that are either trivial or have full rank. Existence of full-rank tilings is exhibited by showing that each tiling is uniquely associated with a perfect binary code. Moreover, it is shown that periodic full-rank tilings may be further decomposed into smaller tilings, and then the existence of nonperiodic full-rank tilings is deduced. Finally, we generalize the well-known Lloyd theorem, originally stated for tilings by spheres, for the case of arbitrary tilings.

Key words. tiling, tessallation, Hamming space, hypercube, factorization, finite graphs, perfect codes

AMS subject classifications. 05B45, 05A18, 94B25, 94B60, 52C25

1. Introduction.

Given a body in an n-dimensional metric space, is it possible to tile the space with translations of this body? This problem has been extensively studied for the Euclidean space $\mathbb{R}^n$; see [29, 32] and references therein. The case of Euclidean spaces of small dimension has received considerable attention, and substantial progress was achieved for tilings of the Euclidean plane [11, 9]. In particular, Penrose [24] demonstrated the existence of nonperiodic tilings of $\mathbb{R}^2$. Another related topic is tilings of the plane with a finite collection of specially defined bodies. See, for example, publications on polyomino tilings and related problems [12].

In another setting, one may define tilings of finite abelian groups G, where by a tiling we mean a decomposition of the form $G = A + V$ such that $|V| \cdot |A| = |G|$. In other words, tiling is a partition of G into additive cosets of a tile set V. The problem of describing all possible tilings of abelian groups was first brought up by Hajós [13] and is largely unsolved. Hajós suggested a first effort in this direction by considering tilings of a certain type, which will be called *periodic* in this paper. For a precise definition of such tilings, see §5. If all the tilings of a given group G are periodic, then they can be described accurately with the help of a recursive decomposition. After a substantial amount of effort, the problem of determining all the abelian groups that admit only periodic tilings was finally solved by Sands [30], following ingenious ideas of de Bruijn [3] and Rédei [28].

* Received by the editors January 18, 1995; accepted for publication (in revised form) August 15, 1995.

† Departément Informatique, École Nationale Superieure des Télécommunications, 46 rue Barrault, 75634 Paris, France.

‡ Department of Electrical Engineering, Tel-Aviv University, Ramat-Aviv 69978, Israel.

§ Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801. The research of this author was supported in part by the National Science Foundation and in part by JSEP grant N00014-9610129.

¶ Departément Reseaux, École Nationale Superieure des Télécommunications, 46 rue Barrault, 75634 Paris, France.

In this work we are interested in tiling the Hamming space $\mathbb{F}_2^n$ consisting of all binary n-tuples. This problem falls into both of the above-mentioned categories of tilings, since $\mathbb{F}_2^n$ is at the same time a metric space and a finite abelian group. We will show that other features of tilings besides periodicity, and in particular their rank, may be employed for the recursive decomposition of tilings. More generally, we shall be interested in describing those subsets of $\mathbb{F}_2^n$ that are tiles.

Results on the problem of tiling $\mathbb{F}_2^n$ have applications in coding theory. Thus, certain particular cases of this problem have already been studied in the coding theory literature [20, 22]. For example, tilings of $\mathbb{F}_2^n$ with Hamming spheres are known as perfect binary codes [19, 20]. Although the parameters of all such tilings have been determined in [19, 34], a complete classification remains an open problem. Another example is tilings of $\mathbb{F}_2^n$ by the so-called L-spheres, which are unions of some of the shells of the Hamming sphere. For this case all possible parameters are also known [4, 15], whenever $L \subset \{0, 1, \ldots, n/2\}$.

In general, tilings of $\mathbb{F}_2^n$ with arbitrary bodies correspond to perfect codes correcting an arbitrary set of errors. Conditions for the existence of codes correcting an arbitrary set of errors and estimates of their cardinality were investigated in [6, 7, 8, 16, 17]. Such codes are useful, for instance, for correcting errors at the output of logic networks—the set of errors depends on the structure of the network and a single error in an element of the network may lead to an error of greater multiplicity at the network output [8]. Other examples where the problem of correcting a given set of errors arises are concatenated coding schemes [33] and "artificial noise" channels, which occur if the process of data transmission is considered as a game situation [6]. Tilings of $\mathbb{F}_2^n$ were also found useful in [35] for the design of soft-decision decoders for BCH codes.

This paper is organized as follows. We start with some notation and definitions in §2. In §3 we present a sufficient condition for a given set V to be a tile in terms of the cardinality of the difference set of V. In §4 we provide a complete classification of tiles of cardinality ≤ 8. Tilings with sets of large rank are also considered in §4. In §5 we define periodicity of tilings in $\mathbb{F}_2^n$, which is analogous to periodicity of tilings in the Euclidean plane. We prove that nonperiodic tilings of $\mathbb{F}_2^n$ do not exist for $n \leq 5$ and that the nonperiodic tiling of $\mathbb{F}_2^6$ is unique up to coordinate transformations. An explicit construction of nonperiodic tilings of $\mathbb{F}_2^n$ for all $n \geq 6$ is also presented. In §6 we introduce the concept of proper tilings and show that the classification of tilings of $\mathbb{F}_2^n$ may be reduced to the study of proper tilings. This leads to a recursive decomposition of tilings into tilings of smaller and smaller size and ultimately shows that any tiling of $\mathbb{F}_2^n$ may be constructed from tilings that are either trivial or have full rank. In §7 we show that a tiling (V, A) is uniquely associated with a perfect binary code of length $|V| - 1$. This allows us to deduce the existence of full-rank tilings from the existence of full-rank perfect codes, established in [10]. A construction of tilings from perfect codes is also presented in §7. In §8 we describe a technique for further decomposing a periodic full-rank tiling into tilings of smaller size. This technique is then employed to show that nonperiodic full-rank tilings exist. In §9 we generalize the well-known Lloyd theorem [20, Chap. 7], originally stated for tilings by spheres, for the case of arbitrary tilings. We give examples where such generalization may be used to prove the nonexistence of certain tilings.

2. Preliminaries. Let $\mathbb{F}_2^n$ denote the vector space of dimension n over GF(2). The Hamming distance between vectors $x, y \in \mathbb{F}_2^n$, denoted $d(x, y)$, is the number of positions where x and y differ. The Hamming weight of x is $wt(x) = d(x, \mathbf{0})$,

where $\mathbf{0}$ denotes the all-zero vector. Given a subset $C \subset \mathbb{F}_2^n$ and $x \in \mathbb{F}_2^n$, we denote $d(x, C) = \min_{y \in C} d(x, y)$. If $|C| = M$ and $\min_{x \neq y \in C} d(x, y) = d$, we shall say that C is an (n, M, d) binary code. For any $x \in \mathbb{F}_2^n$ and a nonnegative integer $R \leq n$, the Hamming sphere of radius R about x is given by $\mathcal{B}_n(x, R) = \{ y \in \mathbb{F}_2^n : d(x, y) \leq R \}$. Given $x = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_2^n$, we denote by $\mathrm{supp}(x)$ the subset of $\{1, 2, \ldots, n\}$ consisting of all i such that $x_i \neq 0$. Given a subset $V \subset \mathbb{F}_2^n$ with $\mathbf{0} \in V$, we denote

$$iV \stackrel{\text{def}}{=} \underbrace{V + V + \cdots + V}_{i \text{ times}},$$

where $+$ stands for the direct sum. Thus iV is the set of all vectors in $\mathbb{F}_2^n$ that may be represented as a sum of some i elements of V. The linear span of V, denoted $\langle V \rangle$, is the subspace of $\mathbb{F}_2^n$ generated by V. The rank of V is given by $\mathrm{rank}(V) = \dim \langle V \rangle$. We let $\rho(V)$ denote the minimum number of elements of V required to generate any vector in its linear span; namely,

$$\rho(V) \stackrel{\text{def}}{=} \min_j \{ j : jV = \langle V \rangle \}.$$

It is easy to see that $\rho(V)$ is the covering radius of the linear code defined by a parity-check matrix $H(V)$, having the elements of $V \setminus \{\mathbf{0}\}$ as its columns.

We shall say that a given set V is a *tile* of $\mathbb{F}_2^n$ if it is possible to partition $\mathbb{F}_2^n$ into disjoint additive cosets of V. Note that the set of coset representatives A is also a tile of $\mathbb{F}_2^n$. Without loss of generality (w.l.o.g) we assume that both V and A contain the $\mathbf{0}$ element, unless stated otherwise. Evidently, each $x \in \mathbb{F}_2^n$ has a unique representation of the form $x = v + a$, where $v \in V$ and $a \in A$. Thus we have the following definition.

DEFINITION 2.1. *The pair (V, A) is a tiling of $\mathbb{F}_2^n$ if $V + A = \mathbb{F}_2^n$ and $2V \cap 2A = \{\mathbf{0}\}$.*

A trivial necessary condition for (V, A) to be a tiling of $\mathbb{F}_2^n$ is that $|V| = 2^k$ and $|A| = 2^{n-k}$ for some $0 \leq k \leq n$. Thus, hereafter, when denoting a subset of $\mathbb{F}_2^n$ by V, we assume that $|V| = 2^k$.

If both V and A are linear subspaces of $\mathbb{F}_2^n$, then (V, A) is a tiling if and only if $A = \mathbb{F}_2^n / V$. Hence, in what follows we mainly focus on those tilings where at least one of the sets V, A is not linear. However, we shall say that $V \subset \mathbb{F}_2^n$ is a *linear tile* if there is a tiling (V, A) such that A is a linear subspace of $\mathbb{F}_2^n$.

3. A sufficient condition. A well-known example of a linear tile is the Hamming sphere, in which case the set of coset representatives A is a perfect binary code. More precisely, a sphere of radius $R < (n-1)/2$ is a (linear) tile if and only if $R = 1$, $n = 2^m - 1$ or $R = 3$, $n = 23$ (cf. [22]). In this section we present a sufficient condition for a set V to be a linear tile, which shows that many more such tiles exist.

THEOREM 3.1. *If $|2V| < 2\,|V|$ then V is a linear tile.*

Proof. Let $|V| = 2^k$ and suppose that $|2V| < 2^{k+1}$. Then either $k = n$, in which case $V = \mathbb{F}_2^n$ is a trivial linear tile, or $|2V| < 2^n$ and there exists an $a_1 \in \mathbb{F}_2^n \setminus 2V$. Set $V_1 = V \cup (a_1 + V)$. Since $a_1 \notin 2V$, we have $V \cap (a_1 + V) = \emptyset$ and $|V_1| = 2\,|V| = 2^{k+1}$. If $V_1 = \mathbb{F}_2^n$ then $(V, \{\mathbf{0}, a_1\})$ is a tiling of $\mathbb{F}_2^n$ and we are done. Otherwise, since $|2V_1| = |2V \cup (a_1 + 2V)| < 2\,|V_1|$, there exists an $a_2 \in \mathbb{F}_2^n \setminus 2V_1$, and we set $V_2 = V_1 \cup (a_2 + V_1)$ with $V_1 \cap (a_2 + V_1) = \emptyset$ and $|V_2| = 2\,|V_1|$. Continuing in this manner we construct $V_{n-k} = \mathbb{F}_2^n$ and a tiling (V, A) of $\mathbb{F}_2^n$, where A is a subspace of $\mathbb{F}_2^n$ generated by $a_1, a_2, \ldots, a_{n-k}$. □

Using the results of Zémor [38], it is, in principle, possible to characterize all the sets V which satisfy the condition of Theorem 3.1. Here we give an example of construction that produces such a set. Fix a subspace $S \subset \mathbb{F}_2^n$ and a nonzero element $s \in S$. Partition S into cosets modulo the two-element subspace $\{\mathbf{0}, s\}$, and construct V by choosing one and only one element from each coset. By assumption, we need $\mathbf{0} \in V$ and therefore $s \notin V$. But then, by construction, s is not in $V + V$ and hence $|2V| < |S| = 2|V|$.

In the following sections we will be particularly interested in those tiles V which satisfy $\langle V \rangle = \mathbb{F}_2^n$. It can be shown that for these tiles $|2V| \geq \frac{7}{4}|V|$. More bounds on the cardinality of $V + V$ for a given set V may be found in [38].

The condition $|2V| < 2|V|$ of Theorem 3.1 may be somewhat relaxed if additional information pertaining to V is available. For instance, consider the following corollary.

COROLLARY 3.2. *If $|2V| = 2|V|$ then V is a tile if and only if $2V$ is not linear.*

Proof. ($\Leftarrow$) If $2V$ is not linear then $2V \neq 4V$ and there is an $a_1 \in 4V \setminus 2V$. Set $V_1 = V \cup (a_1 + V)$. Since $a_1 \notin 2V$, we have $V \cap (a_1 + V) = \emptyset$ and $|V_1| = 2|V|$ as before. Now $2V_1 = 2V \cup (a_1 + 2V)$, and since $a_1 \in 4V$ it follows that $2V \cap (a_1 + 2V) \neq \emptyset$. Thus $|2V_1| < 2|2V| = 2|V_1|$. Applying Theorem 3.1 to V_1, we conclude that there exists a subspace $A_1 \subset \mathbb{F}_2^n$ such that (V_1, A_1) is a tiling. But then so is $(V, A_1 + \{\mathbf{0}, a_1\})$. ($\Rightarrow$) If $2V$ is linear then $\langle V \rangle = 2V$, and therefore V does not tile its linear span. The claim now follows directly from Proposition 6.1. □

Note that $2V$ is not linear if and only if $\rho(V) > 2$. In fact, the argument of Corollary 3.2 may be pushed a little further, provided that $\rho(V)$ is large enough.

COROLLARY 3.3. *If any of the following hold, then V is a linear tile.*

(a) $|2V| \leq 2|V| + 2$ *and* $\rho(V) > 2$;
(b) $|2V| \leq 2|V| + 4$ *and* $\rho(V) > 5$;
(c) $|2V| \leq 2|V| + 5$ *and* $\rho(V) > 20$.

Proof. (a) If $\rho(V) > 2$ there exists an $a_1 \in 3V \setminus 2V$, say $a_1 = v_1 + v_2 + v_3$. As before let $V_1 = V \cup (a_1 + V)$ with $|V_1| = 2|V|$ and $2V_1 = 2V \cup (a_1 + 2V)$. Note that $2V \cap (a_1 + 2V)$ must contain the six vectors $v_1, v_2, v_3, v_1 + v_2, v_1 + v_3, v_2 + v_3$. Thus $|2V_1| \leq 2|2V| - 6 \leq 2|V_1| - 2$. Again, applying Theorem 3.1 to V_1, we conclude that V_1 is a linear tile. Hence so is V.

(b) A similar argument yields $|2V_1| \leq 2|V_1| + 2$ in this case. Since $\rho(V) > 5$, it follows that $2V_1 = 2V \cup (a_1 + 2V) \neq \langle V \rangle$ or, in other words, $\rho(V_1) > 2$. Hence, the proof of case (a) may be used to establish that V_1, and therefore also V, is a tile.

(c) Similarly, $|2V_1| \leq 2|V_1| + 4$, and since $\rho(V) > 20$ we have $\rho(V_1) > 5$. Thus, the proof of case (b) applies. □

4. Classification of small tiles. In this section we completely characterize all tiles of size ≤ 8. In addition, we also consider tilings with sets of large rank. First, we have the following simple proposition.

PROPOSITION 4.1. *If $|V| \leq 4$ then V is a tile.*

Proof. Notice that $|2V| \leq \binom{|V|}{2} + 1$, and for $|V| \leq 4$ we have $\binom{|V|}{2} + 1 < 2|V|$. Hence V is a linear tile by Theorem 3.1. □

Next we classify tiles of size 8. Let $V = \{\mathbf{0}, v_1, \ldots, v_7\}$. When is V a tile? The answer to this question is greatly facilitated by the following simple fact: V is a tile of $\mathbb{F}_2^n$ if and only if it is a tile of its own linear span $\langle V \rangle$. The proof of this statement is postponed until Proposition 6.1. Here, we distinguish between several cases according to the rank of V.

CLAIM 4.2. *If* $\operatorname{rank}(V) = 7$ *then V is a tile.*

Proof. Let C be the perfect binary Hamming code of length 7. Define

$$A = \{ c_1v_1 + c_2v_2 + \cdots + c_7v_7 : (c_1, c_2, \ldots, c_7) \in C \}.$$

We claim that (V, A) is a tiling of $\langle V \rangle$. Since $|V| = 2^3$, $|A| = 2^4$, and $|\langle V \rangle| = 2^7$, it would suffice to show that $2V \cap 2A = \{\mathbf{0}\}$. If $a_1 + a_2 \in 2V$ for some distinct $a_1, a_2 \in A$, then there are two distinct codewords in C at distance ≤ 2 from each other. However, since the minimum distance of C is 3, this is a contradiction. □

Note that the proof of Claim 4.2 amounts to choosing the appropriate coordinates for $\langle V \rangle$ so that the set V of rank 7 becomes the Hamming sphere $\mathcal{B}_7(\mathbf{0}, 1)$. In general, we may always assume w.l.o.g. that $\mathcal{B}_r(\mathbf{0}, 1) \subset V$, where $r = \text{rank}(V)$.

Consider the case where $\text{rank}(V) = 6$. After a suitable choice of coordinates for $\langle V \rangle$, we have $V = \mathcal{B}_6(\mathbf{0}, 1) \cup \{x\}$, where $x \in \mathbb{F}_2^6$ is of weight ≥ 2. In this case, V is a tile if and only if $wt(x) \neq 4, 5$. The proof of this follows from Proposition 4.5, and is therefore postponed.

Now let $\text{rank}(V) = 5$. Again, with a suitable choice of coordinates for $\langle V \rangle$, we have $V = \mathcal{B}_5(\mathbf{0}, 1) \cup \{x, y\}$.

CLAIM 4.3. *V is a tile if and only if one of the following holds:*

(a) $wt(x) = 2$, $wt(y) = 2$, *and* $wt(x + y) = 2$;
(b) $wt(x) = 3$, $wt(y) = 2$, *and* $wt(x + y) = 1$;
(c) $wt(x) = 3$, $wt(y) = 2$, *and* $wt(x + y) = 5$;
(d) $wt(x) = 3$, $wt(y) = 3$, *and* $wt(x + y) = 2$.

Proof. Note that $2V$ may be written as $\mathcal{B}_5(\mathbf{0}, 2) \cup \mathcal{B}_5(x, 1) \cup \mathcal{B}_5(y, 1) \cup \{x + y\}$. Hence, V is a tile of $\langle V \rangle$ if and only if there exists a set $A \subset \mathbb{F}_2^5$ of cardinality 4, such that

$$2A \cap \mathcal{B}_5(\mathbf{0}, 2) = \{\mathbf{0}\}, \tag{1}$$

$$2A \cap \mathcal{B}_5(x, 1) = 2A \cap \mathcal{B}_5(y, 1) = 2A \cap \{x + y\} = \emptyset. \tag{2}$$

Condition (1) means that the Hamming distance between any two vectors in A is at least 3. Since the (5,4,3) binary code is unique [22], we have that

$$A = \{\, 00000, 11100, 10011, 01111 \,\} \tag{3}$$

up to a permutation of coordinates. Note that A is linear, and therefore $2A = A$. Hence, condition (2) translates into

$$d(x, A) \geq 2, \qquad d(y, A) \geq 2, \qquad x + y \notin A. \tag{4}$$

The cases (a)–(d) now may be derived by inspection from (3) and (4). □

For $\text{rank}(V) = 4$ we have $V = \mathcal{B}_4(\mathbf{0}, 1) \cup \{x, y, z\}$. In this case it is obvious (from Corollary 3.2) that V is a tile if and only if $2V \neq \langle V \rangle$.

CLAIM 4.4. *V is a tile if and only if one of the following holds:*

(a) $wt(x) = wt(y) = wt(z) = 2$ *and* $wt(x + y) = wt(x + z) = wt(y + z) = 2$;
(b) $wt(x) = wt(y) = 2$, $wt(z) = 3$ *and* $wt(x+y) = 2$, $wt(x+z) = wt(y+z) = 1$;
(c) $wt(x) = 2$, $wt(y) = wt(z) = 3$ *and* *either* $wt(x + y) = 1$ *or* $wt(x + z) = 1$;
(d) $wt(x) = wt(y) = wt(z) = 3$.

Proof. All we need to show is that there exists a vector in $\mathbb{F}_2^4$ which is not a sum of two elements from $\mathcal{B}_4(\mathbf{0}, 1) \cup \{x, y, z\}$. This is easily done by inspection. For instance, (1111) is such a vector in case (a), and so $A = \{0000, 1111\}$. In all other cases A has the form $\{0000, 1110\}$. □

Finally, if $\operatorname{rank}(V) = 3$ then V is linear and hence is a trivial tile. Note that the foregoing classification shows that in all cases where $|V| \leq 8$ and V is a tile, it is also a linear tile. This also follows directly from Corollary 7.3.

We now consider tiles of large rank, that is, those tiles for which $\operatorname{rank}(V)$ is close to the upper bound $|V|-1$. For instance, if $\operatorname{rank}(V) = |V|-1$ then w.l.o.g. $V = \mathcal{B}_r(\mathbf{0}, 1)$ and is therefore a tile. If $\operatorname{rank}(V) = |V| - 2$ then w.l.o.g. $V = \mathcal{B}_r(\mathbf{0}, 1) \cup \{x\}$ for some $x \in \mathbb{F}_2^r$ of weight ≥ 2.

PROPOSITION 4.5. *If $V = \mathcal{B}_r(\mathbf{0}, 1) \cup \{x\}$ then V is a tile, provided $wt(x) \neq r-2, r-1$.*

Proof. We have $2V = \mathcal{B}_r(\mathbf{0}, 2) \cup \mathcal{B}_r(x, 1)$ with $r = 2^k - 2$. Thus, we may take A to be an $(r, 2^{r-k}, 3)$ linear code, obtained by shortening the $(r+1, 2^{r-k+1}, 3)$ Hamming code C, provided $d(x, 2A) = d(x, A) \geq 2$. It is well known [10, 22] that for any $3 \leq w \leq r-2$ or $w = r+1$ there exists a codeword $c \in C$ of weight w. Hence, if $wt(x) \neq r-2, r-1$, we may always find a permutation of C such that c coincides with x in the first r coordinates and has a 1 in the last coordinate. Upon shortening (that is, taking all codewords with a zero) in the last coordinate, we obtain A with $d(x, A) \geq 2$. □

As a consequence of Proposition 4.5, we have the following claim.

CLAIM 4.6. *If $|V| = 8$ and $\operatorname{rank}(V) = 6$, that is $V = \mathcal{B}_6(\mathbf{0}, 1) \cup \{x\}$, then V is a tile if and only if $wt(x) \neq 4, 5$.*

It follows from the proof of Proposition 4.5 that if $V = \mathcal{B}_r(\mathbf{0}, 1) \cup \{x\}$ and a tiling (V, A) exists, then A must have the parameters (n, M, d) of a shortened Hamming code. Further, it is easy to show that there cannot be a vector of weight $n-2$ or $n-1$ at distance ≥ 2 from any shortened perfect code C of length $n = 2^k - 2$ (either linear or nonlinear). Assume to the contrary that x is such a vector, and extend C to a perfect code $C_e = C_0 \cup C_1$ of length $n+1$. Here $C_0 = \{(c|0) : c \in C\}$ and $C_1 = \{(c|1) : c \in C^*\}$ for some C^*, where $(\cdot|\cdot)$ denotes concatenation. Since $d((x|0), C_e) \leq 1$ while $d((x|0), C_0) \geq 2$, it follows that $d((x|0), C_1) \leq 1$. Therefore $x \in C^*$ and hence C_e contains a codeword of weight $n-1$ or n, which is a contradiction. Thus, if any code with the parameters $(2^k - 2, 2^{2^k-k-2}, 3)$ can be obtained by shortening a perfect code, then no tiling is possible for $wt(x) = r-2$ or $wt(x) = r-1$. This is certainly true for $k = 3$, since the $(6, 8, 3)$ shortened Hamming code is unique, which establishes the "only if" part of Claim 4.6.

5. Periodicity of tilings. A tiling T of the Euclidean space $\mathbb{R}^n$ is said to be *periodic* if there exists a translation mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$ which takes T into itself. In other words, T is periodic if $t + T = T$ for some nonzero $t \in \mathbb{R}^n$. Similarly, we shall say that a tile V of $\mathbb{F}_2^n$ is periodic if there exists a nonzero $v \in \mathbb{F}_2^n$ such that $v + V = V$. Note that since $\mathbf{0} \in V$ by assumption, we must have $v \in V$. We call v a *periodic point* of V. To keep the notation rigorous, we shall consider $\mathbf{0}$ to be a periodic point of any set. A periodic point is also called a stabilizer of the set in some texts [14]. Given a tiling (V, A) of $\mathbb{F}_2^n$, we shall say that it is *nonperiodic* if neither V nor A contains a nonzero periodic point.

The existence of nonperiodic tilings of $\mathbb{R}^2$ is a well-known problem in tessellation theory [24]. The "kites and darts" tiling of Penrose [24] is a notable example of a nonperiodic tiling of the plane with two distinct polygons. It is still unknown, however, whether there exists a single (nonconvex) body which tiles the plane only nonperiodically.

In this section we consider the periodicity of tilings in $\mathbb{F}_2^n$. First, we construct a nonperiodic tiling of $\mathbb{F}_2^6$ and then show that $\mathbb{F}_2^6$ is the smallest binary space which

admits a nonperiodic tiling. We also present without proof a simple general construction of tilings in $\mathbb{F}_2^n$, which establishes the existence of nonperiodic tilings for all $n \geq 6$. Although this also follows from the general classification of groups that admit nonperiodic tilings, concluded by Sands [30], we hope these constructions will prove insightful. Furthermore, we show that the nonperiodic tiling of $\mathbb{F}_2^6$ is unique up to coordinate transformations.

PROPOSITION 5.1. *Let*

$$V = \{\mathbf{0}, v_1, \ldots, v_7\} = \left\{ \begin{array}{cccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right\}, \tag{5}$$

$$A = \{\mathbf{0}, a_1, \ldots, a_7\} = \left\{ \begin{array}{cccccccc} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right\}, \tag{6}$$

where the elements of $V, A \subset \mathbb{F}_2^6$ *are represented as column vectors. Then* (V, A) *is a nonperiodic tiling of* $\mathbb{F}_2^6$.

Proof. To see that (V, A) is a tiling, define $A_0 = \{\mathbf{0}, a_1, a_2, a_3\}$ and $A_1 = \{a_4, a_5, a_6, a_7\}$. Then obviously $2V \cap (A_0 + A_1) = \emptyset$ (consider the last row in (5), (6) above), and it remains to show that $2V \cap 2A_0 = 2V \cap 2A_1 = \{\mathbf{0}\}$. Note that the first five rows of (5) correspond to $\mathcal{B}_5(\mathbf{0}, 1) \cup \{x, y\}$ with $wt(x + y) = 5$, while the first five rows in both A_0 and A_1 are isomorphic to the tile set in (3). Thus, $2V \cap 2A_0 = 2V \cap 2A_1 = \{\mathbf{0}\}$ follows from Claim 4.3. To see that (V, A) is nonperiodic, observe that $\operatorname{rank}(V) = \operatorname{rank}(A) = 5$. It is obvious that a set of cardinality 8 and rank 5 cannot be periodic. □

We now show that the nonperiodic tiling of Proposition 5.1 is the smallest possible.

PROPOSITION 5.2. *If* (V, A) *is a nonperiodic tiling, then* $|V| \geq 8$ *and* $|A| \geq 8$.

Proof. Obviously, any tile of cardinality 2 is linear and, hence, periodic. Now let (V, A) be a tiling of $\mathbb{F}_2^n$ with $|A| = 4$. We claim that either A is linear or V is periodic. Indeed, let $A = \{\mathbf{0}, a, b, c\}$ and define $A' = \{\mathbf{0}, a, b, a + b\}$. Then $2A' \subseteq 2A$ which implies $2V \cap 2A' = \{\mathbf{0}\}$. Hence, both (V, A) and (V, A') are tilings of $\mathbb{F}_2^n$, and therefore we must have $c + V = (a + b) + V$. This is only possible if $c = a + b$, in which case A is linear, or if $c = (a + b) + v$, where v is a nonzero periodic point of V. □

Furthermore, it can be shown that the tiling of Proposition 5.1 is unique. Namely, in any nonperiodic tiling of $\mathbb{F}_2^6$ both tiles must be of the form (5), up to coordinate transformations. For example, under the coordinate transformation corresponding to taking $\{a_1, a_2, a_4, a_5, a_6, \mathbf{1}\}$ as the basis of $\mathbb{F}_2^6$, where $\mathbf{1}$ denotes the all-one vector, the tile A given in (6) maps into (5). In view of Proposition 5.2, in order to establish the uniqueness of (5), (6) we only need to consider tiles of cardinality 8. All such tiles have been classified in §4. Furthermore, if $|V| = 8$ and V is a nonperiodic tile of $\mathbb{F}_2^6$, then obviously $4 \leq \operatorname{rank}(V) \leq 6$. If $\operatorname{rank}(V) = 6$ then V can tile $\mathbb{F}_2^6 = \langle V \rangle$ only linearly, in view of Proposition 4.5 and the fact that a $(6, 8, 3)$ code must be linear. Hence $\operatorname{rank}(V) = 4$ or $\operatorname{rank}(V) = 5$. Following the various cases in Claims 4.3 and

4.4, it may be readily verified that none of them produces a nonperiodic tiling, except case (c) in Claim 4.3. This is precisely the case corresponding to (5).

We now describe a general construction of tilings in $\mathbb{F}_2^n$ which shows that nonperiodic tilings exist for all odd $n \geq 7$. Let $\nu = 2^m - 1$, where $m \geq 3$, and let $h_1, h_2, \ldots, h_\nu$ be the distinct nonzero elements of $\mathbb{F}_2^m$ arranged in some definite, say lexicographic, order. Fix a permutation π on the set $\{1, 2, \ldots, \nu\}$, and consider $A_0, A_1, V \subset \mathbb{F}_2^{2m+1}$ given by

$$A_0 = \left\{ \begin{array}{ccccc} \mathbf{0} & h_1 & h_2 & \cdots & h_\nu \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ 0 & 0 & 0 & \cdots & 0 \end{array} \right\}, \tag{7}$$

$$A_1 = \left\{ \begin{array}{ccccc} \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & h_1 & h_2 & \cdots & h_\nu \\ 1 & 1 & 1 & \cdots & 1 \end{array} \right\}, \tag{8}$$

$$V = \left\{ \begin{array}{ccccc} \mathbf{0} & h_1 & h_2 & \cdots & h_\nu \\ \mathbf{0} & h_{\pi(1)} & h_{\pi(2)} & \cdots & h_{\pi(\nu)} \\ 0 & 0 & 0 & \cdots & 0 \end{array} \right\}, \tag{9}$$

where the elements of A_0, A_1, V are again represented as column vectors, and $\binom{\cdot}{\cdot}$ denotes concatenation. Let $A = A_0 \cup A_1$. Then it is easy to see that (V, A) is a tiling of $\mathbb{F}_2^{2m+1}$. Further, the set $A = A_0 \cup A_1$ is clearly nonperiodic. It may also be shown that if the permutation π is given by, for instance,

$$\pi = (1,4,2)(3,6)(5,7)(8,9)(10,11)\cdots(\nu-1,\nu),$$

then the set V in (9) is nonperiodic as well. Thus, we have constructed nonperiodic tilings of $\mathbb{F}_2^n$ for all odd $n \geq 7$. To exhibit nonperiodic tilings of $\mathbb{F}_2^n$ for even n we use a variant of the same construction. As before, let $\nu = 2^m - 1$. Now let π be any derangement of the set $\{1, 2, \ldots, \nu\}$. Define $A, V \in \mathbb{F}_2^{2m+2}$ as follows:

$$A = \left\{ \begin{array}{ccccc} \mathbf{0} & h_1 & h_2 & \cdots & h_\nu \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \end{array} \right\} \bigcup \left\{ \begin{array}{ccccc} \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & h_1 & h_2 & \cdots & h_\nu \\ 1 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{array} \right\}, \tag{10}$$

$$V = \left\{ \begin{array}{ccccc} \mathbf{0} & h_1 & h_2 & \cdots & h_\nu \\ \mathbf{0} & h_{\pi(1)} & h_{\pi(2)} & \cdots & h_{\pi(\nu)} \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \end{array} \right\} \bigcup \left\{ \begin{array}{ccccc} \mathbf{0} & h_1 & h_2 & \cdots & h_\nu \\ \mathbf{0} & h_1 & h_2 & \cdots & h_\nu \\ 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \end{array} \right\}. \tag{11}$$

Then (V, A) is a nonperiodic tiling of $\mathbb{F}_2^{2m+2}$ for all $m \geq 2$. We omit the proof of (7)–(11).

6. Recursive decomposition of tilings. In this section we present a recursive decomposition (construction) of tilings, which shows that any tiling of $\mathbb{F}_2^n$ may be decomposed into (constructed from) smaller tilings of certain particular type. First, we prove the following proposition which was used in the classification of §4.

PROPOSITION 6.1. *A set $V \subset \mathbb{F}_2^n$ is a tile of $\mathbb{F}_2^n$ if and only if it is a tile of $\langle V \rangle$.*

Proof. ($\Leftarrow$) Since $\langle V \rangle$ is linear it is a tile of $\mathbb{F}_2^n$. Thus, if $(\langle V \rangle, A_1)$ is a tiling of $\mathbb{F}_2^n$ and (V, A_0) is a tiling of $\langle V \rangle$, then evidently $(V, A_0 + A_1)$ is a tiling of $\mathbb{F}_2^n$. ($\Rightarrow$) Let (V, A) be a tiling of $\mathbb{F}_2^n$ and define $A_0 = A \cap \langle V \rangle$. We claim that (V, A_0) is a tiling of $\langle V \rangle$. Indeed, since $A_0 \subseteq A$ and $2V \cap 2A = \{\mathbf{0}\}$, it follows that $2V \cap 2A_0 = \{\mathbf{0}\}$. Further, since $\langle V \rangle \subset \mathbb{F}_2^n = V + A$, any $w \in \langle V \rangle$ can be written as $w = v + a$, where $v \in V$ and $a \in A$. However, since $\langle V \rangle$ is linear we have $a = v + w \in \langle V \rangle$, so $a \in A_0$. It follows that $\langle V \rangle \subseteq V + A_0$. The converse inclusion $V + A_0 \subseteq \langle V \rangle$ is obvious from $V, A_0 \subset \langle V \rangle$. □

In view of Proposition 6.1, we shall be particularly interested in those tilings (V, A) for which $\mathbb{F}_2^n = \langle V \rangle$. Note that until now the order of the sets in the tiling pair (V, A) was of no importance, since the roles of V and A were completely symmetric. However, this is no longer true if we require $\langle V \rangle = \mathbb{F}_2^n$.

DEFINITION 6.1. *The ordered pair (V, A) is a* proper tiling *of $\mathbb{F}_2^n$ if (V, A) is a tiling of $\mathbb{F}_2^n$ and $\langle V \rangle = \mathbb{F}_2^n$.*

The following proposition shows that the classification of all tilings in $\mathbb{F}_2^n$ may, in principle, be reduced to the study of proper tilings. Let V be a tile of $\mathbb{F}_2^n$ with $\langle V \rangle \neq \mathbb{F}_2^n$ or, equivalently, $\operatorname{rank}(V) < n$. Denote $m = 2^{n-r} - 1$, where $r = \operatorname{rank}(V)$.

THEOREM 6.2. *The pair (V, A) is a tiling of $\mathbb{F}_2^n$ if and only if A has the following form:*

1. *For $i = 0, 1, \ldots, m$, let $A_i \subset \langle V \rangle$ be such that (V, A_i) is a tiling of $\langle V \rangle$.*
2. *Let $c_0 = \mathbf{0}, c_1, \ldots, c_m$ be a set of representatives for $\mathbb{F}_2^n / \langle V \rangle$.*
3. *For $i = 0, 1, \ldots, m$, let v_i be any element of $\langle V \rangle$.*

Then

$$A = A_0 \cup (v_1 + c_1 + A_1) \cup \cdots \cup (v_m + c_m + A_m). \tag{12}$$

Proof. ($\Leftarrow$) Let A be as in (12), and denote $|V| = 2^k$. Then $|A_i| = 2^{r-k}$ for all i, and $|A| = (m+1)2^{r-k} = 2^{n-k}$. Hence, it remains to show that $2V \cap 2A = \{\mathbf{0}\}$. We have $2A = \mathcal{U} \cup \mathcal{W}$, where $\mathcal{U} = \cup_{i=0}^{m} 2A_i$ and $\mathcal{W} = \cup_{0 \le i < j \le m} (v_i + v_j + c_i + c_j + A_i + A_j)$. Now, $2V \cap \mathcal{U} = \{\mathbf{0}\}$ since $2V \cap 2A_i = \{\mathbf{0}\}$ for all i. Since $c_i + c_j \notin \langle V \rangle$ for all $0 \le i < j \le m$, it follows that $\langle V \rangle$ and $\mathcal{W}$ are disjoint, and hence $2V \cap \mathcal{W} = \emptyset$. ($\Rightarrow$) Let (V, A) be a tiling of $\mathbb{F}_2^n$. Pick any set of representatives $c_0 = \mathbf{0}, c_1, c_2, \ldots, c_m$ for $\mathbb{F}_2^n / \langle V \rangle$ and define

$$A_i \;=\; v_i + c_i + (A \cap (c_i + \langle V \rangle)),$$

where v_i is any element of $\langle V \rangle$ such that $\mathbf{0} \in A_i$. To see that such v_i exists, note that $c_i + (A \cap c_i + \langle V \rangle) \subset \langle V \rangle$. We have $v_i + c_i + A_i = A \cap (c_i + \langle V \rangle)$ and, hence,

$$\bigcup_{i=0}^{m} (v_i + c_i + A_i) \;=\; \bigcup_{i=0}^{m} A \cap (c_i + \langle V \rangle) \;=\; A. \tag{13}$$

We need to show that (V, A_i) is a tiling of $\langle V \rangle$ for all i. Clearly $2A_i = (A \cap (c_i + \langle V \rangle)) + (A \cap (c_i + \langle V \rangle)) \subset 2A$, and therefore $2V \cap 2A_i = \{\mathbf{0}\}$. Note that $A_i \subset \langle V \rangle$, which implies $V + A_i \subseteq \langle V \rangle$. Thus, to establish that (V, A_i) is a tiling of $\langle V \rangle$, it remains to show that $|A_i| = 2^{r-k}$. Since $2V \cap 2A_i = \{\mathbf{0}\}$, we obviously have $|A_i| \le 2^{r-k}$. However,

$(m+1)2^{r-k} = 2^{n-k} = |A| \leq \sum_{i=0}^{m} |A_i|$, where the last inequality is from (13). This implies $|A_i| = 2^{r-k}$ and completes the proof. □

Remark. The vectors $v_1, v_2, \ldots, v_m$ and the sets $A_0, A_1, \ldots, A_m$ in Theorem 6.2 are not necessarily distinct. It is assumed that $\mathbf{0} \in A_i$ for all $i = 0, 1, \ldots, m$.

The analysis of Theorem 6.2 shows that if all the proper tilings of $\mathbb{F}_2^r$ are known for $r = 1, 2, \ldots, n$, we can construct all the tilings of $\mathbb{F}_2^n$. However, as will now be shown, the set of all the proper tilings is not the smallest class of tilings which permits complete classification of all tilings of $\mathbb{F}_2^n$. Indeed, let (V, A) be a proper tiling of $\langle V \rangle$ and consider the tiling (A, V). Unless $\operatorname{rank}(A) = \operatorname{rank}(V)$, this tiling is not proper and, hence, by Theorem 6.2

$$V = V_0 \cup (a_1 + c_1 + V_1) \cup \cdots \cup (a_m + c_m + V_m), \tag{14}$$

where (A, V_i) is a proper tiling of $\langle A \rangle$ for all i, $\mathbf{0}, c_1, \ldots, c_m$ are representatives of $\langle V \rangle / \langle A \rangle$, and $a_1, a_2, \ldots, a_m \in \langle A \rangle$. Thus, using (14), each of the tilings (V, A_i) of $\langle V \rangle$ in Theorem 6.2 may be decomposed into yet smaller tilings, unless $\langle A_i \rangle = \langle V \rangle$. This process may be iterated until a complete decomposition of the original tiling (V, A) is obtained.

Example. Consider the tiling (V, A) of $\mathbb{F}_2^6$ given in (5) and (6). This tiling is not proper since $\operatorname{rank}(V) = \operatorname{rank}(A) = 5 < 6$. If we take $c_0 = \mathbf{0}$ and $c_1 = (000001)$ as the representatives of $\mathbb{F}_2^6 / \langle V \rangle$, we can write $A = A_0 \cup (c_1 + A_1)$ with

$$A_0 = \left\{ \begin{array}{cccc} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array} \right\}, \qquad A_1 = \left\{ \begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{array} \right\}.$$

Note that (V, A_0) and (V, A_1) are proper tilings of $\langle V \rangle$. Since $\operatorname{rank}(A_0) = \operatorname{rank}(A_1) < \operatorname{rank}(V)$, we may further decompose (V, A_0) and (V, A_1) as follows. Let $\mathbf{0}, c_{0,1}, \ldots, c_{0,7}$ and $\mathbf{0}, c_{1,1}, \ldots, c_{1,7}$ be the representatives of $\langle V \rangle / \langle A_0 \rangle$ and $\langle V \rangle / \langle A_1 \rangle$, respectively. We may take, for instance,

$$\{c_{0,1}, c_{0,2}, \ldots, c_{0,7}\} = \{c_{1,1}, c_{1,2}, \ldots, c_{1,7}\} = \left\{ \begin{array}{ccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right\}.$$

Then the corresponding decompositions of V are given by

$$V = V_{0,0} \cup (a_{0,1} + c_{0,1} + V_{0,1}) \cup \cdots \cup (a_{0,7} + c_{0,7} + V_{0,7}),$$
$$V = V_{1,0} \cup (a_{1,1} + c_{1,1} + V_{1,1}) \cup \cdots \cup (a_{1,7} + c_{1,7} + V_{1,7}),$$

where

$$a_{0,1}, a_{0,2}, \ldots, a_{0,7} = \mathbf{0}, \mathbf{0}, (110100), \mathbf{0}, (101010), (011110), \mathbf{0} \in \langle A_0 \rangle,$$
$$a_{1,1}, a_{1,2}, \ldots, a_{1,7} = \mathbf{0}, \mathbf{0}, (110010), \mathbf{0}, (101110), (011100), \mathbf{0} \in \langle A_1 \rangle,$$

and

$$V_{0,0} = \cdots = V_{0,7} = V_{1,0} = \cdots = V_{1,7} = \{\mathbf{0}\}.$$

Indeed, since A_0 and A_1 are linear, both $(A_0, V_{0,i}) = (A_0, \{\mathbf{0}\})$ and $(A_1, V_{1,i}) = (A_1, \{\mathbf{0}\})$ are trivial tilings of $\langle A_0 \rangle = A_0$ and $\langle A_1 \rangle = A_1$, respectively. □

As illustrated in the foregoing example, the recursive decomposition of (12) and (14) terminates when trivial tilings, of the form $(V, \{\mathbf{0}\})$ with V linear, are obtained. The only other case where the recursion of (12), (14) stops is when a tiling (V, A) with $\langle V \rangle = \langle A \rangle$ is encountered.

DEFINITION 6.2. *A tiling (V, A) of $\mathbb{F}_2^n$ is said to be of* full rank *if $\langle V \rangle = \langle A \rangle$.*

It is easy to see that a tiling (V, A) of $\mathbb{F}_2^n$ is of full rank if and only if $\text{rank}(V) = \text{rank}(A) = n$. In the sense described in this section, any tiling of $\mathbb{F}_2^n$ can be constructed in a unique way from the trivial tilings and tilings of full rank. This clearly leads to the following question: do full-rank tilings exist? We shall settle this question in the affirmative in the next section.

7. Tilings and perfect binary codes. A perfect binary code of length n and Hamming distance $2R+1$ is a tiling of $\mathbb{F}_2^n$ with Hamming spheres of radius R. Perfect codes have been extensively studied; see for instance [10, 18, 23, 25, 26, 34, 36]. It is known [18, 34] that perfect codes exist only for $R=0$, $R=n$, $R=(n-1)/2$ with n odd, $R=1$ with $n=2^m-1$, and $R=3$ with $n=23$. Since the first three cases are trivial while the last case corresponds to the well-known binary Golay code [22], we shall henceforth use the word "perfect" to refer to the perfect codes with $R=1$ and $n=2^m-1$.

It can be shown that each tiling (V, A) of $\mathbb{F}_2^n$ is uniquely associated with a perfect binary code of length $\nu = |V| - 1$. A similar result in the context of coverings was established by Blokhuis and Lam [2]. Let $H(V)$ be an $n \times \nu$ matrix having the elements of $V \setminus \{\mathbf{0}\}$, arranged in some fixed order, as its columns. For $x \in \mathbb{F}_2^\nu$, let $s(x) = H(V)x^t$ denote the syndrome of x with respect to $H(V)$.

PROPOSITION 7.1. *Let*

$$C = \{\, c \in \mathbb{F}_2^\nu \;:\; s(c) = H(V)c^t \in A \,\}. \tag{15}$$

Then C is a perfect code of length ν.

Proof. We first show that $d(C) = \min_{c_1, c_2 \in C} d(c_1, c_2) \geq 3$. Denote $a_1 = s(c_1)$ and $a_2 = s(c_2)$, where $a_1, a_2 \in A$ by (15). Suppose that $a_1 = a_2$ or, equivalently, $s(c_1+c_2) = \mathbf{0}$. Then $d(c_1, c_2) = wt(c_1+c_2) \geq 3$ since the columns of $H(V)$ are distinct. Now suppose that $a_1 \neq a_2$. Note that $a_1 + a_2 = s(c_1 + c_2) = \sum_{i \in \text{supp}(c_1+c_2)} v_i$. Further note that $a_1 + a_2 \notin 2V$, since $2V \cap 2A = \{\mathbf{0}\}$. Hence, again $d(c_1, c_2) = |\text{supp}(c_1 + c_2)| \geq 3$. It remains to show that $d(x, C) \leq 1$ for all $x \in \mathbb{F}_2^\nu$. Since $V + A = \mathbb{F}_2^n$, we have $s(x) = v + a$ for some $v \in V$ and $a \in A$. If $v = \mathbf{0}$ then $x \in C$ by (15) and we are done. Otherwise, let $c \in \mathbb{F}_2^\nu$ be a vector which coincides with x in all positions except one, which is the position corresponding to the location of v in $H(V)$. Then $d(x, c) = 1$, and $s(c) = a$ which implies that $c \in C$ by (15). □

Proposition 7.1 provides a means for constructing a perfect code C from any given tiling (V, A). We shall say that this code C is the perfect code *associated* with (V, A). It should be pointed out that the correspondence between sets V, A such that $V + A = \mathbb{F}_2^n$ and coverings by spheres of radius $R = 1$ has been initially noticed by Blokhuis and Lam in [2]. The relevance of their rank, however, seems to have been overlooked in [2]. We now elaborate on this issue.

PROPOSITION 7.2. *If C is the perfect code associated with a proper tiling (V, A), then*

$$\text{rank}(C) = \nu - \text{rank}(V) + \text{rank}(A).$$

Proof. Define $C_0 = \{c \in \mathbb{F}_2^\nu : s(c) = \mathbf{0}\}$. Clearly, C_0 is a linear code and $\dim C_0 = \nu - \operatorname{rank}(V)$. Now let $A = \{\mathbf{0}, a_1, \ldots, a_\mu\}$ where $\mu = |A| - 1$. Since $\langle V \rangle = \mathbb{F}_2^n$, we can always find a set $C_1 = \{\mathbf{0}, c_1, \ldots, c_\mu\} \subset \mathbb{F}_2^\nu$ such that $s(c_i) = a_i$ for all i, $\operatorname{rank}(C_1) = \operatorname{rank}(A)$, and $\langle C_1 \rangle \cap C_0 = \{\mathbf{0}\}$. Then $C = \bigcup_{c \in C_1} (c + C_0)$ and $\operatorname{rank}(C) = \dim C_0 + \operatorname{rank}(C_1) = \nu - \operatorname{rank}(V) + \operatorname{rank}(A)$. □

Remark. If (V, A) is not a *proper* tiling then Proposition 7.2 does not apply. This is so because for $\operatorname{rank}(V) < n$ there exist elements $a \in A$ which cannot be represented in the form $H(V)x^t$ for any $x \in \mathbb{F}_2^\nu$. In this case we have

$$\operatorname{rank}(C) \;=\; \nu - \operatorname{rank}(V) + \operatorname{rank}(A_0) \;<\; \nu - \operatorname{rank}(V) + \operatorname{rank}(A),$$

where $A_0 = A \cap \langle V \rangle$.

Note that C_0 is a linear subcode of C (regardless of whether (V, A) is proper or not), and C itself is linear, i.e., equivalent to the Hamming code of length ν, if and only if $A_0 = A \cap \langle V \rangle$ is linear. The following corollary is an immediate consequence of this fact.

COROLLARY 7.3. *If $|V| \leq 8$ and (V, A) is a proper tiling, then A is linear.*

Proof. The perfect code C associated with (V, A) has length at most 7 and, hence, is equivalent (see, e.g., [25]) to the Hamming $(7, 16, 3)$ code or to the $(3, 2, 3)$ repetition code. □

We now employ the foregoing results on the relation between tilings and perfect codes to establish the existence of full-rank tilings. Indeed, a full-rank perfect code (that is, a perfect code of length ν and rank ν) together with the Hamming sphere $\mathcal{B}_\nu(\mathbf{0}, 1)$ constitute an example of a full-rank tiling. Furthermore, we have shown that full-rank tilings exist if and only if there exist full-rank perfect codes, for if (V, A) is a full-rank tiling then the associated perfect code must have full rank by Proposition 7.2. Unfortunately, none of the classical constructions of nonlinear perfect codes given in [1, 23, 25, 26, 36] produces perfect codes of full rank. On the other hand, such codes have been recently constructed by Etzion and Vardy in [10].

THEOREM 7.4 (see [10]). *For all $m \geq 4$, there exists a full-rank perfect code of length $2^m - 1$.*

Thus, full-rank tilings of $\mathbb{F}_2^n$ exist for all $n = 2^m - 1$ with $m \geq 4$. It is shown in the next section that they also exist for $n = 2^m - 2$ with $m \geq 4$, and in particular for $n = 14$. As a consequence of Corollary 7.3, full-rank tilings do not exist for $n \leq 7$. It is still an open question whether full-rank tilings exist for $n = 8, 9, \ldots, 13$, and many other values of n. However, we have the following theorem.

THEOREM 7.5. *Full-rank tilings of $\mathbb{F}_2^n$ exist for all sufficiently large n.*

Proof. Let (V_1, A_1) and (V_2, A_2) be tilings of $\mathbb{F}_2^{n_1}$ and $\mathbb{F}_2^{n_2}$, respectively. Define

$$V = \{\, (v_1 \,|\, v_2) \;:\; v_1 \in V_1,\; v_2 \in V_2 \,\},$$
$$A = \{\, (a_1 \,|\, a_2) \;:\; a_1 \in A_1,\; a_2 \in A_2 \,\}.$$

Then clearly $2V \cap 2A = \{\mathbf{0}\}$ and $|V| \cdot |A| = 2^{n_1+n_2}$, which means that (V, A) is a tiling of $\mathbb{F}_2^{n_1+n_2}$. Now $\operatorname{rank}(V) = \operatorname{rank}(V_1) + \operatorname{rank}(V_2)$ and $\operatorname{rank}(A) = \operatorname{rank}(A_1) + \operatorname{rank}(A_2)$. Thus, if (V_1, A_1) and (V_2, A_2) are full-rank tilings then so is (V, A). By Theorem 7.4 there exist full-rank tilings of $\mathbb{F}_2^n$ for $n = 15, 31, 63, \ldots$. Since 15 and 31 are relatively prime, by the conductor theorem of Frobenius [31, p. 376] we have full-rank tilings of $\mathbb{F}_2^n$ for all $n \geq n_0$, where n_0 is sufficiently large. □

Remark. In fact, using the results of the next section in the proof of Theorem 7.5 it is possible to show that full-rank tilings of $\mathbb{F}_2^n$ exist for all $n \geq 112$. This leaves exactly 53 values of n for which we do not know whether full-rank tilings exist.

We conclude this section with a construction of proper tilings from perfect binary codes. This construction is, in a sense, the converse of Proposition 7.1. Let C be a perfect binary code of length ν and let Γ be a linear subcode of C such that $\Gamma + C = C$. For instance, by Proposition 8.1 of the next section, we may take as Γ any linear subspace of the set of all the periodic points of C (the *kernel* of C in the terminology of [1, 27]). Denote $\gamma = \dim \Gamma$, and let $H(\Gamma)$ be a $(\nu - \gamma) \times \nu$ parity-check matrix of Γ. Note that the matrix $H(\Gamma)$ is full rank by assumption. Take $V = \{\mathbf{0}\} \cup \{\text{the columns of } H(\Gamma)\}$ and define $A = \{H(\Gamma)c^t : c \in C\}$.

PROPOSITION 7.6. *With V and A as defined above, the ordered pair (V, A) is a proper tiling of $\mathbb{F}_2^{\nu-\gamma}$.*

Proof. For $x \in \mathbb{F}_2^\nu$ let $s(x) = H(\Gamma)x^t$. We claim that

$$C = \{ c \in \mathbb{F}_2^\nu : s(c) = H(\Gamma)c^t \in A \}. \tag{16}$$

Indeed, if $c \in C$ then $s(c) \in A$ by the definition of A. To see the converse inclusion, consider a vector $x \in \mathbb{F}_2^\nu$ such that $s(x) = a \in A$. By the definition of A, there exists $c \in C$ such that $s(c) = a$. Hence $s(x + c) = \mathbf{0}$, which implies that $x + c \in \Gamma$. But since $\Gamma + C = C$ it follows that $x = (x + c) + c \in C$.

It is now easy to see that $d(C) = 3$ implies $2V \cap 2A = \{\mathbf{0}\}$. Suppose to the contrary that $v_1 + v_2 = a_1 + a_2$ for some $v_1 \neq v_2 \in V$ and $a_1 \neq a_2 \in A$. By (16) and the fact that $\langle V \rangle = \mathbb{F}_2^{\nu-\gamma}$, there exists a codeword $c_1 \in C$ such that $s(c_1) = a_1$. Let $c_2 \in \mathbb{F}_2^\nu$ be a vector which coincides with c_1 in all but the two positions corresponding to the locations of v_1 and v_2 in $H(\Gamma)$. Then $s(c_2) = a_1 + v_1 + v_2 = a_2 \in A$, and therefore $c_2 \in C$ by (16). This is a contradiction, since $d(c_1, c_2) = 2$.

The fact that $V + A = \mathbb{F}_2^{\nu-\gamma}$ also follows from (16). Let $x \in \mathbb{F}_2^{\nu-\gamma}$. Since $\langle V \rangle = \mathbb{F}_2^{\nu-\gamma}$, there exists $y \in \mathbb{F}_2^\nu$ such that $x = s(y)$. If $y \in C$ then $x \in A$ by (16), and we are done. Otherwise, there exists a codeword $c \in C$ at distance 1 from y. Let $s(c) = a \in A$. Then $x = a + v$, where v is the column of $H(\Gamma)$ located at the position where c and y differ. □

Note that γ is possibly 0, in which case V is a sphere and $A = C$. However, as will be shown in the next section, given any perfect code C we may always find a linear subcode $\Gamma \neq \{\mathbf{0}\}$ such that $\Gamma + C = C$.

8. Decomposition of full-rank tilings. In this section we show that a periodic full-rank tiling may be further decomposed into smaller tilings in a way similar to Theorem 6.2. Analysis of the rank and periodicity of tilings resulting from such recursive decomposition is presented. Using this analysis we deduce the existence of nonperiodic full-rank tilings.

Let $A \subset \mathbb{F}_2^n$ and let A_0 denote the set of all the periodic points of A. The set A_0 is sometimes called the *kernel* of A [1, 27]. The following proposition is rather obvious and was first established in [1]. We include the proof herein for completeness.

PROPOSITION 8.1. *The set A_0 is a linear subcode of A. Furthermore, A is the union of disjoint additive cosets of A_0.*

Proof. Proposition 8.1 holds vacuously if A is either nonperiodic or linear. Hence assume that $A_0 \neq \{\mathbf{0}\}$ and $A_0 \neq A$. To see that A_0 is linear, note that $(a_1 + a_2) + A = a_1 + (a_2 + A) = a_1 + A = A$ for any $a_1, a_2 \in A_0$. To see that A_0 tiles A, let $a \in A \setminus A_0$. Then $(a + A_0) \cap A_0 = \emptyset$ since A_0 is linear, and $a + A_0 \subset A$ since A_0 consists of periodic points of A. If $A = A_0 \cup (a + A_0)$ we are done; otherwise, continue in this manner until A is exhausted. □

It follows from Proposition 8.1 that there exists $A' \subset A$ such that $A' + A_0 = A$ and $2A_0 \cap 2A' = \{\mathbf{0}\}$. We shall write $A' = A/A_0$. More generally, for any set $V \subset \mathbb{F}_2^n$,

we define $V' = V/A_0$ as follows. Fix a basis $a_1^*, a_2^*, \ldots, a_m^*$ for A_0 where $m = \dim A_0$, and complete this to a basis $a_1^*, a_2^*, \ldots, a_m^*, b_1, b_2, \ldots, b_{n-m}$ for $\mathbb{F}_2^n$. Then each vector $v = \sum_{i=1}^m \alpha_i a_i^* + \sum_{i=1}^{n-m} \beta_i b_i$ in V is mapped onto the vector $v' = \sum_{i=1}^{n-m} \beta_i b_i$ in V/A_0. Thus V/A_0 is just the projection of V onto $\mathbb{F}_2^n/A_0$. Note that the mapping from V to V/A_0 is not necessarily one-to-one. Thus, for instance, the mapping from A to $A' = A/A_0$ is $|A_0|$ to one, and it is easy to see that the two definitions of A/A_0 coincide.

THEOREM 8.2. *Let (V, A) be a tiling of $\mathbb{F}_2^n$ and let A_0 be the set of periodic points of A. Then A has the following form:*

$$A = A' \cup (c_1 + A') \cup \cdots \cup (c_{2^m-1} + A'), \tag{17}$$

where $(V/A_0, A')$ is a tiling of $\mathbb{F}_2^n/A_0$, and $\mathbf{0}, c_1, \ldots, c_{2^m-1}$ are representatives for $\mathbb{F}_2^n/A_0$ in $\mathbb{F}_2^n$.

Proof. Set $A' = A/A_0$ and $\{\mathbf{0}, c_1, c_2, \ldots, c_{2^m-1}\} = A_0$. Then (17) holds by Proposition 8.1, and all we need to show is that $(V/A_0, A/A_0)$ is a tiling of $\mathbb{F}_2^n/A_0$. First, we claim that the mapping from V to V/A_0 is one-to-one in this case. Indeed, suppose to the contrary that $\exists v_1, v_2 \in V$ which map onto the same vector $v' \in V/A_0$. Then $v_1 = \sum_{i=1}^m \alpha_i a_i^* + v'$ and $v_2 = \sum_{i=1}^m \beta_i a_i^* + v'$ which implies that $v_1 + v_2 = \sum_{i=1}^m (\alpha_i + \beta_i) a_i^* \in A_0$. This is a contradiction since $A_0 \subset A$ and $2V \cap 2A = \{\mathbf{0}\}$. Thus $|V/A_0| = |V|$, and therefore $|V/A_0| \cdot |A/A_0| = |\mathbb{F}_2^n/A_0|$. It remains to show that $2(V/A_0) \cap 2(A/A_0) = \{\mathbf{0}\}$. Again, suppose to the contrary that $v_1' + v_2' = a_1' + a_2'$ for some $v_1' \neq v_2' \in V/A_0$ and $a_1' \neq a_2' \in A/A_0$. Let v_1, v_2 be the two vectors in V which map onto v_1', v_2', and let a_1, a_2 be some two vectors in A that map onto a_1', a_2'. Then

$$v_1 + v_2 = \sum_{i=1}^m \alpha_i a_i^* + \sum_{i=1}^m \beta_i a_i^* + v_1' + v_2',$$

$$a_1 + a_2 = \sum_{i=1}^m \gamma_i a_i^* + \sum_{i=1}^m \delta_i a_i^* + a_1' + a_2',$$

which implies, by linearity of A_0, that $v_1 + v_2 = a_0 + a_1 + a_2$ for some $a_0 \in A_0$. But, since A_0 consists of periodic points of A, we have $a_0 + a_1 \in A$ which contradicts $2V \cap 2A = \{\mathbf{0}\}$. □

Remark. Note that Proposition 8.1 and Theorem 8.2 hold without change if A_0 is any linear subspace of the set of all the periodic points of A.

Using Theorem 8.2, a full-rank tiling (V, A) of $\mathbb{F}_2^n$ may be further decomposed into smaller tilings, provided that at least one of the sets V, A is periodic. We have the following sufficient condition for periodicity of proper tilings.

PROPOSITION 8.3. *Let (V, A) be a proper tiling of $\mathbb{F}_2^n$ and let $\widetilde{v} = \sum_{v \in V} v$. Then $\widetilde{v}$ is a periodic point of A.*

Proof. Let C be the perfect binary code of length $\nu = |V| - 1$ associated with (V, A). Since the weight distribution of any perfect code containing $\mathbf{0}$ is uniquely determined [22], the vector $\mathbf{1}$ of weight ν belongs to $c + C$ for all $c \in C$. Thus, $\mathbf{1}$ is a periodic point of C. Now, $\widetilde{v} = H(V)\mathbf{1}^t = s(\mathbf{1})$ which shows that $\widetilde{v} \in A$. Furthermore, since the tiling (V, A) is proper, for any $a \in A$ there exists $c \in C$ such that $s(c) = a$. But then $\mathbf{1} + c \in C$, and therefore $s(\mathbf{1} + c) = s(\mathbf{1}) + s(c) = \widetilde{v} + a$ belongs to A. □

By Proposition 8.3 a full-rank tiling (V, A) is nonperiodic only if

$$\sum_{v \in V} v = \sum_{a \in A} a = \mathbf{0}.$$

In particular, a full-rank tiling consisting of the Hamming sphere and a full-rank perfect code is necessarily periodic.

Example. Using Theorems 7.4 and 8.2 it is possible to construct a full-rank tiling of $\mathbb{F}_2^{14}$. Let $V = \mathcal{B}_{15}(\mathbf{0}, 1)$ and let A be the full-rank perfect code of length 15 constructed in [10]. Since $\mathbf{1}$ is a periodic point of A, we may take $A_0 = \{\mathbf{0}, \mathbf{1}\}$ and let $u_1, u_2, \ldots, u_{14}, \mathbf{1}$ be the corresponding basis of $\mathbb{F}_2^{15}$, where u_j denotes a vector of weight 1 with the nonzero entry in the jth position. Then

$$V/A_0 = \{\mathbf{0}, u_1, \ldots, u_{14}\} \cup \{(111111111111110)\},$$

and A/A_0 is the set of all codewords of A with a zero in the last position. Upon puncturing in the last coordinate we obtain a full-rank tiling (V', A') of $\mathbb{F}_2^{14}$.

The following two propositions are concerned with the periodicity of A/A_0 and V/A_0.

PROPOSITION 8.4. *If A_0 contains all the periodic points of A, then A/A_0 is nonperiodic.*

Proof. Let a be a periodic point of A/A_0. Then $a + A = a + (A/A_0) + A_0 = (A/A_0) + A_0 = A$. Hence $a \in A_0$. But $(A/A_0) \cap A_0 = \{\mathbf{0}\}$, and therefore $a = \mathbf{0}$. □

PROPOSITION 8.5. *The mapping from V to V/A_0 takes periodic points of V into periodic points of V/A_0.*

Proof. Let $v_0 = \sum_{i=1}^{m} \alpha_i a_i^* + \sum_{i=1}^{n-m} \beta_i b_i$ be a periodic point of V, and let $v_0' = \sum_{i=1}^{n-m} \beta_i b_i$ be its image in V/A_0. Clearly, for any $v = \sum_{i=1}^{m} \gamma_i a_i^* + \sum_{i=1}^{n-m} \delta_i b_i$ in V, the fact that $v_0 + v \in V$ implies that $v_0' + v' = \sum_{i=1}^{n-m} (\beta_i + \delta_i) b_i$ is in V/A_0. □

By Proposition 8.5, if V is periodic then so is V/A_0 and, hence, Theorem 8.2 can be applied to the tiling $(V/A_0, A/A_0)$. In general, this process may be iterated until we obtain a decomposition of the original tiling (V, A) into nonperiodic tilings. At each iteration, one of the two tile sets loses all its periodic points by Proposition 8.4. However, the recursion of Theorem 8.2 does not necessarily terminate after two iterations, since the other tile set may acquire new periodic points—that is, V/A_0 can be periodic even if the original set V is not. Such a situation is illustrated in the following example.

Example. Let $V = \mathcal{B}_7(\mathbf{0}, 1)$, and let A be the linear Hamming code generated by

$$\{a_1, a_2, a_3, a_4\} = \left\{ \begin{array}{cccc} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{array} \right\}.$$

Take, for instance, $A_0 = \langle a_1, a_2, a_3 \rangle$. An appropriate basis for $\mathbb{F}_2^7$ is $a_1, a_2, a_3, u_1, u_2, u_3, u_4$. Then, upon puncturing in the last three coordinates,

$$V/A_0 = \left\{ \begin{array}{cccccccc} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right\}. \tag{18}$$

Note that V/A_0 in (18) is periodic (with $(1000)^t$ being the unique nonzero periodic point), even though $V = \mathcal{B}_7(\mathbf{0}, 1)$ is not.

Using the iterative decomposition described above, we can deduce the existence of nonperiodic full-rank tilings. First, note that

$$\operatorname{rank}(A) - m \;\le\; \operatorname{rank}(A/A_0) \;\le\; n - m, \tag{19}$$

$$\operatorname{rank}(V) - m \;\le\; \operatorname{rank}(V/A_0) \;\le\; n - m, \tag{20}$$

where $m = \dim A_0$. The inequalities (19), (20) show that if (V, A) is a full-rank (proper) tiling of $\mathbb{F}_2^n$, then $(V/A_0, A/A_0)$ is a full-rank (proper) tiling of $\mathbb{F}_2^n/A_0$. Thus, starting with any full-rank tiling (V, A), iterative application of Theorem 8.2 produces a decomposition of (V, A) into nonperiodic full-rank tilings. In fact, if $V = \mathcal{B}_{15}(\mathbf{0}, 1)$ and $A = C$, where C is a full-rank perfect code of length 15, we need only one iteration. This is so because $|\mathcal{B}_{15}(\mathbf{0}, 1)| = 16$ and by Corollary 7.3 full-rank tilings (V, A) do not exist for $|V| \le 8$. Thus, if C_0 is the set of all the periodic points of C, then the tiling $(\mathcal{B}_{15}(\mathbf{0}, 1)/C_0, C/C_0)$ is full rank and nonperiodic.

We conclude this section by observing that as a consequence of Theorems 6.2 and 8.2, the classification of tiles reduces to the classification of nonperiodic full-rank tiles. That is, if we can determine whether any given nonperiodic full-rank set is a tile, then by using Theorems 6.2 and 8.2 we can determine whether any given set is a tile. The next section provides a means for showing that certain sets cannot be tiles.

9. Nonexistence results: Generalized Lloyd theorem. In this section we derive several necessary conditions for the existence of tilings. Our main result herein is a generalization of the Lloyd theorem [21], originally stated for tilings with Hamming spheres, for arbitrary tiles. The Lloyd theorem plays a crucial role in the classification of perfect binary codes [18, 34]; for more details see [20, Chap. 7]. Other generalizations of the Lloyd theorem may be found in [19].

As in [22], we represent a vector $v = (v_1, v_2, \ldots, v_n) \in \mathbb{F}_2^n$ by its image $z^v = z_1^{v_1} z_2^{v_2} \cdots z_n^{v_n}$ in the group algebra QG, where G is isomorphic to $\mathbb{F}_2^n$. Then for any $u, v \in \mathbb{F}_2^n$ and $V \subset \mathbb{F}_2^n$, the characters $\chi_u(\cdot)$ are defined in the standard way (cf. [22, Chap. 5]),

$$\chi_u(z^v) \;=\; (-1)^{\langle u,v\rangle},$$

$$\chi_u(V) \;=\; \sum_{v\in V} \chi_u(z^v),$$

where $\langle \cdot, \cdot \rangle$ stands for the inner product modulo 2 in $\mathbb{F}_2^n$. The Krawtchouk polynomial of degree k in the indeterminate x is defined by

$$P_k(x) \;=\; \sum_{i=0}^{k} (-1)^i \binom{x}{i} \binom{n-x}{k-i},$$

for $k = 0, 1, \ldots, n$. The distance distribution of a subset $A \subset \mathbb{F}_2^n$ is given by

$$D_i(A) = \frac{1}{|A|} \sum_{a\in A} W_i(a + A),$$

where $W_i(a + A)$ is the number of vectors of weight i in $a + A$. The MacWilliams transform of $D_i(A)$ is given by

$$D_i'(A) \;=\; \frac{1}{|A|} \sum_{j=0}^{n} D_j(A) P_i(j).$$

Now let (V, A) be a tiling of $\mathbb{F}_2^n$. Define the sets $N(A), N'(A) \subset \{0, 1, \ldots, n\}$ as follows:

$$N(A) = \{\, j \,:\, D_j(A) \neq 0 \,\},$$
$$N'(A) = \{\, j \,:\, D'_j(A) \neq 0 \,\}.$$

Thus, $N(A)$ is the set of distances occurring in A while $N'(A)$ is the set of distances occurring in the formal dual $A' = \{a : \chi_a(A) \neq 0\}$ of A. Further, let

$$U = \{\, u \,:\, \chi_u(V) = 0 \,\},$$
$$Q(U) = \{\, j \,:\, \exists u \in U \text{ such that } wt(u) = j \,\}.$$

THEOREM 9.1. $N'(A) \subseteq Q(U) \cup \{0\}$.

Proof. Since $A + V = \mathbb{F}_2^n$, in the group algebra QG we have $\chi_u(V)\,\chi_u(A) = \chi_u(\mathbb{F}_2^n)$. Since $\chi_u(\mathbb{F}_2^n) = 0$ for all $u \neq \mathbf{0}$, either $\chi_u(V) = 0$ or $\chi_u(A) = 0$ or both $\chi_u(V) = \chi_u(A) = 0$, unless $u = \mathbf{0}$. Now, consider the following sum [22, p. 139]:

$$S_j(A) \stackrel{\text{def}}{=} \frac{1}{|A|^2} \sum_{wt(u)=j} \chi_u(A + A) = \frac{1}{|A|^2} \sum_{wt(u)=j} |\chi_u(A)|^2 \geq 0. \tag{21}$$

Furthermore, since $\chi_u(A + A) = \sum_{i \in N(A)} \sum_{\substack{wt(w)=i \\ w \in A+A}} \chi_u(z^w)$, we have

$$S_j(A) = \frac{1}{|A|^2} \sum_{wt(u)=j} \sum_{i \in N(A)} \sum_{\substack{wt(w)=i \\ w \in A+A}} \chi_u(z^w) = \frac{1}{|A|} \sum_{i \in N(A)} D_i(A) \sum_{wt(u)=j} \chi_u(z^x).$$

Here x is an arbitrary vector of weight i, and we have used the fact that $\sum_{wt(u)=j} \chi_u(z^x)$ depends only on the weight of x. In fact [22, p. 135], $\sum_{wt(u)=j} \chi_u(z^x) = P_j(i)$, and therefore

$$S_j(A) = \frac{1}{|A|} \sum_{i \in N(A)} D_i(A) P_j(i) = D'_j(A). \tag{22}$$

Now let $j \in \{1, 2, \ldots, n\}$. Clearly, $j \in N'(A)$ if and only if $D'_j(A) \neq 0$. From (21) and (22) it follows that if $D'_j(A) \neq 0$, then there exists a vector u of weight j such that $\chi_u(A) \neq 0$. Since $\chi_u(V)\,\chi_u(A) = 0$ unless $u = \mathbf{0}$, for this vector u we must have $\chi_u(V) = 0$. Hence $u \in U$ and $j \in Q(U)$. □

If V is the Hamming sphere of radius R and $u \in \mathbb{F}_2^n$ is a vector of weight j then

$$\chi_u(V) = \sum_{v \in V} (-1)^{\langle u,v \rangle} = \sum_{i=0}^{R} \sum_{wt(v)=i} (-1)^{\langle u,v \rangle} = P_0(j) + P_1(j) + \cdots + P_R(j).$$

This is precisely the Lloyd polynomial $L_R(x)$ evaluated at $x = j$. Thus, in this case $Q(U)$ is just the set of integer zeros of $L_R(x)$, lying between 1 and n. Furthermore, if A is a perfect binary code then $|N'(A)| = R + 1$, which follows, for instance, from Proposition 9.4 below. Thus, for tilings with Hamming spheres, Theorem 9.1 implies that the Lloyd polynomial $L_R(x)$ has at least R distinct zeros in the set $\{1, 2, \ldots, n\}$. This, together with $\deg L_R(x) \leq R$, is precisely the Lloyd theorem.

The following three propositions give several additional necessary conditions for the existence of a tiling (V, A).

PROPOSITION 9.2. $A' \subseteq U \cup \{\mathbf{0}\}$.

Proof. Recall that $A' = \{a : \chi_a(A) \neq 0\}$. The proposition now follows from the fact that $\chi_u(V)\,\chi_u(A) = 0$ for $u \neq \mathbf{0}$. □

PROPOSITION 9.3. $|U| \geq |V| - 1$.

Proof. Evidently, $\max_{u \in \mathbb{F}_2^n} |\chi_u(A)| = |A|$. Thus, taking into account that $\chi_u(A+A) = |\chi_u(A)|^2 = 0$ for $u \notin U \cup \{\mathbf{0}\}$, we have

$$\frac{1}{|A|^2} \sum_{u \in \mathbb{F}_2^n} \chi_u(A+A) \;=\; \frac{1}{|A|^2} \sum_{u \in U \cup \{\mathbf{0}\}} |\chi_u(A)|^2 \;\leq\; \frac{|U|+1}{|A|^2} \max_{u \in \mathbb{F}_2^n} |\chi_u(A)|^2 \;\leq\; |U|+1.$$

On the other hand,

$$\frac{1}{|A|^2} \sum_{u \in \mathbb{F}_2^n} \chi_u(A+A) \;=\; \frac{1}{|A|^2} \sum_{j=0}^{n} \sum_{wt(u)=j} \chi_u(A+A) \;=\; \sum_{j=0}^{n} D'_j(A),$$

where the second equality follows from (21) and (22). But, since $D'_j(A)$ is the MacWilliams transform of the distance distribution of A, we have $\sum_{j=0}^{n} D'_j(A) = 2^n / |A| = |V|$. □

PROPOSITION 9.4. *$|Q(U)| \geq R$, where R is the smallest integer such that $\sum_{i=0}^{R} \binom{n}{i} \geq |V|$.*

Proof. Let $\rho = \max_{b \in \mathbb{F}_2^n} \min_{a \in A} d(b,a)$ be the covering radius of A. Note that $|N'(A)| - 1$ is the number of nonzero weights in the formal dual of A. Therefore

$$|Q(U)| \;\geq\; |N'(A)| - 1 \;\geq\; \rho$$

where the first inequality follows from Theorem 9.1 and the second from the results of [5]. By the sphere-covering bound we have

$$|A| \sum_{i=0}^{\rho} \binom{n}{i} \;\geq\; 2^n.$$

This, together with $|A| = 2^n / |V|$, implies that $|V| \leq \sum_{i=0}^{\rho} \binom{n}{i}$. □

Example. Let $V \subset \mathbb{F}_2^{29}$ be the union of $\mathcal{B}_{29}(\mathbf{0}, 3)$ and some six arbitrarily chosen vectors of weight ≥ 4. We will use Proposition 9.4 to show that V cannot be a tile. Note that $|V| = 2^{12}$ and therefore $R = 4$. Let $u \in \mathbb{F}_2^{29}$ and let x denote the weight of u. For the set V at hand, we obviously have $|\chi_u(V) - L_3(x)| \leq 6$, where $L_3(x) = \chi_u(\mathcal{B}_{29}(\mathbf{0}, 3)) = P_0(x) + P_1(x) + P_2(x) + P_3(x)$ is the Lloyd polynomial. Thus, $\chi_u(V)$ may vanish only if

$$|L_3(x)| \;=\; \left| 4090 - \frac{2618}{3}\, x + 60\, x^2 - \frac{4}{3}\, x^3 \right| \;\leq\; 6.$$

Direct calculation shows that only $x = 15$ satisfies this inequality; hence $|Q(U)| \leq 1$. In fact, $|Q(U)| = 1$ in this case. However, Proposition 9.4 requires $|Q(U)| \geq R = 4$, a contradiction.

Using the known bounds on the size of linear codes [37], it is often possible to show that certain sets cannot be linear tiles. For $D \subset \{1, 2, \ldots, n\}$, let $M(D)$ denote the maximum number of codewords in a linear code C, whose weight spectrum (that is, the set of all i for which $D_i(C) \neq 0$) belongs to $D \cup \{0\}$.

PROPOSITION 9.5. *If V is a linear tile then $|V| \leq M\big(Q(U)\big)$.*

Proof. Recall that V is a linear tile if there exists a tiling (V, A) with A being linear. If A is linear then $A' = \{a : \chi_a(A) \neq 0\} = A^\perp$ is a linear code of size $2^n/|A| = |V|$. Further, $A' \subset U \cup \{\mathbf{0}\}$ by Proposition 9.2. Therefore

$$|V| = |A'| \leq M\big(Q(U)\big). \qquad \square$$

Example. Let $n = 32$ and let $V = \mathcal{B}_{32}(\mathbf{0}, 2) \setminus \mathcal{X}$, where $\mathcal{X}$ consists of some 17 arbitrary vectors of weight 2. Note that $|V| = 2^9$. Reasoning as in the preceding example, we conclude that a vector $u \in \mathbb{F}_2^{32}$ of weight x may belong to U only if $|L_2(x)| = |P_0(x) + P_1(x) + P_2(x)| \leq 17$. This inequality holds for $x = 13, 14, \ldots, 20$. Therefore, by Proposition 9.5, the existence of a $(32, 2^9, 13)$ linear code is a necessary condition for V to be a linear tile. But from [37] we know that such a code does not exist.

Example. Let $n = 6$ and $V = \mathcal{B}_6(\mathbf{0}, 1) \cup (111110)$. It is easy to verify that U consists of all vectors of weight 3 or 4 which contain zero in the last coordinate. So, if V is a linear tile then by Proposition 9.5 there exists a linear code of length 5 and size at least $|V| = 8$, with nonzero weights 3 and 4. This is impossible, and therefore V cannot be a linear tile. However, by Proposition 6.1 and Corollary 7.3, any tile of size ≤ 8 is also a linear tile. Hence, we conclude that $\mathcal{B}_6(\mathbf{0}, 1) \cup (111110)$ is not a tile. A similar argument applies to $\mathcal{B}_6(\mathbf{0}, 1) \cup (111100)$. Thus, we have an alternative proof of the "only if" part in Claim 4.6.

Acknowledgment. We are indebted to Noga Alon and Tuvi Etzion for helpful discussions.

REFERENCES

[1] H. Bauer, B. Ganter, and F. Hergert, *Algebraic techniques for nonlinear codes*, Combinatorica, 3 (1983), pp. 21–33.

[2] A. Blokhuis and C. W. H. Lam, *More coverings by rook domains*, J. Combin. Theory Ser. A, 36 (1984), pp. 240–244.

[3] N. G. de Bruijn, *On the factorization of finite abelian groups*, Indag. Math. (N.S.), 15 (1953), pp. 258–264.

[4] G. D. Cohen and P. Frankl, *On tilings of the binary vector space*, Discrete Math., 31 (1980), pp. 271–277.

[5] P. Delsarte, *Four fundamental parameters of a code and their combinatorial significance*, Inform. and Control, 23 (1973), pp. 407–438.

[6] M. Deza, *The effectiveness of noise correction or detection*, Problems of Inf. Trans., 1 (1965), pp. 29–39.

[7] M. Deza and F. Hoffman, *Some results related to generalized Varshamov–Gilbert bound*, IEEE Trans. Inform. Theory, 4 (1977), pp. 517–518.

[8] M. Deza, M. Karpovsky, and V. Milman, *Codes correcting an arbitrary set of errors*, Revue du Cethedec, 66 (1981), pp. 65–76.

[9] A. W. M. Dress and R. Scharlau, *The 37 combinatorial types of minimal, nontransitive, equivariant tilings of the Euclidean plane*, Discrete Math., 60 (1986), pp. 121–138.

[10] T. Etzion and A. Vardy, *Perfect binary codes: Constructions, properties, and enumeration*, IEEE Trans. Inform. Theory, 40 (1994), pp. 754–763.

[11] L. Fejes Tóth, *Legerrungen in der Ebene, auf der Kugel und in Raum*, 2nd ed., Springer-Verlag, New York, 1972.

[12] S. W. Golomb, *Polyominoes*, 2nd ed., Princeton University Press, Princeton, NJ, 1994.

[13] G. Hajós, *Sur la factorisation des groupes abéliens*, Časopis Pěst Math. Rys., 74 (1949), pp. 157–162.

[14] T. W. Hungerford, *Algebra*, Holt, Rinehart and Winston, New York, 1974.

[15] M. Karpovsky, *Weight distribution of translates, covering radius and perfect codes correcting errors of given multiplicities*, IEEE Trans. Inform. Theory, 27 (1981), pp. 462–472.

[16] M. Karpovsky and V. Milman, *Coordinate density of sets of vectors*, Discrete Math., 24 (1978), pp. 177–184.

[17] ———, *On subspaces contained in subsets of finite homogeneous spaces*, Discrete Math., 22 (1978), pp. 273–280.
[18] J. H. VAN LINT, *Nonexistence theorems for perfect error-correcting-codes*, in Computers in Algebra and Number Theory, Vol. 4, SIAM-AMS Proceedings, Philadelphia, PA, 1971.
[19] ———, *A survey of perfect codes*, Rocky Mountain J. Math., 5 (1975), pp. 199–224.
[20] ———, *Introduction to Coding Theory*, Springer-Verlag, New York, 1992.
[21] S. P. LLOYD, *Binary block coding*, Bell Syst. Tech. J., 36 (1957), pp. 517–535.
[22] F. J. MACWILLIAMS AND N. J. A. SLOANE, *The Theory of Error-Correcting Codes*, North-Holland, New York, 1977.
[23] M. MOLLARD, *A generalized parity function and its use in the construction of perfect codes*, SIAM J. Alg. Disc. Meth., 7 (1986), pp. 113–115.
[24] R. PENROSE, *Pentaplexity: A class of nonperiodic tilings of the plane*, Math. Intelligencer, 2 (1979), pp. 32–37.
[25] K. T. PHELPS, *A combinatorial construction of perfect codes*, SIAM J. Alg. Disc. Meth., 4 (1983), pp. 398–403.
[26] ———, *A general product construction for error-correcting codes*, SIAM J. Alg. Disc. Meth., 5 (1984), pp. 224–228.
[27] K. T. PHELPS AND M. LEVAN, *Kernels of nonlinear Hamming codes*, Des. Codes Cryptogr., 6 (1995), pp. 247–257.
[28] L. RÉDEI, *Lacunary Polynomials Over Finite Fields*, North-Holland, New York, 1973.
[29] C. A. ROGERS, *Packing and Covering*, Cambridge University Press, London, 1964.
[30] A. D. SANDS, *On the factorization of finite abelian groups* II, Acta Math. Sci., 13 (1962), pp. 153–169.
[31] A. SCHRIJVER, *Theory of Linear and Integer Programming*, Wiley, New York, 1986.
[32] S. K. STEIN, *Tiling space by congruent polyhedra*, Bull. Amer. Math. Soc., 80 (1974), pp. 819–820.
[33] N. L. TAN, L. R. WELCH, AND R. A. SCHOLTZ, *Correcting a specified set of likely error patterns*, IEEE Trans. Inform. Theory, 41 (1995), pp. 272–279.
[34] A. TIETÄVÄINEN AND A. PERKO, *There are no unknown perfect binary codes*, Ann. Univ. Turku., Ser. AI, 148 (1971), pp. 3–10.
[35] A. VARDY AND Y. BE'ERY, *Maximum-likelihood soft decision decoding of BCH codes*, IEEE Trans. Inform. Theory, 40 (1994), pp. 546–554.
[36] J. L. VASILIEV, *On nongroup close-packed codes*, Problemi. Tekhn. Kibernet. Robot., 8 (1962), pp. 375–378.
[37] T. VERHOEFF, *An updated table of minimum-distance bounds for binary linear codes*, IEEE Trans. Inform. Theory, 33 (1987), pp. 665–680.
[38] G. ZÉMOR, *Subset sums in binary spaces*, European J. Combin., 13 (1992), pp. 221–230.

SIAM J. DISCRETE MATH.
Vol. 9, No. 3, pp. 413–423, August 1996

006

NONEQUIVALENT q-ARY PERFECT CODES*

TUVI ETZION†

Abstract. We construct a set of $q^{q^{cn}}$ nonequivalent q-ary perfect single error-correcting codes of length n over $GF(q)$ for sufficiently large n and a constant $c = \frac{1}{q} - \epsilon$. The construction is based on a small subcode A of the q-ary Hamming code of length n for which A and $q-1$ of its cosets $A_1, \ldots, A_{q-1}$ cover the same subset V. We show a few isomorphic and nonisomorphic ways in which A can be chosen, and we prove the uniqueness of these ways to choose A.

Key words. Hamming codes, isomorphism, nonequivalent codes, perfect codes

AMS subject classifications. 94B, 05B

1. Introduction. Let F_q^n be a vector space of dimension n over $GF(q)$. A subset of F_q^n is a q-ary code of length n. Two codes $C_1, C_2 \subset F_q^n$, are said to be *isomorphic* if there exists a permutation π such that $C_2 = \{\pi(c) : c \in C_1\}$. They are said to be *equivalent* if there exists a vector v and a permutation π such that $C_2 = \{v + \pi(c) : c \in C_1\}$. The Hamming *distance* between vectors $u, v \in F_q^n$, denoted $d(u, v)$, is the number of coordinates in which u and v differ. Without loss of generality (w.l.o.g.), we shall assume, unless stated otherwise, that the all-zero vector is in C.

A code C of length n is *perfect* if for some integer $r \geq 0$ every $x \in F_q^n$ is within distance r from exactly one codeword of C. The study of perfect codes is one of the most fascinating subjects in coding theory. It is well known [3] that the only parameters for nontrivial perfect codes are those of the two Golay codes and the Hamming codes. The Hamming codes have length $n_m = \frac{q^m-1}{q-1}$, $m \geq 2$, and $r = 1$. They are single error-correcting codes, and henceforth when we use the words "perfect code" we will refer to codes with $r = 1$ and length n_m. The Hamming codes are the only linear codes with these parameters [3, p. 77]. For $q = 2$, constructions for nonequivalent perfect codes were presented by Phelps [5], [6], Etzion and Vardy [1], and others. For other q's, constructions of nonlinear codes were presented by Schönheim [9], Lindström [2], and Mollard [4]. Nonequivalent perfect codes were generated by Phelps [7]. The construction for $q = 2$ given in Etzion and Vardy [1] has the advantage of obtaining the largest known set of nonequivalent perfect codes. It is also possible to obtain from the construction of [1] perfect codes with other properties as different ranks [1] and different kernels [8]. In this paper we generalize the construction of Etzion and Vardy [1] to any alphabet of size q, where q is a power of a prime.

In §2 we present a construction of a set of $q^{q^{\frac{n_m-1}{q}+1-\log_q(n_m(q-1)+1)}}$ distinct perfect codes of length n. This set contains at least $q^{q^{\frac{n_m-1}{q}+1-\log_q(n_m(q-1)+1)}-n_m(1+\log_q n_m)}$ nonequivalent codes. This is the largest known set of nonequivalent perfect codes. The construction is based on a small subcode A of the q-ary Hamming code for which A and $q-1$ of its cosets $A_1, \ldots, A_{q-1}$ cover the same subset V. This set A is important

* Received by the editors December 5, 1994; accepted for publication (in revised form) August 25, 1995. This research was supported in part by the EPSRC of the United Kingdom under grant GR/K38847.

† Computer Science Department, Royal Holloway, University of London, Egham, Surrey TW20 0EX, United Kingdom (etzion@cs.technion.ac.il). The author is on leave of absence from the Computer Science Department, Technion - Israel Institute of Technology, Haifa 32000, Israel.

in constructions of perfect codes with different ranks [1] and different kernels [8]. In §3 we show a few isomorphic and nonisomorphic ways in which this set A can be chosen, and we prove the uniqueness of the ways in which A is chosen.

2. Construction for nonequivalent perfect codes. The parity check matrix of the Hamming code of length $n_{m+1} = \frac{q^{m+1}-1}{q-1} = q^m + q^{m-1} + \cdots + q + 1$, over $GF(q)$, consists of n_{m+1} pairwise linear independent column vectors of length $m+1$ over $GF(q)$. We will take the n_{m+1} column vectors of the form $(0 \cdots 01x_1 \cdots x_r)^T$ for all $0 \le r \le m$, where for $x_i \in GF(q)$, $1 \le i \le r$. Let α be a primitive element in $GF(q)$. Then the $2 \times (q+1)$ parity check matrix of the Hamming code of length n_2 has the form

$$H_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 0 & \alpha^0 & \alpha^1 & \cdots & \alpha^{q-2} \end{bmatrix},$$

and its $(q-1) \times (q+1)$ generator matrix has the form

$$G_2 = \begin{bmatrix} 1 & \alpha^{q-1} & -\alpha^{q-1} & 0 & \cdots & 0 \\ 1 & \alpha^{q-2} & 0 & -\alpha^{q-2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha & 0 & 0 & \cdots & -\alpha \end{bmatrix} = \begin{bmatrix} 1 & & \\ \vdots & & X \\ 1 & & \end{bmatrix},$$

where $\boldsymbol{X}$ is a $(q-1) \times q$ matrix.

Assume the Hamming code of length $n_m = \frac{q^m-1}{q-1} = q^{m-1} + \cdots + q + 1$ with $m \ge 2$ has the $m \times n_m$ parity check matrix H_m of the form

$$H_m = \begin{bmatrix} 0 & & & & \\ \vdots & S_1 & S_2 & \cdots & S_{n_m-1} \\ 0 & & & & \\ 1 & & & & \end{bmatrix},$$

where the S_i's are column vectors.

Also assume that the $(n_m - m) \times n_m$ generator matrix G_m has the form

$$G_m = \left[\begin{array}{ccccccc} 1 & & & & & & \\ \vdots & X & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & \\ 1 & & & & & & \\ \vdots & 0 & X & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & \\ \vdots & \vdots & \vdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \cdots & \cdots & \vdots \\ 1 & & & & & & \\ \vdots & 0 & 0 & \cdots & \cdots & \cdots & X \\ 1 & & & & & & \\ \hline & & & F & & & \end{array}\right],$$

where $\boldsymbol{F}$ is a $t \times n_m$ matrix with $t = (n_m - m) - (q-1)\frac{n_m - 1}{q} = n_{m-1} + 1 - m$.

Now, we generate the following $(m+1) \times (n_m q + 1)$ parity check matrix H_{m+1}:

$$H_{m+1} = \begin{bmatrix} 0 & & & & & & & & & & 0 & 0 & \cdots & 0 \\ \vdots & S_1 & S_1 & \cdots & S_1 & \cdots & S_{n_m-1} & S_{n_m-1} & \cdots & S_{n_m-1} & \vdots & \vdots & \cdots & \vdots \\ 0 & & & & & & & & & & 0 & 0 & \cdots & 0 \\ 0 & & & & & & & & & & 1 & 1 & \cdots & 1 \\ 1 & 0 & \alpha^0 & \cdots & \alpha^{q-2} & \cdots & 0 & \alpha^0 & \cdots & \alpha^{q-2} & 0 & \alpha^0 & \cdots & \alpha^{q-2} \end{bmatrix}.$$

LEMMA 2.1. *H_{m+1} is a parity check matrix of the Hamming code of length $n_m q + 1$.*

Proof. Assume H_m has all the n column vectors of length m of the form $(0 \cdots 01 x_1 \cdots x_r)^T$ for all r, $0 \le r \le m-1$, where $x_i \in GF(q)$, $1 \le i \le r$. By induction starting from the basis H_2, we can easily prove that H_{m+1} is a parity check matrix of the Hamming code of length n_{m+1}. □

Now, let G_{m+1} be the following $(n_m q - m) \times (n_m q + 1)$ matrix:

$$G_{m+1} = \left[\begin{array}{cccccccc} 1 & & & & & & & \\ \vdots & X & 0 & \cdots & \cdots & \cdots & 0 & \\ 1 & & & & & & & \\ 1 & & & & & & & \\ \vdots & 0 & X & \cdots & \cdots & \cdots & 0 & \\ 1 & & & & & & & \\ \vdots & \vdots & \vdots & \cdots & \cdots & \cdots & \vdots & \\ \vdots & \vdots & \vdots & \cdots & \cdots & \cdots & \vdots & \\ \vdots & \vdots & \vdots & \cdots & \cdots & \cdots & \vdots & \\ 1 & & & & & & & \\ \vdots & 0 & 0 & \cdots & \cdots & \cdots & X & \\ 1 & & & & & & & \\ \hline & & & & F' & & & \end{array}\right],$$

where $\boldsymbol{F}'$ is some $t' \times (n_m q + 1)$ matrix; $t' = n_m - m$.

LEMMA 2.2. *The generator matrix of the Hamming code with parity check matrix H_{m+1} has the form of G_{m+1}.*

Proof. From the form of H_2 and G_2 it follows that the matrix

$$\begin{bmatrix} 0 & S & S & S & \cdots & S \\ 1 & 0 & \alpha^0 & \alpha^1 & \cdots & \alpha^{q-2} \end{bmatrix},$$

for any given column vector S, is orthogonal to the matrix

$$\begin{bmatrix} 1 & \\ \vdots & X \\ 1 & \end{bmatrix}.$$

Taking this fact into account the claim follows directly from the form of H_{m+1} and G_{m+1}. □

We say that a vector v covers the set U if for any $u \in U$ we have $d(v,u) \leq 1$. A code C covers a set U if for every element $u \in U$ there exists a codeword $c \in C$ such that $d(c,u) \leq 1$. Let $C(G)$ denote the code generated by a generator matrix G.

LEMMA 2.3. *If G^1_{m+1} is the matrix consisting of the first $n_m(q-1)$ rows of G_{m+1}, then $C(G^1_{m+1})$ and $(\alpha^j \vdots 0 \cdots 0) + C(G^1_{m+1})$, $j \geq 0$, cover the same subset of $F_q^{n_m q+1}$.*

Proof. Since $G_2 = G_2^1$ and $C(G_2)$ is a perfect code, it follows that its coset $(\alpha^j \vdots 0 \cdots 0) + C(G_2)$ is also a perfect code and thus $C(G_2^1)$ and $(\alpha^j \vdots 0 \cdots 0) + C(G_2^1)$ cover the same subset of F_q^{q+1}. Let $v = (\gamma \vdots u_1, \ldots, u_{n_m}) \in C(G^1_{m+1})$, where $(\delta_i \vdots u_i) \in C(G_2)$, $u_i \in F_q^q$, $\delta_i \in GF(q)$, and $\gamma = \sum_{i=1}^{n_m} \delta_i$. This is the form of codewords from $C(G^1_{m+1})$ as follows from the form of G_2 and G^1_{m+1}. We will show that every vector which is covered by v is also covered by a codeword of $(\alpha^j \vdots 0 \cdots 0) + C(G^1_{m+1})$. Obviously, $v + (\beta \vdots 0 \cdots 0)$, $\beta \in GF(q)$, is covered by $v + (\alpha^j \vdots 0 \cdots 0) \in (\alpha^j \vdots 0 \cdots 0) + C(G^1_{m+1})$. So, we only have to show that any word of the form $(\gamma \vdots u_1 \cdots u_{i-1} u_i' u_{i+1} \cdots u_{n_m})$, where u_i and u_i' differ in exactly one position, is covered by a codeword of $(\alpha^j \vdots 0 \cdots 0) + C(G^1_{m+1})$. We know that the vector $(\delta_i \vdots u_i')$ is covered by a codeword $v_i' \in (\alpha^j \vdots 0 \cdots 0) + C(G_2)$ because $(\alpha^j \vdots 0 \cdots 0) + C(G_2)$ is a perfect code. Since $(\delta_i \vdots u_i) \in C(G_2)$ and since the minimum distance of $C(G_2)$ is 3, it follows that a vector of the form $(x \vdots u_i')$ is not in $C(G_2)$ and hence also not in $(\alpha^j \vdots 0 \cdots 0) + C(G_2)$. Therefore, $v_i' = (\delta_i \vdots u_i'')$, where u_i'' differs in exactly one position from u_i' and in exactly two positions from u_i. Since $(\delta_i \vdots u_i'') \in (\alpha^j \vdots 0 \cdots 0) + C(G_2)$, it follows that $(\delta_i - \alpha^j \vdots u_i'') \in C(G_2)$ and hence $(\gamma - \alpha^j \vdots u_1 \cdots u_{i-1} u_i'' u_{i+1} \cdots u_{n_m}) \in C(G^1_{m+1})$. Hence, $(\gamma \vdots u_1 \cdots u_{i-1} u_i' u_{i+1} \cdots u_{n_m})$ is covered by $(\gamma \vdots u_1 \cdots u_{i-1} u_i'' u_{i+1} \cdots u_{n_m}) \in (\alpha^j \vdots 0 \cdots 0) + C(G^1_{m+1})$. Thus, every vector which is covered by $C(G^1_{m+1})$ is also covered by $(\alpha^j \vdots 0 \cdots 0) + C(G^1_{m+1})$ and since $C(G^1_{m+1})$ and $(\alpha^j \vdots 0 \cdots 0) + C(G^1_{m+1})$ have the same size the lemma follows. □

Now, we can write G_m as

$$G_m = \left[\begin{array}{c} G^1_m \\ \hline \boldsymbol{F} \end{array}\right], \quad \text{where} \quad \boldsymbol{F} = \begin{bmatrix} f_1 \\ \vdots \\ f_t \end{bmatrix},$$

where f_i is a $1 \times n$ matrix. Let c_j, $1 \leq j \leq q^t$, be the q^t codewords formed from $\boldsymbol{F}$. By Lemma 2.3 we have that $c_j + C(G^1_m)$ and $(\alpha^j \vdots 0 \cdots 0) + c_j + C(G^1_m)$ cover the same subset of $F_q^{n_m}$.

LEMMA 2.4. *Given the vector $(g_1, g_2, \ldots, g_{q^t})$, $g_i \in GF(q)$, $1 \leq i \leq q^t$, the code*

$$C = \bigcup_{i=1}^{q^t} ((g_i \vdots 0 \cdots 0) + c_i + C(G^1_m))$$

forms a q-ary perfect code.

Proof. If $g_i = 0$ for all i, then C is the Hamming code. The lemma now follows from the fact that $c_i + C(G^1_m)$ and $(g_i \vdots 0 \cdots 0) + c_i + C(G^1_m)$ cover the same subset of $F_q^{n_m}$. □

Let $\Omega(n_m)$ be the set of perfect codes constructed in Lemma 2.4. Obviously $|\Omega(n_m)| = q^{q^t} = q^{q^{n_{m-1}+1-m}} = q^{q^{\frac{n_m-1}{q}+1-\log_q(n_m(q-1)+1)}}$. Given a perfect code C of length n_m, there are at most $q^{n_m} n_m! \leq q^{n_m} q^{n_m \cdot \log_q n_m} = q^{n_m(1+\log_q n_m)}$ different perfect codes equivalent to C. Hence we have Theorem 2.5.

THEOREM 2.5. *$\Omega(n_m)$ contains at least $q^{q^{\frac{n_m-1}{q}+1-\log_q(n_m(q-1)+1)}-n_m(1+\log_q n_m)}$ nonequivalent perfect codes.*

A more precise enumeration will slightly improve the result of Theorem 2.5. Finally, we would like to mention that given a perfect code C one might permute symbols independently in each position to obtain another perfect code. If we consider these perfect codes as equivalent we will have that there are at most $q^{n_m}n_m!(q!)^n$ different perfect codes equivalent to C. But, this will hardly influence the result of Theorem 2.5.

3. Splitting submatrices of the Hamming code. In Lemma 2.3 we have proved that $C(G^1_{m+1})$ and $(\alpha^j\vdots 0\cdots 0)+C(G^1_{m+1})$, $j\geq 0$, cover the same subset of $F_q^{n_mq+1}$. The following question is of interest and importance. For a given i, $2\leq i\leq n_mq+1$, does there exist an $n_m(q-1)\times(n_mq+1)$ submatrix G^i_{m+1} of G_{m+1} such that $C(G^i_{m+1})$ and $\alpha^je_i+C(G^i_{m+1})$, $j\geq 0$, where $e_i=(0\cdots 010\cdots 0)$ with the 1 in position i, cover the same subset of $F_q^{n_mq+1}$? For $q=2$ these submatrices exist as proved in [1]. These subcodes together with $C(G^1_{m+1})$ were used in [1] to construct codes with various ranks and in [8] to construct codes with various kernels. This submatrix, G^i_{m+1}, $1\leq i\leq n_{m+1}$, will be called a *splitting submatrix* of G_{m+1} and these submatrices are the subject of this section.

In this section we will prove that for each i, $1\leq i\leq n_{m+1}$, a splitting submatrix G^i_{m+1} of G_{m+1} exists. We will also prove the uniqueness of these submatrices. In order to simplify the understanding of the construction for G^i_{m+1} we will permute the columns of the code such that column i will become the first column.

We start by considering the Hamming code of length $q+1$. As shown in §2, a parity check matrix of the code H_2 has the form

$$H_2=\begin{bmatrix}0&1&1&1&\cdots&1\\1&0&\alpha^0&\alpha^1&\cdots&\alpha^{q-2}\end{bmatrix},$$

the generator matrix of the code has the form

$$\begin{bmatrix}\alpha^0-0&1&-1&0&\cdots&0&0\\\alpha^1-\alpha^0&0&1&-1&\cdots&0&0\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\\alpha^{q-2}-\alpha^{q-3}&0&0&0&\cdots&1&-1\end{bmatrix},$$

and from this matrix we can immediately compute

$$G_2=\begin{bmatrix}1&\\\vdots&X\\1&\end{bmatrix}$$

as given is §2.

If $q=p$ then we can take instead of H_2 the check matrix

$$\begin{bmatrix}0&1&1&1&\cdots&1\\1&0&1&2&\cdots&p-1\end{bmatrix},$$

and its generator matrix has the form

$$\begin{bmatrix}1&1&-1&0&\cdots&0&0\\1&0&1&-1&\cdots&0&0\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\1&0&0&0&\cdots&1&-1\end{bmatrix}.$$

For a given $\beta \in GF(q)$, if the parity check matrix has the column $\binom{1}{\beta}$ as the first column then we take the parity check matrix

$$H_2^* = \begin{bmatrix} 1 & 0 & 1 & 1 & \cdots & 1 \\ \beta & 1 & \beta+\alpha^0 & \beta+\alpha^1 & \cdots & \beta+\alpha^{q-2} \end{bmatrix},$$

and its generator matrix is

$$G_2^* = \begin{bmatrix} 1 & \alpha^0 & -1 & 0 & \cdots & 0 \\ 1 & \alpha^1 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{q-2} & 0 & 0 & \cdots & -1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ \vdots & & \boldsymbol{Y} \\ 1 & & \end{bmatrix},$$

where $\boldsymbol{Y}$ is a $(q-1) \times q$ matrix.

The proof that H_2^* is a parity check matrix of the Hamming code is based on the following simple lemma.

LEMMA 3.1. *If β is an element in $GF(q)$ then the set of elements $\{\beta+\alpha^i : 0 \leq i \leq q-2\}$ consists of all the elements of $GF(q)$.*

We will make extensive use of this lemma in our constructions to prove that the parity check matrix of the codes, which we will construct, has all the columns of length $m+1$ and the form $(0 \cdots 01x_1 \cdots x_r)^T$ for all r, $0 \leq r \leq m$, where $x_i \in GF(q)$, $1 \leq i \leq r$.

Now, we want to form the Hamming code of length n_{m+1}, which has as the first column in the parity check matrix the column vector of length $m+1$, $(0 \cdots 01a_1 \cdots a_r)^T$ for some r, $0 \leq r \leq m$, and some a_i's, $a_i \in GF(q)$, $1 \leq i \leq r$. We start with the parity check matrix H_{m+1-r} and the splitting submatrix G^1_{m+1-r} given in §2. For $r = m$ we take $H_1 = [1]$ and G_1^1 is an empty matrix. Now, let $H^*_{m+1-r} = H_{m+1-r}$ and $G^*_{m+1-r} = G^1_{m+1-r}$. Assume we have constructed the $i \times n_i$ parity check matrix H_i^*, $i \geq m+1-r$, where $S = (0 \cdots 01a_1 \cdots a_{i-m-1+r})^T$ is the first column of H_i^*, and the $(q^{i-1}-1) \times n_i$ splitting submatrix G_i^* which is a submatrix of the generator matrix of the Hamming code with the parity check matrix H_i^*. Assume further that G_i^* has the form

$$G_i^* = \begin{bmatrix} 1 & & & & & & \\ \vdots & \boldsymbol{Z}_1 & \boldsymbol{0} & \cdots & \cdots & \cdots & \boldsymbol{0} \\ 1 & & & & & & \\ 1 & & & & & & \\ \vdots & \boldsymbol{0} & \boldsymbol{Z}_2 & \cdots & \cdots & \cdots & \boldsymbol{0} \\ 1 & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & & & & & & \\ \vdots & \boldsymbol{0} & \boldsymbol{0} & \cdots & \cdots & \cdots & \boldsymbol{Z}_{n_{i-1}} \\ 1 & & & & & & \end{bmatrix},$$

where $\boldsymbol{Z}_i$ is either the $(q-1) \times q$ matrix $\boldsymbol{X}$ given in §2 or the $(q-1) \times q$ matrix $\boldsymbol{Y}$ given in this section. W.l.o.g. we will further assume that there exists an integer l,

$0 \le l \le n_{i-1}$, such that for each j, $1 \le j \le l$, $\boldsymbol{Z}_j = \boldsymbol{X}$, and for each j, $l+1 \le j \le n_{i-1}$, $\boldsymbol{Z}_j = \boldsymbol{Y}$. Now, assume H_i^* has the form

$$H_i^* = \begin{bmatrix} S & T_1 & T_2 & \cdots & T_{n_{i-1}} \end{bmatrix},$$

where T_j, $1 \le j \le n_{i-1}$ is an $i \times q$ matrix.

We distinguish between two cases.

Case 1. If $a_{i-m+r} = 0$, we generate the following $(i+1) \times n_{i+1}$ parity check matrix H_{i+1}^*:

$$H_{i+1}^* = \begin{bmatrix} S & \cdots & T_j & T_j & \cdots & T_j & \cdots & T_k & T_k & \cdots & T_k & \cdots & 0 & & & & \\ & & & & & & & & & & & & \vdots & S & S & \cdots & S \\ 0 & \cdots & 0 \cdot L & \alpha^0 \cdot L & \cdots & \alpha^{q-2} \cdot L & \cdots & 0 \cdot K & \alpha^0 \cdot K & \cdots & \alpha^{q-2} \cdot K & \cdots & 0 & & & & \\ & & & & & & & & & & & & 1 & \alpha^0 & \alpha^1 & \cdots & \alpha^{q-2} \end{bmatrix},$$

where $1 \le j \le l$, $l+1 \le k \le n_{i-1}$, L is an $1 \times q$ matrix of the form $L = [1 \cdots 1]$, and K is a $1 \times q$ matrix of the form $K = [\alpha^0\ \alpha^0\ \alpha^1\ \cdots\ \alpha^{q-2}]$.

Case 2. If $a_{i-m+r} = \alpha^c$, we generate the following $(i+1) \times n_{i+1}$ parity check matrix H_{i+1}^*:

$$H_{i+1}^* = \begin{bmatrix} S & \cdots & T_j & T_j & \cdots & T_j & \cdots & T_k & T_k & \cdots & T_k & 0 & & & & \\ & & & & & & & & & & & \vdots & S & S & \cdots & S \\ \alpha^c & \cdots & M+0 & M+\alpha^0 & \cdots & M+\alpha^{q-2} & \cdots & N+0 \cdot K & N+\alpha^0 \cdot K & \cdots & N+\alpha^{q-2} \cdot K & 0 & & & & \\ & & & & & & & & & & & 1 & \alpha^c+\alpha^0 & \alpha^c+\alpha^1 & \cdots & \alpha^c+\alpha^{q-2} \end{bmatrix},$$

where $1 \le j \le l$, $l+1 \le k \le n_{i-1}$, M is an $1 \times q$ matrix of the form $M = [0\ \alpha^c\ \alpha^{c+1}\ \cdots\ \alpha^{c+q-2}]$, and N is a $1 \times q$ matrix of the form $N = [0\ \alpha^c\ \cdots\ \alpha^c]$.

LEMMA 3.2. *H_{i+1}^* is a parity check matrix for the Hamming code.*

Proof. This is an immediate consequence from the fact that H_i^* is a parity check matrix of the Hamming code, Lemma 3.1, and the observation that in the four $q \times q$ matrices,

$$\begin{bmatrix} 0 \cdot L \\ \alpha^0 \cdot L \\ \vdots \\ \alpha^{q-2} \cdot L \end{bmatrix}, \begin{bmatrix} 0 \cdot K \\ \alpha^0 \cdot K \\ \vdots \\ \alpha^{q-2} \cdot K \end{bmatrix}, \begin{bmatrix} M+0 \\ M+\alpha^0 \\ \vdots \\ M+\alpha^{q-2} \end{bmatrix}, \text{ and } \begin{bmatrix} N+0 \cdot K \\ N+\alpha^0 \cdot K \\ \vdots \\ N+\alpha^{q-2} \cdot K \end{bmatrix},$$

each column is a permutation of the elements of $GF(q)$. □

Note that the definition of H_{i+1}^* coincides with the definition of H_2^* given in §2 and hence there is no ambiguity. Now, we are in a stage to produce G_{i+1}^*.

LEMMA 3.3. *The* $(q^i - 1) \times n_{i+1}$ *matrix*

$$G_{i+1}^* = \begin{bmatrix} 1 & & & & & & \\ \vdots & Z_1 & 0 & \dots & \dots & \dots & 0 \\ 1 & & & & & & \\ 1 & & & & & & \\ \vdots & 0 & Z_2 & \dots & \dots & \dots & 0 \\ 1 & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & & & & & & \\ \vdots & 0 & 0 & \dots & \dots & \dots & Z_{n_i} \\ 1 & & & & & & \end{bmatrix}$$

is a splitting submatrix of the generator matrix of the Hamming code which is orthogonal to H_{i+1}^*. *Moreover, for each* j, $1 \le j \le lq$, $\boldsymbol{Z}_i = \boldsymbol{X}$ *and for each* j, $lq+1 \le j \le n_i$, $\boldsymbol{Z}_j = \boldsymbol{Y}$.

Proof. Lemma 3.3 follows immediately after a careful analysis of the structure of G_i^*, H_i^*, H_{i+1}^*, and G_{i+1}^* together with the structure of H_2, G_2, H_2^*, and G_2^*. □

An immediate consequence is Theorem 3.4.

THEOREM 3.4. *For each* i, $2 \le i \le n_m q + 1$, *there exists a splitting submatrix* G_{m+1}^i *of the Hamming code of length* n_{m+1}.

Similar lemmas and theorems such as Lemmas 2.3 and 2.4 and Theorem 2.5 can be obtained by using the splitting submatrix G_{m+1}^i.

Given the first column of the parity check matrix of the Hamming code, an interesting question is whether there exist different splitting submatrices for the code by proper rearrangement of the other columns. If $q = 2$ then $G_2 = G_2^* = [1\ 1\ 1]$ and all the splitting submatrices are isomorphic. This structure was used to construct perfect codes with different ranks in [1] and different kernels in [8]. Although we can obtain similar results from different splitting submatrices (isomorphic splitting submatrices were not vital in those constructions), it is of interest to examine if there are nonisomorphic splitting submatrices given the first column of the parity check matrix of the Hamming code. Now, we will prove that the splitting submatrix is unique, given the first column of the parity check matrix.

We start by examining the vectors, starting with either 0 or 1, which are orthogonal to the rows of G_2 and G_2^*. From the structure of H_2 we have that the vectors starting with either 0 or 1, which are orthogonal to the rows of G_2, have the form

$$(1) \qquad (\ 0 \vdots \beta,\ \dots,\ \beta\ ), \quad \beta \in GF(q),$$

$$(2) \qquad (\ 1 \vdots \beta,\ \beta + \alpha^0,\ \dots,\ \beta + \alpha^{q-2}\ ), \quad \beta \in GF(q).$$

From the structure of H_2^* we have that the vectors, starting with either 0 or 1, which are orthogonal to rows of G_2^*, have the form

$$(3) \qquad (\ 0 \vdots \beta,\ \beta\alpha^0,\ \dots,\ \beta\alpha^{q-2}\ ), \quad \beta \in GF(q),$$

$$(4) \qquad (\, 1 \,\vdots\, \beta,\ \beta\alpha^0 + 1,\ \ldots,\ \beta\alpha^{q-2} + 1\,), \quad \beta \in GF(q).$$

For a row $v = (\gamma \vdots u_1 u_2 \cdots u_r)$, $\gamma \in GF(q)$, $u_i \in F_q^q$, $1 \le i \le r$, of the Hamming code, we say that $(\gamma \vdots u_i)$ is a subrow of v for $1 \le i \le r$. Now, assume that we are constructing the parity check matrix H of the Hamming code of length n_{m+1} which have as a first column in the parity check matrix the column vector of length $m+1$, $S = (0 \cdots 01 a_1 \cdots a_r)^T = (s_1 \cdots s_{m+1})^T$ for some r, $0 \le r \le m$, and some a_i's, $a_i \in GF(q)$, $1 \le i \le r$. Now, we distinguish between two cases.

Case 1. $s_1 = 1$. Since the first row of H consists only of 0's and 1's, it follows that only subrows of type (4) with $\beta = 0$ appear in the first row of H and hence only

$$G = \begin{bmatrix} 1 & & & & & & & \\ \vdots & Y & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & & \\ 1 & & & & & & & \\ \vdots & 0 & Y & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & & & & & & & \\ \vdots & 0 & 0 & \cdots & \cdots & \cdots & Y \\ 1 & & & & & & & \end{bmatrix}$$

can be a splitting submatrix of the generator matrix of the code.

Case 2. $s_1 = 0$. All the 1's in the first row of H must participate in subrows of type (1). This implies that we can write any splitting submatrix of the code as

$$G = \begin{bmatrix} 1 & & & & & & & \\ \vdots & Z_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & & \\ 1 & & & & & & & \\ \vdots & 0 & Z_2 & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & & & & & & & \\ \vdots & 0 & 0 & \cdots & \cdots & \cdots & Z_{n_m} \\ 1 & & & & & & & \end{bmatrix},$$

where $\boldsymbol{Z}_i = \boldsymbol{X}$, $1 \le i \le q^{m-1}$, and H has the form

$$H = \begin{bmatrix} 0 & 1 & \cdots & 1 & 0 & \cdots & 0 \\ s_2 & & & & & & \\ \vdots & & \tilde{H}_1 & & & \tilde{H}_2 & \\ s_{m+1} & & & & & & \end{bmatrix}.$$

The matrix

$$\tilde{H} = \begin{bmatrix} s_2 & & \\ \vdots & & \tilde{H}_2 \\ s_{m+1} & & \end{bmatrix}$$

is the parity check matrix of the Hamming code of length n_m, with a splitting submatrix

$$\tilde{G} = \begin{bmatrix} 1 & & & & & & \\ \vdots & Z_{q^{m-1}+1} & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & \\ 1 & & & & & & \\ \vdots & 0 & Z_{q^{m-1}+2} & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & & & & & & \\ \vdots & 0 & 0 & \cdots & \cdots & \cdots & Z_{n_m} \\ 1 & & & & & & \end{bmatrix}.$$

We proceed to examine $\tilde{H}$ and $\tilde{G}$ inductively in the same manner until we reach $s_j = 1$, $j = m + 1 - r$, using Cases 1 and 2. This process and the constructions of §2 and this section lead to Theorem 3.5.

THEOREM 3.5. *Given the parity check matrix*

$$H^*_{m+1} = \begin{bmatrix} S & H' \end{bmatrix}$$

of the Hamming code, then by ordering the columns of $\boldsymbol{H}'$ *we can obtain the unique splitting submatrix of the generator matrix of the code. This unique splitting submatrix* G^*_{m+1} *has the form*

$$G^*_{m+1} = \begin{bmatrix} 1 & & & & & & \\ \vdots & Z_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & \\ 1 & & & & & & \\ \vdots & 0 & Z_2 & \cdots & \cdots & \cdots & 0 \\ 1 & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & & & & & & \\ \vdots & 0 & 0 & \cdots & \cdots & \cdots & Z_{n_m} \\ 1 & & & & & & \end{bmatrix},$$

where $\boldsymbol{Z}_i = \boldsymbol{X}$ *for* $1 \le i \le l$ *and* $\boldsymbol{Z}_i = \boldsymbol{Y}$ *for* $l+1 \le i \le n_m$, $l = \sum_{j=r}^{m-1} q^j$.

Finally, we will mention that the intersection between $C(G^{i_1}_{m+1})$ and $C(G^{i_2}_{m+1})$ for $i_1 \neq i_2$ is not empty since the zero codeword belongs to both of them. Also, $C(G^{i_1}_{m+1}) \neq C(G^{i_2}_{m+1})$ and the proof is done by a careful analysis of the codewords of weight 3 in the codes. But finding $C(G^{i_1}_{m+1}) \bigcap C(G^{i_2}_{m+1})$ is not easy, except for the case $q = 2$ which was dealt with in [1].

Acknowledgment. The author would like to thank Alexander Vardy for his constructive comments.

REFERENCES

[1] T. ETZION AND A. VARDY, *Perfect codes: Constructions, properties and enumeration*, IEEE Trans. Inform. Theory, 40 (1994), pp. 754–763.

[2] B. LINDSTRÖM, *On group and non-group perfect codes in q symbols*, Math. Scand., 25 (1969), pp. 149–158.

[3] F. J. MACWILLIAMS AND N. J. A. SLOANE, *The Theory of Error Correcting Codes*, North-Holland, Amsterdam, 1977.

[4] M. MOLLARD, *A generalized parity function and its use in the construction of perfect codes*, SIAM J. Alg. Disc. Meth., 7 (1986), pp. 113–115.

[5] K. T. PHELPS, *A combinatorial construction of perfect codes*, SIAM J. Alg. Disc. Meth., 4 (1983), pp. 398–403.

[6] ———, *A general product construction for error-correcting codes*, SIAM J. Alg. Disc. Meth., 5 (1984), pp. 224–228.

[7] ———, *A product construction for perfect codes over arbitrary alphabets*, IEEE Trans. Inform. Theory, 30 (1984), pp. 769–771.

[8] K. T. PHELPS AND M. LEVAN, *Kernels of nonlinear hamming codes*, Des. Codes Cryptogr., 6 (1995), pp. 247–257.

[9] J. SCHÖNHEIM, *On linear and nonlinear single-error-correcting q-ary perfect codes*, Inform. and Control, 12 (1968), pp. 23–26.

SIAM J. DISCRETE MATH.
Vol. 9, No. 3, pp. 424–452, August 1996

007

ORTHOGONAL ARRAYS, RESILIENT FUNCTIONS, ERROR-CORRECTING CODES, AND LINEAR PROGRAMMING BOUNDS*

JÜRGEN BIERBRAUER†, K. GOPALAKRISHNAN‡, AND D. R. STINSON§

Abstract. Orthogonal arrays (OAs) are basic combinatorial structures, which appear under various disguises in cryptology and the theory of algorithms. Among their applications are universal hashing, authentication codes, resilient and correlation-immune functions, derandomization of algorithms, and perfect local randomizers. In this paper, we give new explicit bounds on the size of orthogonal arrays using Delsarte's linear programming method. Specifically, we prove that the minimum number of rows in a binary orthogonal array of length n and strength t is at least $2^n - (n2^{n-1}/t + 1)$ and also at least $2^n - (2^{n-2}(n+1)/\lceil \frac{t+1}{2} \rceil)$. We also prove that these bounds are as powerful as the linear programming bound itself for many parametric situations.

An (n, m, t)-resilient function is a function $f : \{0,1\}^n \longrightarrow \{0,1\}^m$ such that every possible output m-tuple is equally likely to occur when the values of t arbitrary inputs are fixed by an opponent and the remaining $n-t$ input bits are chosen independently at random. A basic problem is to maximize t given m and n, i.e., to determine the largest value of t such that an (n, m, t)-resilient function exists. In this paper, we obtain upper and lower bounds for the optimal values of t where $1 \leq n \leq 25$ and $1 \leq m < n$. The upper bounds are derived from Delsarte's linear programming bound, and the lower bounds come from constructions based on error-correcting codes. We also obtain new explicit upper bounds for the optimal values of t.

It was proved by Chor et al. in [*Proc. 26th IEEE Symp. on Foundations of Computer Science*, 1985, pp. 396–407] that an $(n, 2, t)$-resilient function exists if and only if $t < \lfloor \frac{2n}{3} \rfloor$. This result was generalized by Friedman [*Proc. 33rd IEEE Symp. on Foundations of Computer Science*, 1992, pp. 314–319], who proved a bound for general m. We also prove some new bounds, and complete the determination of the optimal resiliency of resilient functions with $m = 3$ and most of the cases for $m = 4$. Several other infinite classes of (optimal) resilient functions are also constructed using the theory of anticodes.

Key words. resilient functions, orthogonal arrays, error-correcting codes, linear programming bound, anticodes

AMS subject classifications. 05B15, 94A60, 94B05, 94B65

1. Introduction.

Orthogonal arrays (OAs) are basic combinatorial structures. They and some natural generalizations appear under various disguises in cryptology and the theory of algorithms. Among the possible applications we mention universal hashing and authentication codes, resilient and correlation-immune functions, derandomization of algorithms, and perfect local randomizers.

Here, we concentrate on resilient functions, two possible applications of which are mentioned in [4] and [9]. The first application concerns the generation of shared random strings in the presence of faulty processors. The second involves renewing a partially leaked cryptographic key. (One setting in which this would be relevant is quantum cryptography [3].) Correlation-immune functions are used in stream ciphers

* Received by the editors July 11, 1994; accepted for publication (in revised form) August 25, 1995. This paper is an expanded and revised version of the extended abstract "Bounds for Resilient Functions and Orthogonal Arrays" by Jürgen Bierbrauer, K. Gopalakrishnan, and D. R. Stinson, which appeared in Lecture Notes in Computer Science 839, 1994, pp. 247–256 (*Advances in Cryptology, Proceedings of CRYPTO'94*).

† Department of Mathematical Sciences, Michigan Technological University, Houghton, MI 49931.

‡ Department of Computer Science, Wichita State University, Wichita, KS 67260. This author's research was supported in part by NSF grant CCR-9121051.

§ Department of Computer Science and Engineering and Center for Communication and Information Science, University of Nebraska–Lincoln, Lincoln, NE 68588. This author's research was supported in part by NSF grant CCR-9121051.

as combining functions for running-key generators that are resistant to a correlation attack (see, for example, Rueppel [25]).

The concept of binary resilient functions was introduced and studied in the papers Chor et al. [9], Bennett, Brassard, and Robert [4], Friedman [12], and Stinson [28]. Here is the definition. Let $n \geq m \geq 1$ be integers and suppose

$$f : \{0,1\}^n \longrightarrow \{0,1\}^m.$$

We will think of f as being a function that accepts n input bits and produces m output bits. Let $t \leq n$ be an integer. Suppose $(x_1, \ldots, x_n) \in \{0,1\}^n$, where the values of t arbitrary inputs are fixed by an opponent, and the remaining $n - t$ input bits are chosen independently at random. Then f is said to be *t-resilient* provided that every possible output m-tuple is equally likely to occur. More formally, the property can be stated as follows. For every t-subset $\{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$, for every choice of $z_j \in \{0,1\} (1 \leq j \leq t)$, and for every $(y_1, \ldots, y_m) \in \{0,1\}^m$, we have

$$Pr(f(x_1, \ldots, x_n) = (y_1, \ldots, y_m) | x_{i_j} = z_j, 1 \leq j \leq t) = \frac{1}{2^m}.$$

We will refer to such a function f as an (n, m, t)-resilient function. Here are some examples from [9] (all addition is modulo 2).

(1) $m = 1$, $t = n - 1$. Define $f(x_1, \ldots, x_n) = x_1 + \cdots + x_n$.
(2) $m = n - 1$, $t = 1$. Define $f(x_1, \ldots, x_n) = (x_1 + x_2, x_2 + x_3, \ldots, x_{n-1} + x_n)$.
(3) $m = 2$, $n = 3h$, $t = 2h - 1$. Define

$$f(x_1, \ldots, x_{3h}) = (x_1 + \cdots + x_{2h}, x_{h+1} + \cdots + x_{3h}).$$

Many interesting results on resilient functions can be found in [4], [9], [12], [28], [14], and [13]. The basic problem is to maximize t given m and n, or equivalently, to maximize m given n and t. An (n, m, t)-resilient function is said to be *optimal* if an $(n, m, t + 1)$-resilient function does not exist.

The paper is organized as follows. In §2 the basic definition of orthogonal arrays and the associated terminology are introduced. After a brief survey of the classical bounds on the size of an orthogonal array in §2.1, stronger bounds based on linear programming are developed in §2.2. In §3 the connections between orthogonal arrays and resilient functions are explored, and as a consequence, the lower bounds on the size of an orthogonal array are translated into upper bounds on the optimal resiliency of resilient functions. In §4 lower bounds on the optimal resiliency of resilient functions are developed. The lower bounds come from various constructions of resilient functions. In §4.1 constructions using linear codes are presented, and in §4.2 constructions using nonlinear codes are presented. Some constructions of new resilient functions from old are discussed in §4.3. The development up to this stage culminates in §4.4 where a table of upper and lower bounds for the optimal value of t when $1 \leq n \leq 25$ and $1 \leq m < n$ is presented.

Several properties of Krawtchouk polynomials are used extensively in the rest of the paper, and because of their great importance, they are dealt with separately in §5. In §6, we consider a recent result of Levenshtein and discuss the implication of the result to resilient functions. The result is that the minimum size $N(n, t)$ of a binary orthogonal array of strength t and length n satisfies the equality $2N(n, t) = N(n + 1, t + 1)$, when t is even.

New explicit bounds on the size of orthogonal arrays are derived in §7 using Delsarte's linear programming method. Specifically, we prove that the minimum

number of rows in a binary orthogonal array of length n and strength t is at least $2^n - (n2^{n-1}/t + 1)$ and also at least $2^n - (2^{n-2}(n+1)/\lceil \frac{t+1}{2} \rceil)$. We prove in §8 that these two bounds are as powerful as the linear programming bound itself for many parametric situations.

It was proved by Chor et al. in [9] that an $(n, 2, t)$-resilient function exists if and only if $t < \lfloor \frac{2n}{3} \rfloor$. The corresponding question for $m \geq 3$ was studied by Friedman [12], who gave some partial results. As a consequence of our new explicit bounds, we have completed the determination of the optimal resiliency of resilient functions with $m = 3$, and we have also done most of the cases for $m = 4$. This is the main theme of §9. The theory of anticodes provides a method for constructing good linear codes. In many parametric situations, the resilient functions derived from these linear codes are optimal. In §9.1, anticodes are studied and such parametric situations are characterized. We conclude the paper with several lingering open problems in §10.

2. Orthogonal arrays. An *orthogonal array* $OA_\lambda(t, k, v)$ is a $\lambda v^t \times k$ array of v symbols, such that in any t columns of the array every one of the possible v^t ordered t-tuples of symbols occurs in exactly λ rows. Usually t is referred to as the *strength* of the orthogonal array, k is called the number of *factors*, v is called the number of *levels*, and λ is called the *index* of the orthogonal array. If $\lambda = 1$, then we write $OA(t, k, v)$. An orthogonal array is said to be *simple* if no two rows are identical. Of course, an array with $\lambda = 1$ is simple. In this paper, we consider only simple arrays.

A *large set of orthogonal arrays* $LOA_\lambda(t, k, v)$ is a set of v^{k-t}/λ simple arrays $OA_\lambda(t, k, v)$ such that every possible k-tuple of symbols occurs in exactly one of the orthogonal arrays in the set.

2.1. Classical bounds. One classical bound for orthogonal arrays is the Rao bound [24], proved in 1947. We record the Rao bound as the following theorem.

THEOREM 2.1. *Suppose there exists an* $OA_\lambda(t, k, v)$. *Then*

$$\lambda v^t \geq 1 + \sum_{i=1}^{t/2} \binom{k}{i} (v-1)^i$$

if t *is even, and*

$$\lambda v^t \geq 1 + \binom{k-1}{(t-1)/2} (v-1)^{(t+1)/2} + \sum_{i=1}^{(t-1)/2} \binom{k}{i} (v-1)^i$$

if t *is odd.*

Another classical bound which provides necessary conditions for the existence of orthogonal arrays of index unity ($\lambda = 1$) is the Bush bound [7], proved in 1952. This bound is as follows.

THEOREM 2.2. *Suppose there exists an* $OA(t, k, v)$, *where* $t > 1$. *Then*

$$\begin{aligned} k &\leq v + t - 1 \quad \text{if } v \geq t,\ v \text{ even}, \\ k &\leq v + t - 2 \quad \text{if } v \geq t \geq 3,\ v \text{ odd}, \\ k &\leq t + 1 \quad \text{if } v \leq t. \end{aligned}$$

2.2. Bounds based on linear programming. We can often find better lower bounds on the size of orthogonal arrays by using Delsarte's linear programming bound; this is the main theme of this section.

While developing the bounds based on linear programming techniques, we will be using several standard results from coding theory without proof; the reader is referred to MacWilliams and Sloane [22] for background information on error-correcting codes.

An (n, M, d) binary error-correcting code $\mathcal{C}$ is a set of M vectors of length n such that the Hamming distance between any two vectors in $\mathcal{C}$ is at least d. A code is said to be *linear* if the codewords form an m-dimensional subspace of $[GF(2)]^n$, where $M = 2^m$. We will denote such a linear code as an $[n, m, d]$ linear code. Let A_i be the number of codewords having Hamming weight i. The sequence $(A_0, A_1, \ldots, A_n)$ is called the *weight distribution* of the code. The homogeneous polynomial

$$\sum_{i=0}^{n} A_i x^{n-i} y^i$$

is called the *weight enumerator* of $\mathcal{C}$ and is denoted by $W_{\mathcal{C}}(x, y)$. Clearly

$$A_0 + A_1 + \cdots + A_n = M.$$

The *distance distribution* of the code is defined to be the sequence $(B_0, B_1, \ldots, B_n)$, where

$$B_i = \frac{1}{M} |\{(u, v) : u, v \in \mathcal{C}, d(u, v) = i\}|.$$

Note that $B_0 = 1$ and $B_0 + B_1 + \cdots + B_n = M$. It is easy to observe that, for a linear code, the distance distribution is identical to the weight distribution.

If $\mathcal{C}$ is a linear code, the *dual code* $\mathcal{C}^{\perp}$ is defined to be the set of all vectors u having inner product zero with every codeword of $\mathcal{C}$. Let A_i' denote the number of codewords having weight i in $\mathcal{C}^{\perp}$. The weight enumerators of $\mathcal{C}$ and $\mathcal{C}^{\perp}$ are related by the MacWilliams identity shown below. For a proof, see, for example, [22, p. 127].

$$W_{\mathcal{C}^{\perp}}(x, y) = \frac{1}{M} W_{\mathcal{C}}(x + y, x - y).$$

Equivalently,

$$\sum_{k=0}^{n} A_k' x^{n-k} y^k = \frac{1}{M} \sum_{i=0}^{n} A_i (x + y)^{n-i} (x - y)^i.$$

If we write

$$(x + y)^{n-i} (x - y)^i = \sum_{k=0}^{n} P_k(i) x^{n-k} y^k,$$

then we get

$$A_k' = \frac{1}{M} \sum_{i=0}^{n} A_i P_k(i).$$

$P_k(i)$ is the value of the *Krawtchouk polynomial* $P_k(x)$ at integer i and can be explicitly described as

$$P_k(i) = \sum_{j=0}^{k} (-1)^j \binom{i}{j} \binom{n-i}{k-j}, \qquad k = 0, 1, 2, \ldots .$$

In the case of nonlinear codes, we cannot talk of the dual code $\mathcal{C}^\perp$, since the code $\mathcal{C}$ is not a vector space. However, we can still transform the distance distribution $(B_0, B_1, \ldots, B_n)$ of the code $\mathcal{C}$ in a similar way using the Krawtchouk polynomials to obtain the *dual distance distribution* $(B_0', B_1', \ldots, B_n')$, which is defined as follows:

$$B_k' = \frac{1}{M} \sum_{i=0}^{n} B_i P_k(i).$$

If $B_i' = 0$ for $1 \le i \le d' - 1$ and $B_{d'}' \neq 0$, then d' is called the *dual distance* of the code $\mathcal{C}$. This concept was defined by Delsarte [11]. If $\mathcal{C}$ is linear then $(B_0', B_1', \ldots, B_n')$ is indeed the distance distribution as well as the weight distribution of the dual code $\mathcal{C}^\perp$, and d' is the minimum distance of $\mathcal{C}^\perp$. It turns out that even when $\mathcal{C}$ is not a linear code, the dual distance d' has a combinatorial significance. The following theorem describes the combinatorial significance of the dual distance d', and the proof, due to Delsarte, can be found in [22, p. 139] and [11].

THEOREM 2.3. *If we write the vectors in $\mathcal{C}$ as rows of an $M \times n$ array, then any set of $r \le d' - 1$ columns contains each r-tuple exactly $M/2^r$ times, and d' is the largest number with this property. In other words, $\mathcal{C}$ is an orthogonal array $OA_\lambda(d' - 1, n, 2)$ where $\lambda = M/2^{d'-1}$.*

It is clear that for any code $\mathcal{C}$, we have $B_i \ge 0$ for $i = 0, 1, \ldots, n$. On the other hand, it is a nontrivial theorem (see, for example, [22, p. 139] and [11]) that $B_i' \ge 0$ for $i = 0, 1, \ldots, n$.

Suppose a simple $OA_\lambda(t, n, 2)$ exists. Let M be the number of rows in this orthogonal array. A lower bound on M can be obtained by solving a suitable linear programming problem, if we view the rows of this orthogonal array as codewords of a code $\mathcal{C}$ (see [22, §4 of Chap. 17] and [10] for similar approaches to a different problem). Let $(B_0, B_1, \ldots, B_n)$ be the distance distribution of $\mathcal{C}$ and let $(B_0', B_1', \ldots, B_n')$ be its transform. Then

$$M = B_0 + B_1 + \cdots + B_n.$$

Also

$$\begin{aligned} B_k' &= \frac{1}{M} \sum_{i=0}^{n} B_i P_k(i) \\ &= \frac{1}{M} \left(\sum_{i=1}^{n} B_i P_k(i) + \binom{n}{k} \right). \end{aligned}$$

Further,

$$\sum_{k=0}^{n} B_k' = \frac{1}{M} \sum_{i=0}^{n} B_i \sum_{k=0}^{n} P_k(i) = \frac{1}{M} B_0 2^n = \frac{2^n}{M}.$$

Here we have used

$$\sum_{k=0}^{n} P_k(i) = \delta_{i,0} 2^n.$$

This is a property of the Krawtchouk polynomials and can be found in §5 as equation (7). Moreover, the Krawtchouk transform is an involutional operation on the set of real $(n+1)$-tuples with nonvanishing sum and constant term $= 1$. Hence, if we compute

$$B_k'' = \frac{M}{2^n}\left(\sum_{i=1}^{n} B_i' P_k(i) + \binom{n}{k}\right), \qquad k = 0, 1, \ldots, n,$$

then $B_k'' = B_k$.

In view of Theorem 2.3, the dual distance of C is at least $t+1$ since it is an orthogonal array of strength t. Hence we have $B_k' = 0$ for $1 \le k \le t$. So we formulate the following linear programming problem which we will refer to as $LP1$:

Maximize $w_{t+1} + w_{t+2} + \cdots + w_n$
subject to

$$\sum_{i=t+1}^{n} w_i P_k(i) \ge -\binom{n}{k} \text{ for } 1 \le k \le n$$

$w_i \ge 0$ for $t+1 \le i \le n$

Let $W = W(n,t)$ be the optimal solution to the above linear programming problem. Then we have

$$\frac{2^n}{M} = \sum_{i=0}^{n} B_i' \le 1 + W. \tag{1}$$

Thus, by solving a linear program, we get a lower bound for the size of an orthogonal array.

3. Upper bounds for resilient functions. In this section, we will be concerned with developing upper bounds for the optimum value of t for a given n and m. It is easy to see that $n \ge m + t$, and so we have the upper bound

$$t \le n - m. \tag{2}$$

However, the above bound is usually very weak, and better upper bounds can be obtained by exploring the relationship between resilient functions and orthogonal arrays. These connections were explored and several results were proved in [28], [9], and [8]. We will briefly survey them and then use the results for our purposes.

The proof of the following theorem which elucidates the connection between resilient functions and orthogonal arrays can be found in [28].

THEOREM 3.1. *An (n,m,t)-resilient function is equivalent to a large set of orthogonal arrays $LOA_{2^{n-m-t}}(t,n,2)$.*

In view of Theorem 3.1, any necessary condition for the existence of an orthogonal array $OA_{2^{n-m-t}}(t,n,2)$ is also a necessary condition for the existence of an (n,m,t)-resilient function. We obtain the following corollary, which gives a necessary condition

for the existence of an (n, m, t) - resilient function from the classical Rao bound for orthogonal arrays.

COROLLARY 3.2 (see [9, 28]). *Suppose there exists an (n, m, t)-resilient function. Then*

$$m \leq n - \log_2 \left[\sum_{i=0}^{t/2} \binom{n}{i} \right]$$

if t is even, and

$$m \leq n - \log_2 \left[\binom{n-1}{(t-1)/2} + \sum_{i=0}^{(t-1)/2} \binom{n}{i} \right]$$

if t is odd.

Proof. Set $v = 2$, $k = n$ in Theorem 2.1 and apply Theorem 3.1. □

Similarly, from the classical Bush bound, we can obtain the following corollary, which was proved in [4] from first principles.

COROLLARY 3.3 (see [4, 28]). *There exists an (n, m, t)-resilient function with $n = m + t$ if and only if $m = 1$ or $t = 1$.*

Proof. The cases $t = 1$ and $m = 1$ were given earlier in examples. So, suppose $n = m + t$ and $2 \leq t \leq n - 2$. Apply Theorem 2.2 with $v = 2$ to get $m + t \leq t + 1$, or $m \leq 1$, a contradiction. □

The upper bounds based on the Rao bound for orthogonal arrays are stronger than the ones obtained using the trivial bound of equation (2). However, we can often do better by using Delsarte's linear programming bound. In view of Theorem 3.1, an (n, m, t)-resilient function exists if and only if an $LOA_{2^{n-m-t}}(t, n, 2)$ exists. Clearly, an $LOA_{2^{n-m-t}}(t, n, 2)$ exists only if an $OA_{2^{n-m-t}}(t, n, 2)$ exists. The number of rows in this orthogonal array is given by

$$M = 2^{n-m-t} 2^t = 2^{n-m}. \tag{3}$$

In view of the bound of inequality (1) of §2.2, this immediately implies that

$$m \leq \log_2(1 + W).$$

Thus we have an upper bound for the optimal value of m, given n and t. The upper bound for the optimal value of t for a given n and m can be trivially computed once we have a table of upper bounds for the optimal value of m. The upper bounds depicted in Table 2 are obtained in this manner using MAPLE V to do the necessary computation.

In a forthcoming book [16], Hedayat, Sloane, and Stufken provide a compilation of a table of minimal possible index of orthogonal arrays with 2, 3, and 4 levels, k factors, and strength t. Upper bounds for the optimal value of t can also be obtained from these tables.

We now have three upper bounds on the optimal value of t, viz., the trivial bound of equation (2), the bound based on the Rao bound for orthogonal arrays, and the bound based on Delsarte's linear programming (LP) bound. In Table 1, a comparison of these three bounds for some values of n and m is provided. This table illustrates the relative strengths of the various bounds. The improvements provided by the stronger bounds are more marked for larger values of n.

TABLE 1
A comparison of the bounds.

n	m	Trivial bound	Rao bound	LP bound
18	2	16	14	11
19	2	17	15	11
20	2	18	15	12
18	3	15	12	9
19	3	16	13	9
20	3	17	13	10
18	4	14	10	8
19	4	15	11	9
20	4	16	12	9

4. Lower bounds for resilient functions. Obviously, any method of construction of resilient functions yields a lower bound on the optimal value of t. The most important construction method for resilient functions uses linear error-correcting codes, which are discussed in §4.1. Recently, Stinson and Massey [29] constructed resilient functions from nonlinear codes as well. This construction is discussed in §4.2. Some constructions of new resilient functions from old ones are developed in §4.3. Finally, a table of bounds for resilient functions is presented in §4.4

4.1. Constructions using linear codes. The most basic and well-known construction method for resilient functions uses linear error-correcting codes, and therefore we will explore the connection between t-resilient functions and error-correcting codes in this section. A resilient function constructed in this way is said to be a linear resilient function.

An $[n, m, d]$ linear code is an m-dimensional subspace C of $[GF(2)]^n$ such that the Hamming distance between any two vectors in C is at least d. Let G be an $m \times n$ matrix whose rows form a basis for C; G is called a *generating matrix* for C. The following construction for resilient functions was given in [9, 4].

THEOREM 4.1. *Let G be a generating matrix for an $[n, m, d]$ linear code C. Define a function $f : [GF(2)]^n \longrightarrow [GF(2)]^m$ by the rule $f(x) = xG^T$. Then f is an $(n, m, d-1)$-resilient function.*

This result can easily be seen to be true using the orthogonal array characterization of Theorem 3.1. The inverse image $f^{-1}(0, 0, \ldots, 0)$ is in fact the dual code $C^\perp$. Any $d-1$ columns of the generating matrix for $C^\perp$ (= the parity check matrix for C) are linearly independent, since the minimum distance of C is d. This implies that any set of $d-1$ columns of $C^\perp$ contains each $d-1$ tuple exactly $|C^\perp|/2^{d-1} = 2^{n-m-d+1}$ times, and hence it follows that $C^\perp$ is an orthogonal array $OA_{2^{n-m-d+1}}(d-1, n, 2)$. Now, any other inverse image $f^{-1}(y)$ is an additive coset of $C^\perp$, and thus is also an $OA_{2^{n-m-d+1}}(d-1, n, 2)$. Hence we obtain 2^m OAs that form a large set. By Theorem 3.1, it follows that f is an $(n, m, d-1)$-resilient function.

Thus, whenever an $[n, m, d]$ linear code exists, so does an $(n, m, d-1)$-resilient function. Brouwer and Verhoeff [6] provide a compilation of the best known linear binary codes. These provide lower bounds for the optimum value of t for a given n and m.

4.2. Constructions using nonlinear codes. Often the lower bounds derived from the constructions based on linear codes do not match the upper bounds. In some such cases, constructions from nonlinear codes might be of help.

An (n, M, d) code is said to be a *systematic code* if there exist m coordinates such that every possible m-tuple occurs in exactly one codeword within the m spec-

ified coordinates (of course, this implies $M = 2^m$). These m coordinates are called *information bits* and the remaining $n - m$ coordinates are called *parity-check bits.* Clearly, any linear code is a systematic code. However, there are several well-known classes of nonlinear codes (for example, Preparata codes and Kerdock codes) which are systematic codes. The following theorem was proved by Stinson and Massey in [29].

THEOREM 4.2. *If there exists a systematic (n, M, d) code, $\mathcal{C}$, having a dual distance d', then there is an $(n, n-m, d'-1)$-resilient function, where $M = 2^m$.*

Note that this result subsumes Theorem 4.1, since any linear code is a systematic code.

It was conjectured in [9] and [4] that if there exists a resilient function with certain parameters, then there exists a linear function with the same parameters. However, this conjecture was disproved by Stinson and Massey [29]. They constructed a class of nonlinear resilient functions from the Kerdock codes; further, there are no linear resilient functions with the same parameters. The proof of the following theorem can be found in [29].

THEOREM 4.3. *For any odd integer $r \geq 3$, a $(2^{r+1}, 2^{r+1} - 2r - 2, 5)$-resilient function exists, but no linear resilient function with these parameters exists.*

The first member of the above infinite class will be of special interest to us. It is the $(16, 8, 5)$-resilient function constructed from the Kerdock code $\mathcal{K}(4)$, which is also known as the Nordstrom–Robinson code. This resilient function is optimal since the upper bound for t is 5 when $n = 16$ and $m = 8$.

4.3. New resilient functions from old. We now proceed to describe some constructions of new resilient functions from old. When we apply these constructions to the $(16, 8, 5)$-resilient function, we obtain new resilient functions which are optimal.

THEOREM 4.4. *Suppose there is a linear (n, m, t)-resilient function. Then there exists an $(n-1, m, t-1)$- and an $(n-1, m-1, t)$-resilient function.*

Proof. Since the (n, m, t)-resilient function is linear, it must have been derived from an $[n, m, t+1]$ linear code $\mathcal{C}$. The *punctured code* is obtained by deleting a fixed coordinate from every codeword in $\mathcal{C}$, thus producing a linear $[n-1, m, t]$ code. From this punctured code, we obtain an $(n-1, m, t-1)$-resilient function.

If we choose a coordinate, say i, of $\mathcal{C}$ which is not always zero, then there will be 2^{m-1} codewords in $\mathcal{C}$ with a "0" in the ith coordinate, and 2^{m-1} codewords with a "1" in the ith coordinate. If we delete the codewords with a "1" in the ith coordinate, and then delete the ith coordinate, then we obtain the *shortened code*, which is an $[n-1, m-1, t+1]$ linear code. From this shortened code, we obtain an $(n-1, m-1, t)$-resilient function. □

Unfortunately, Theorem 4.4 is not of much use for us, as the only $(16, 8, 5)$-resilient functions are nonlinear. Let us see if we can generalize these constructions to the case of an arbitrary (nonlinear) resilient function. In one case, we can always do so.

THEOREM 4.5. *Suppose there is an (n, m, t)-resilient function. Then there is an $(n-1, m, t-1)$-resilient function.*

Proof. Let $f : \{0,1\}^n \longrightarrow \{0,1\}^m$ be an (n, m, t)-resilient function. Define $g : \{0,1\}^{n-1} \longrightarrow \{0,1\}^m$ by the rule $g(x_1, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, 0)$. It is easy to see that g is $(t-1)$-resilient. □

Let us recast this procedure in terms of large sets of orthogonal arrays. Suppose A is an $OA_\lambda(t, n, v)$. Let x be a symbol. Delete all rows in which x does not occur in the last column, and then delete the last column. In this way we obtain an $OA_\lambda(t-1, n-1, v)$ (this operation is called "derivation" [30]). If we have a large set of OAs,

then the set of derived arrays also forms a large set. Hence, we obtain the following general result (which holds for arbitrary "v").

THEOREM 4.6. *Suppose there is a large set of* $OA_\lambda(t, n, v)$. *Then there is a large set of* $OA_\lambda(t-1, n-1, v)$.

On the other hand, it does not appear that the existence of an (n, m, t)-resilient function automatically implies the existence of an $(n-1, m-1, t)$-resilient function. Again, it is useful to think of this in terms of orthogonal arrays. Suppose A is an $OA_\lambda(t, n, v)$, and we delete the last column. Then it is clear that we get an $OA_\lambda(t, n-1, v)$ (this operation is called "restriction" [30]). Unfortunately, the set of restricted arrays of a large set of OAs is *never* a large set, since every possible row occurs v times in the set of restricted arrays.

However, in certain special cases we can obtain an $(n-1, m-1, t)$- resilient function from an (n, m, t)-resilient function. One such case is when the original (n, m, t)-resilient function is obtained from a systematic code.

THEOREM 4.7. *Suppose there is an* (n, m, t)*-resilient function derived from a systematic code. Then there is an* $(n-1, m-1, t)$*-resilient function (which is also derived from a systematic code).*

Proof. Let C be the $(n, 2^{n-m}, d)$ systematic code from which the (n, m, t)-resilient function is derived. Then the dual distance of the code is $t+1$. If we choose a parity-check coordinate, say i, of C and delete the ith coordinate from every codeword in C, then we obtain the punctured code, which is an $(n-1, 2^{n-m}, d-1)$ code. It is easy to see that the punctured code is also a systematic code. The deletion of a column of an orthogonal array does not change its strength, and so the dual distance of the punctured code is still $t+1$. Hence we can obtain an $(n-1, m-1, t)$-resilient function from the punctured code. □

A $(16, 8, 5)$-resilient function exists by virtue of Theorem 4.3. Applying Theorem 4.7 repeatedly, starting with the $(16, 8, 5)$-resilient function, we get resilient functions with parameters $(15, 7, 5)$, $(14, 6, 5)$, and $(13, 5, 5)$. Applying Theorem 4.5 to each of these resilient functions, we get resilient functions with parameters $(15, 8, 4)$, $(14, 7, 4)$, $(13, 6, 4)$, and $(12, 5, 4)$. All these resilient functions are optimal, as they match the upper bounds derived from the linear programming bound.

4.4. A table of bounds. In Table 2 upper and lower bounds for the optimal value of t when $1 \le n \le 25$ and $1 \le m < n$ are given. The rows are indexed by values of n and the columns are indexed by values of m. Suppose that the ordered pair (u, l) is the entry in the mth column of the nth row. This indicates that there is no $(n, m, u+1)$ resilient binary function. (Specifically, such a function is ruled out by the linear programming bound of §2.2.) It also means that there exists an (n, m, l)-resilient binary function. Specifically, there exists an $[n, m, l+1]$ binary linear code from which we can obtain the (n, m, l)-resilient binary function unless the entry is marked with a *. The (n, m, l)-resilient functions corresponding to the entries marked with a * are obtained from constructions using nonlinear codes discussed in this section. If there is a single entry in the mth column of the nth row, then it indicates that $u = l$. In other words, the entry is the optimal value of t. It is clear from the table that the optimal value of t is determined exactly for all cases where $1 \le m < n \le 17$.

5. Properties of Krawtchouk polynomials. The Krawtchouk polynomials were introduced and defined in §2.2. In the rest of the paper, we will be using several properties of Krawtchouk polynomials extensively, and so we list them together in this section. This is not intended to be an exhaustive list of their properties; we list

Table 2: Upper and lower bounds for the optimal value of resiliency

n,m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
2	1																							
3	2	1																						
4	3	1	1																					
5	4	2	1	1																				
6	5	3	2	1	1																			
7	6	3	3	2	1	1																		
8	7	4	3	3	1	1	1																	
9	8	5	3	3	2	1	1	1																
10	9	5	4	3	3	2	1	1	1															
11	10	6	5	4	3	3	2	1	1	1														
12	11	7	5	5	4*	3	3	2	1	1	1													
13	12	7	6	5	5*	4*	3	3	2	1	1	1												
14	13	8	7	6	5	5*	4*	3	3	2	1	1	1											
15	14	9	7	7	6	5	5*	4*	3	3	2	1	1	1										
16	15	9	7	7	7	5	5	5*	3	3	3	1	1	1	1									
17	16	10	8	7	7	6	5	5	4	3	3	2	1	1	1	1								
18	17	11	9	(8,7)	7	7	6	5	5	(4,3)	3	3	2	1	1	1	1							
19	18	11	9	(9,8)	(8,7)	7	7	6	5	(5,4)	(4,3)	3	3	2	1	1	1	1						
20	19	12	10	9	(9,8)	(8,7)	7	7	6	5	(5,4)	(4,3)	3	3	2	1	1	1	1					
21	20	13	11	9	9	(9,7)	7	7	7	6	5	(5,4)	3	3	3	2	1	1	1	1				
22	21	13	11	10	9	(9,8)	(8,7)	7	7	7	6	5	4	3	3	3	2	1	1	1	1			
23	22	14	11	11	10	9	(9,8)	(8,7)	7	7	7	6	5	4	3	3	3	2	1	1	1	1		
24	23	15	12	11	11	(10,9)	9	(9,7)	(8,7)	7	7	7	5	5	(4,3)	3	3	3	2	1	1	1	1	
25	24	15	13	11	11	(11,10)	9	(9,8)	(9,7)	(8,7)	7	7	(6,5)	5	(5,4)	(4,3)	3	3	3	2	1	1	1	1

Refer to Section 4.4 to interpret the above table of bounds.

only those properties which we need in the sequel. All these properties can be found either as theorems or as problems in [22, Chap. 5, §7].

Recall that for any positive integer n, the Krawtchouk polynomial $P_k^n(x) = P_k(x)$ is defined by

$$P_k(x) = \sum_{j=0}^{k} (-1)^j \binom{x}{j} \binom{n-x}{k-j}, \qquad k = 0, 1, 2, \ldots .$$

The first few Krawtchouk polynomials are shown below.

$$\begin{aligned} P_0(x) &= 1, \\ P_1(x) &= n - 2x, \\ P_2(x) &= \binom{n}{2} - 2nx + 2x^2. \end{aligned}$$

The following two simple properties deal with the values of Krawtchouk polynomials under certain special situations.

$$P_k(0) = \binom{n}{k}, \tag{4}$$

$$P_0(i) = 1. \tag{5}$$

From the definition of the Krawtchouk polynomials, it immediately follows that

$$\sum_{i=0}^{n} P_i(k) z^i = (1+z)^{n-k}(1-z)^k. \tag{6}$$

Setting $z = 1$ in the identity (6) gives

$$\sum_{i=0}^{n} P_i(k) = 2^n \delta_{k,0}, \tag{7}$$

where $\delta_{r,s}$ is the Kronecker symbol defined by $\delta_{r,s} = 1$ if $r = s$ and $\delta_{r,s} = 0$ if $r \neq s$.

There is also some sort of symmetry among the values of the Krawtchouk polynomial which are captured in the following identities.

$$P_k(i) = (-1)^k P_k(n-i), \tag{8}$$

$$P_k(i) = (-1)^i P_{n-k}(i). \tag{9}$$

Actually, identity (9) can be inferred from identity (8) using the following more general identity, which we refer to as the inversion identity.

$$\binom{n}{i} P_s(i) = \binom{n}{s} P_i(s). \tag{10}$$

Finally, we will need the following simple identity which follows at once from the identity (6):

$$P_k^{n+1}(i) = P_k^n(i) + P_{k-1}^n(i). \tag{11}$$

6. Self-complementary orthogonal arrays. In this section we shall consider a theorem concerning the construction of orthogonal arrays of strength $t+1$ from arrays of strength t when t is even. This result seems to have been rediscovered several times over the years (see [26], [1], [17], and [19]). We present the proof of this theorem (for the sake of completeness) and then discuss its implications to resilient functions.

THEOREM 6.1. *Let t be even. Then an $OA_\lambda(t,n,2)$ exists if and only if an $OA_\lambda(t+1,n+1,2)$ exists, and hence the minimum size $N(n,t)$ of an OA of strength t and length n satisfies the equality*

$$2N(n,t) = N(n+1,t+1).$$

Proof. We will first show that an $OA_\lambda(t+1,n+1,2)$ exists if an $OA_\lambda(t,n,2)$ exists. Suppose an $OA_\lambda(t,n,2)$ exists. Let A be such an OA. Let M denote the number of rows in A. Let P be an array formed by adding a column of zeroes to the array A. Now define the array B to consist of all the rows of the array P and their complements. More formally, a binary vector x of length $n+1$ is a row of B if and only if either x or $1+x$ is a row of P, where 1 is the all-ones vector of length $n+1$. We shall prove that the array B is indeed an $OA_\lambda(t+1,n+1,2)$.

Let $(A_0, A_1, \ldots, A_n)$ be the distance distribution of the OA A considered as a code, and similarly let $(B_0, B_1, \ldots, B_n, B_{n+1})$ be the distance distribution of the OA B. As we saw earlier, it is always the case that $B_0 = 1$. Also

$$\begin{aligned} B_{n+1} &= \frac{1}{2M}|\{(u,v) : u,v \in B, d(u,v) = n+1\}| \\ &= 1, \end{aligned}$$

since for every row u of B, there is exactly one row v of B such that $d(u,v) = n+1$ (v is simply the complement of u), and B has $2M$ rows. Let us now express B_i in terms of the A_i's. By definition,

$$B_i = \frac{1}{2M}|\{(u,v) : u,v \in B, d(u,v) = i\}|.$$

The number of ordered pairs (u,v) such that $d(u,v) = i$ and both u and v are rows of P is simply $A_i M$ since the last element is zero for all the rows of P. We will denote by P^c the complements of the rows of P. As P^c is merely a translate of P, the number of ordered pairs (u,v) such that $d(u,v) = i$ and both u and v are rows of P^c is once again $A_i M$.

Now suppose $u \in P$ and $v \in P^c$. Then clearly $1+v \in P$ and $d(u,v) = i$ if and only if $d(u, 1+v) = n+1-i$. So it follows that the number of ordered pairs (u,v) such that $u \in P$, $v \in P^c$, and $d(u,v) = i$ is simply $A_{n+1-i}M$. By symmetry we also have that the number of ordered pairs (u,v) such that $u \in P^c$, $v \in P$, and $d(u,v) = i$ is also $A_{n+1-i}M$.

Altogether, the number of ordered pairs (u,v) such that $d(u,v) = i$ and both u and v are rows of B is simply $2M(A_i + A_{n+1-i})$, and so, by definition, we have for $1 \le i \le n$

$$B_i = A_i + A_{n+1-i}.$$

Thus the distance distribution of B is completely determined by that of A.

Let $(A_0', A_1', \ldots, A_n')$ be the transform of the distance distribution of A and let $(B_0', B_1', \ldots, B_n')$ be the transform of the distance distribution of B. It is always the

case that $A_0' = B_0' = 1$. As A is an orthogonal array of strength t, we also have that $A_i' = 0$ for $1 \leq i \leq t$ by Theorem 2.3. We now compute B_k' in terms of the values of A_i' using the properties of the Krawtchouk polynomials developed in §5. The computation proceeds as follows:

$$
\begin{aligned}
B_k' &= \sum_{i=0}^{n+1} B_i P_k^{n+1}(i) \\
&= B_0 P_k^{n+1}(0) + B_{n+1} P_k^{n+1}(n+1) + \sum_{i=1}^{n} B_i P_k^{n+1}(i) \\
&= \binom{n+1}{k}[1+(-1)^k] + \sum_{i=1}^{n}(A_i + A_{n+1-i}) P_k^{n+1}(i) \\
&\quad \text{using identities (4) and (8)} \\
&= \binom{n+1}{k}[1+(-1)^k] + \sum_{i=1}^{n} A_i P_k^{n+1}(i) + \sum_{i=1}^{n} A_{n+1-i} P_k^{n+1}(i) \\
&= \binom{n+1}{k}[1+(-1)^k] + \sum_{i=1}^{n} A_i P_k^{n+1}(i) + (-1)^k \sum_{i=1}^{n} A_{n+1-i} P_k^{n+1}(n+1-i) \\
&\quad \text{using identity (8)} \\
&= \binom{n+1}{k}[1+(-1)^k] + \sum_{i=1}^{n} A_i P_k^{n+1}(i) + (-1)^k \sum_{i=1}^{n} A_i P_k^{n+1}(i) \\
&= [1+(-1)^k]\left(\binom{n+1}{k} + \sum_{i=1}^{n} A_i P_k^{n+1}(i)\right) \\
&= [1+(-1)^k] \sum_{i=0}^{n} A_i P_k^{n+1}(i) \\
&\quad \text{using identity (4)} \\
&= [1+(-1)^k]\left(\sum_{i=0}^{n} A_i P_k^{n}(i) + \sum_{i=0}^{n} A_i P_{k-1}^{n}(i)\right) \\
&\quad \text{using identity (11)} \\
&= [1+(-1)^k](A_k' + A_{k-1}').
\end{aligned}
$$

As A is an orthogonal array of strength t, its dual distance is $t+1$ and so $A_i' = 0$ for $1 \leq i \leq t$. This fact implies, in view of the above expression for B_k', that $B_k' = 0$ for $2 \leq k \leq t$. Also $B_1' = 0$, as $[1+(-1)^k] = 0$ when $k = 1$. Finally $B_{t+1}' = 0$, as $[1+(-1)^{t+1}] = 0$ when t is even (which is a hypothesis in the theorem). Thus $B_k' = 0$ for $1 \leq k \leq t+1$, and hence it follows that the strength of B as an orthogonal array is at least $t+1$ by Theorem 2.3. Thus, when t is even, an $OA_\lambda(t+1, n+1, 2)$ exists if an $OA_\lambda(t, n, 2)$ exists.

The fact that an $OA_\lambda(t, n, 2)$ exists if an $OA_\lambda(t+1, n+1, 2)$ exists is easy and follows immediately from the proof of Theorem 4.6. Note that we do not need the assumption that t is even for this part. A simple consequence is that the minimum size $N(n, t)$ of an OA of strength t and length n satisfies the equality

$$2N(n, t) = N(n+1, t+1). \qquad \square$$

The above constructions easily extend to large sets of orthogonal arrays as well and hence an equivalent statement can be made about resilient functions.

THEOREM 6.2. *Let t be even. Then an (n, m, t)-resilient function exists if and only if an $(n+1, m, t+1)$-resilient function exists. Hence, if the optimal resiliency t is even for parameters (n, m), then the optimal resiliency is $t+1$ for parameters $(n+1, m)$.*

Note further that an (n, m, t)-resilient function always exists if an $(n+1, m, t+1)$-resilient function exists, as stated in Theorem 4.5; i.e., the assumption that t is even is not required.

Using Theorems 4.5 and 6.2, we can make the following comments about Table 2. For each fixed m, the optimal resiliency increases by at most 1 when n is increased by 1. So all the integers appear in each column in their natural order. An integer may appear several times in a column, but any even integer can appear exactly once in each column as dictated by Theorem 6.2. Stated in another way, for each m the probability that the optimal resiliency t is odd for a randomly picked n is at least half.

7. Explicit bounds for orthogonal arrays and resilient functions. For large values of t, the orthogonal array bounds obtained by the linear programming technique are usually much better than the Rao bound. The disadvantage of the linear programming bound is that one needs to solve a different linear program for every parameter situation. Of course, this can be done in polynomial time by using Karmarkar's algorithm, among others. Nonetheless, the linear programming bound is fairly difficult to compute as compared to the Rao bound. Thus it is of interest to try to derive explicit bounds as corollaries of the linear programming bound. We will pursue this idea in this section.

Let us first form the dual of the linear programming problem LP1. We will refer to the dual as $DLP1$.

$$\text{Minimize } \sum_{k=1}^{n} x_k \binom{n}{k}$$

subject to

$$\sum_{k=1}^{n} x_k P_k(i) \leq -1 \text{ for } i = t+1 \text{ to } n$$

$$x_1, x_2, \ldots x_n \geq 0$$

It is a standard theorem in the theory of linear programming (see, for example, [23]) that in a pair of primal-dual linear programs the optimal value of the maximization problem will always be less than or equal to the value attained by the objective function of the minimization problem at any feasible solution vector. So any feasible solution to the dual linear program DLP1 would yield a lower bound on the size of the orthogonal array of strength t and length n and consequently an upper bound on m of t-resilient functions on n-tuples. In fact, similar techniques were previously used by Delsarte and others to obtain explicit bounds on the size of a code with a given minimum distance [22, §4, Chap. 17] (see [19] for recent results in this vein).

Assume $2(t+1) > n$. Consider the solution vector

$$x_k = \begin{cases} \frac{1}{2(t+1)-n} & \text{if } k = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The nonnegativity conditions are certainly satisfied. As $P_1(x) = n - 2x$, we have

$$\sum_{k=1}^{n} x_k P_k(i) = x_1(n - 2i) \leq -1$$

for $i \geq t+1$. So this is indeed a feasible solution. The value of the objective function at this solution vector is

$$x_1 n = \frac{n}{2(t+1) - n}.$$

Hence it follows that

$$\frac{2^n}{M} \leq 1 + W \leq 1 + \frac{n}{2(t+1) - n} = \frac{2(t+1)}{2(t+1) - n}. \tag{12}$$

As a consequence, we get the following theorem, which was first proved by Friedman [12] using very different methods.

THEOREM 7.1. *Suppose an $OA_\lambda(t, n, 2)$ exists. Let $M = \lambda 2^t$ be the number of rows in the OA. Then*

$$M \geq 2^n - \frac{n 2^{n-1}}{t+1}. \tag{13}$$

Note. The assumption that $2(t+1) > n$ is not needed for the final statement of the theorem as otherwise the right-hand side of the above equation is not positive and hence the theorem will be trivially true.

This theorem along with equation (3) immediately yields an explicit bound on resilient functions as well. We state this bound as the following corollary.

COROLLARY 7.2. *If an (n, m, t)-resilient function exists, then*

$$t \leq \left\lfloor \frac{2^{m-1} n}{2^m - 1} \right\rfloor - 1.$$

Now we derive a second bound. Assume $2t > n - 3$, t even. Consider the solution vector

$$x_k = \begin{cases} \frac{1}{2t+3-n} & \text{if } k = 1 \text{ or } n, \\ 0 & \text{otherwise.} \end{cases}$$

Again, the nonnegativity is obvious. The main condition is

$$\frac{1}{2t+3-n}(P_1(i) + P_n(i)) = \frac{n - 2i + (-1)^i}{2t+3-n} \leq -1$$

for $i \geq t+1$. This is trivially satisfied for $i > t+1$. For $i = t+1$ it is satisfied with equality as $i = t+1$ is odd. The value attained by the objective function at this feasible solution is

$$x_1 \binom{n}{1} + x_n \binom{n}{n} = \frac{n+1}{2t+3-n}.$$

Hence it follows that

$$\frac{2^n}{M} \leq 1 + W \leq \frac{2(t+2)}{2t+3-n}. \tag{14}$$

As a consequence, we get the following theorem.

THEOREM 7.3. *Suppose an $OA_\lambda(t, n, 2)$ exists, where t is even. Let $M = \lambda 2^t$ be the number of rows in the OA. Then*

$$M \geq 2^n - \frac{2^{n-1}(n+1)}{t+2}. \tag{15}$$

Note. The assumption that $2t > n - 3$ is not needed for the final statement of the theorem since otherwise the right-hand side of the above equation is not positive, and so the theorem will be trivially true.

The bound of Theorem 7.3 is exactly half of the bound given by Theorem 7.1 if (n, t) is replaced by $(n+1, t+1)$. So when t is even, the above bound is also implied by Theorem 7.1 in light of Theorem 6.1. This observation is made by Levenshtein [20].

We have the following corollary obtained by combining Theorems 7.1 and 7.3.

COROLLARY 7.4. *Suppose an $OA_\lambda(t, n, 2)$ exists. Let $M = \lambda 2^t$ be the number of rows in the OA. Then*

$$M \geq 2^n - \frac{2^{n-2}(n+1)}{\lceil \frac{t+1}{2} \rceil}. \tag{16}$$

Proof. Note that when t is odd,

$$2^n - \frac{2^{n-2}(n+1)}{\lceil \frac{t+1}{2} \rceil} = 2^n - \frac{2^{n-1}(n+1)}{t+1}.$$

The above bound is inferior to the bound given by Theorem 7.1 and hence is valid. On the other hand, when t is even,

$$2^n - \frac{2^{n-2}(n+1)}{\lceil \frac{t+1}{2} \rceil} = 2^n - \frac{2^{n-1}(n+1)}{t+2}.$$

This bound is valid in view of Theorem 7.3.

Corollary 7.4, together with equation (3), yields a second explicit bound on resilient functions. We state this bound as the following corollary.

COROLLARY 7.5. *If an (n, m, t)-resilient function exists, then*

$$t \leq 2 \left\lfloor \frac{2^{m-2}(n+1)}{2^m - 1} \right\rfloor - 1.$$

Several remarks are now in order. The statement in Corollary 7.2 was conjectured in [9]. It was shown to be true for $m = 1, 2$ (and in these cases, the bound is tight). The conjecture was also proved for arbitrary m in the special case of linear resilient functions, i.e., one where every output bit is a linear function of the input bits. Corollary 7.2 establishes that the conjecture is true for arbitrary m and for arbitrary resilient functions.

As stated earlier, Theorem 7.1 and Corollary 7.2 were previously proved by Friedman [12] in a different setting. His proof involved a study of colorings of the n-dimensional boolean cube and can be interpreted as bounds for orthogonal arrays as well. As shown in this section, these bounds can be derived in a straightforward way from the well-known linear programming bounds of coding theory.

Further, the classical Rao bound can also be obtained as the objective function of an appropriate feasible solution to the dual linear programming problem DLP1. Thus the linear programming technique gives a unified approach to prove the known general bounds on orthogonal arrays and resilient functions as well as new better bounds.

8. The power of explicit bounds. It is clear that the explicit bounds of inequalities (13) and (15) are trivial to compute. One might ask if these explicit bounds are weaker than the bounds computed by the linear programming technique. It turns out that for many parametric situations, these two bounds are as powerful as the linear programming bound itself. We will pursue this issue in this section.

THEOREM 8.1. *When t is odd and $t+1 \leq n \leq 2t+1$, the bound of equation* (13) *is as powerful as the linear programming bound.*

Proof. It has already been shown in equation (12) that

$$1 + W \leq \frac{2(t+1)}{2(t+1) - n}.$$

If we can also show that

$$1 + W \geq \frac{2(t+1)}{2(t+1) - n}$$

when t is odd and $t+1 \leq n \leq 2t+1$, then the proof is completed. We can indeed do so by exhibiting a feasible solution for the primal linear programming problem. Observe first that because of equation (10) the main condition in LP1 may equivalently be written as follows:

$$\sum_{i=t+1}^{n} \frac{w_i P_i(k)}{\binom{n}{i}} \geq -1 \text{ for } 1 \leq k \leq n.$$

Consider the solution vector given by

$$w_i = \begin{cases} \frac{n}{2(t+1)-n} & \text{if } i = t+1, \\ 0 & \text{otherwise.} \end{cases}$$

Because we are assuming $2(t+1) > n$, the nonnegativity is certainly satisfied. The main condition has been reformulated above. It says

$$w_{t+1} P_{t+1}(k) \geq -\binom{n}{t+1},$$

or equivalently,

$$P_{t+1}(k) \geq \binom{n}{t+1}\left[1 - \frac{2(t+1)}{n}\right].$$

That this is true is the content of the following lemma.

LEMMA 8.2. *When t is odd and $t+1 \leq n \leq 2t+1, 1 \leq i \leq n$, it is the case that*

$$P_{t+1}(i) \geq \binom{n}{t+1}\left[1 - \frac{2(t+1)}{n}\right].$$

Clearly the value of the objective function is as claimed. Although this lemma can be viewed as yet another property of the values of Krawtchouk polynomials at certain points, it does not seem to be known. The proof of this lemma is fairly technical and is not relevant to the main theme of this paper, and hence it is omitted here. However, the proof is made available in Appendix A for the interested reader.

Note that we are using the assumption that t is odd only in the proof of this lemma and not anywhere else.

A similar statement is made about the bound of equation (15) in the following theorem.

THEOREM 8.3. *When t is even and $t+1 \leq n \leq 2t+2$, the bound of equation (15) is as powerful as the linear programming bound.*

Proof. The proof will be along the same lines as that of Theorem 8.1. It has already been shown in equation (14) that

$$1 + W \leq \frac{2(t+2)}{2t+3-n}.$$

If we can also show the reverse inequality when t is even and $t+1 \leq n \leq 2t+2$, then the proof is completed. Once again, we can do so by exhibiting a feasible solution for the primal linear programming problem. Consider the solution vector given by

$$w_i = \begin{cases} \frac{t+2}{2t+3-n} & \text{if } i = t+1, \\ \frac{n-t-1}{2t+3-n} & \text{if } i = t+2, \\ 0 & \text{otherwise.} \end{cases}$$

Again nonnegativity is obvious by our assumptions. The value of the objective function is just right as

$$1 + w_{t+1} + w_{t+2} = 1 + \frac{n+1}{2t+3-n} = \frac{2(t+2)}{2t+3-n}.$$

We have to check the main condition of the primal problem in the reformulation given above. As

$$\frac{w_{t+1}}{\binom{n}{t+1}} = \frac{w_{t+2}}{\binom{n}{t+2}} = \frac{t+2}{(2t+3-n)\binom{n}{t+1}},$$

this condition is

$$\frac{t+2}{(2t+3-n)\binom{n}{t+1}}(P_{t+1}(k) + P_{t+2}(k)) \geq -1.$$

As $P_{t+1}(k) + P_{t+2}(k) = P_{t+2}^{n+1}(k)$ by equation (11) and

$$\binom{n}{t+1} = \binom{n+1}{t+2}\frac{t+2}{n+1},$$

this is equivalent to

$$P_{t+2}^{n+1}(k) \geq \binom{n+1}{t+2}\left[1 - \frac{2(t+2)}{n+1}\right].$$

This is just another instance of Lemma 8.2 proved in Appendix A.

We end this section with the following corollary, which is a simple consequence of Theorem 8.3.

COROLLARY 8.4. *When t is even and $n = t+1$, the bound given by equation* (13) *is the same as the one given by* (15)*, and hence is also as powerful as the linear programming bound.*

9. Optimal resilient functions for fixed m. Let us now focus on the special cases where m is a small fixed integer. When $m = 1$, the trivial upper bound of equation (2) says that $t \leq n-1$. The function $f : \{0,1\}^n \longrightarrow \{0,1\}$ defined by the rule $f(x_1, \ldots, x_n) = x_1 + \cdots + x_n$ is clearly an $(n, 1, n-1)$-resilient function. Thus, the optimal value of t is indeed $n-1$, when $m = 1$ (see [9]). When $m = 2$, it was proved in [9] that an (n, m, t)-resilient function exists if and only if $t < \lfloor \frac{2n}{3} \rfloor$. Partial results for $m \geq 3$ were found by Friedman [12]. As a consequence of our new bounds, we have completed the determination of the optimal resiliency of resilient functions with $m = 3$, and we have also done most of the cases for $m = 4$.

The *simplex code* $\mathcal{S}_m$ is the dual of the Hamming code $\mathcal{H}_m$. The generator matrix of $\mathcal{S}_m$ is of size $m \times 2^m - 1$ and has all nonzero binary m-tuples as its columns. It is not too difficult to see that each nonzero codeword of the simplex code has weight 2^{m-1}. Thus $\mathcal{S}_m$ is a linear constant-weight code with parameters $[2^m - 1, m, 2^{m-1}]$. We shall make use of the simplex code in the proof of the following theorem.

THEOREM 9.1. *Suppose there exists an (n, m, t) linear resilient function which is optimal by virtue of meeting one of the two bounds of Corollaries* 7.2 *and* 7.5*. Then there exists an $(n + 2^m - 1, m, t + 2^{m-1})$ linear resilient function which is also optimal by virtue of meeting the same bound.*

Proof. Since the (n, m, t)-resilient function is linear, it must have been constructed from an $[n, m, t+1]$ linear code $\mathcal{C}$. Now consider the linear code $\mathcal{C}'$ whose generating matrix is obtained by pasting the generating matrices of the code $\mathcal{C}$ and the simplex code $\mathcal{S}_m$, i.e.,

$$G_{\mathcal{C}'} = [G_{\mathcal{C}} \mid G_{\mathcal{S}_m}].$$

Clearly $\mathcal{C}'$ is a linear $[n + 2^m - 1, m, t + 1 + 2^{m-1}]$ code, and from this code we can construct a linear resilient function with parameters $(n + 2^m - 1, m, t + 2^{m-1})$.

By assumption, the (n, m, t)-resilient function is optimal as the parameters meet one of the two bounds derived in §7. If we increase n by $2^m - 1$, both the bounds go up by 2^{m-1}, and so it immediately follows that the linear resilient function with parameters $(n + 2^m - 1, m, t + 2^{m-1})$ is also optimal by virtue of meeting the same bound. □

The simplex code itself yields a $(2^m - 1, m, 2^{m-1} - 1)$ linear resilient function, which meets the bound of Theorem 7.2 and hence is optimal. Applying Theorem 9.1, we see that optimal linear resilient functions exist whenever $n \equiv 0 \bmod (2^m - 1)$, a result first shown by Friedman [12].

Let us now consider resilient functions with $m = 3$. The bound of Corollary 7.2 gives $t \leq \lfloor 4n/7 \rfloor - 1$ and the bound of Corollary 7.5 gives $t \leq 2\lfloor (2n+2)/7 \rfloor - 1$. The two bounds are tabulated in Table 3.

TABLE 3
A comparison of the two explicit bounds when $m = 3$.

n	$\lfloor 4n/7 \rfloor - 1$	$2\lfloor (2n+2)/7 \rfloor - 1$
$7h$	$4h-1$	$4h-1$
$7h+1$	$4h-1$	$4h-1$
$7h+2$	$4h$	$4h-1$
$7h+3$	$4h$	$4h+1$
$7h+4$	$4h+1$	$4h+1$
$7h+5$	$4h+1$	$4h+1$
$7h+6$	$4h+2$	$4h+3$

We see from the table that Corollary 7.2 gives the strongest bound for $n \equiv 3, 6 \bmod 7$, Corollary 7.5 gives the strongest bound for $n \equiv 2 \bmod 7$, and the two bounds are equal for $n \equiv 0, 1, 4, 5 \bmod 7$.

Of course these are only upper bounds on t. But matching lower bounds could be obtained as follows. In view of Theorem 4.1, whenever a linear $[n, m, d]$ code exists, so does an $(n, m, d-1)$-resilient function. When $m = 3$ and $4 \leq n \leq 10$, it so happens that the linear resilient functions constructed from the best known linear codes are optimal by virtue of meeting one of the two bounds of Corollaries 7.2 and 7.5. In fact, the bound of Corollary 7.2 is always met except when $n = 9$, in which case the bound of Corollary 7.5 is met. This fact coupled with Theorem 9.1 completely determines the optimal resiliency of resilient functions with $m = 3$. We record the result as the following theorem.

THEOREM 9.2. *The optimal resiliency* t *of resilient functions with* $m = 3$ *is*

$$\left\lfloor \tfrac{4n}{7} \right\rfloor - 1 \text{ if } n \not\equiv 2 \bmod 7 \quad \text{and} \quad \left\lfloor \tfrac{4n}{7} \right\rfloor - 2 \text{ if } n \equiv 2 \bmod 7.$$

Similar analysis settles the question for $m = 4$ in all cases except for two congruence classes modulo 15, as described in the following theorem.

THEOREM 9.3. *The optimal resiliency* t *of resilient functions with* $m = 4$ *is*

$$\left\lfloor \tfrac{8n}{15} \right\rfloor - 1 \text{ if } n \not\equiv 2, 3, 4, 6 \text{ or } 10 \bmod 15 \text{ and}$$

$$\left\lfloor \tfrac{8n}{15} \right\rfloor - 2 \text{ if } n \equiv 2, 6 \text{ or } 10 \bmod 15.$$

Here, it turns out that the bound of Corollary 7.2 is met in those cases when $n \not\equiv 2, 3, 4, 6, 10 \bmod 15$, and the bound of Corollary 7.5 is relevant for $n \equiv 2, 6, 10 \bmod 15$. The two congruence classes $n \equiv 3, 4 \bmod 15$ are unsolved at present.

9.1. Optimal resilient functions from anticodes. It is also possible to determine other infinite classes of optimal resilient functions by using the method of *anticodes* [22, Chap. 17, §6], and we do so in the following.

A well-known technique for constructing good linear codes is to take several copies of the generator matrix of the simplex code $\mathcal{S}_m$ and delete certain columns from it. The columns to be deleted constitute the generator matrix of an *anticode*. An anticode is a code which has an *upper bound* on the distance between its codewords. Note that an anticode may contain repeated codewords, since even a distance of zero between the codewords is allowed.

Consider an anticode whose generator matrix is of size $m \times k$ and whose *maximum distance* is δ. Suppose s is the maximum number of times any column occurs in the generating matrix of the anticode. Then we can form a new code whose generator matrix can be obtained by deleting the columns of the generator matrix of the anticode

from s copies of the generator matrix of $\mathcal{S}_m$ placed side-by-side. The new code has length $s(2^m - 1) - k$, dimension $\leq m$, and minimum distance $s2^{m-1} - \delta$.

Using the above general technique and anticodes formed from subspaces, Solomon and Stiffler [27, 2] constructed a large class of good linear codes. Let V_m denote a m-dimensional vector space over GF(2). Let $B_1, B_2, \ldots, B_p$ be subspaces of V_m of dimensions $u_1, u_2, \ldots, u_p$ such that no nonzero element of V_m is contained in more than s of the B_i. Let $m > u_1 \geq u_2 \geq \cdots \geq u_p \geq 1$. Then they showed that there exists an $[n, m, d]$ binary linear code, where

$$n = s(2^m - 1) - \sum_{i=1}^{p} (2^{u_i} - 1),$$

$$d \geq s2^{m-1} - \sum_{i=1}^{p} 2^{u_i - 1}.$$

Belov, Logachev, and Sandimirov [2] showed that a necessary and sufficient condition for the existence of such subspaces is

$$\sum_{i=1}^{\min(s+1,p)} u_i \leq sm. \tag{17}$$

We now proceed to determine the parametric situations where the resilient functions constructed from the linear codes obtained by the above construction meet the bound of Corollary 7.2 or 7.5, and so are optimal.

The resilient functions so constructed will meet the bound of Corollary 7.2 if and only if

$$\begin{aligned} s2^{m-1} - \sum_{i=1}^{p} 2^{u_i - 1} &= \left\lfloor \frac{2^{m-1} n}{2^m - 1} \right\rfloor \\ &= \left\lfloor \frac{2^{m-1} \left[s(2^m - 1) - \sum_{i=1}^{p} (2^{u_i} - 1) \right]}{2^m - 1} \right\rfloor \\ &= \left\lfloor s2^{m-1} - \frac{2^{m-1} \sum_{i=1}^{p} (2^{u_i} - 1)}{2^m - 1} \right\rfloor \\ &= s2^{m-1} - \left\lceil \frac{2^{m-1} \sum_{i=1}^{p} (2^{u_i} - 1)}{2^m - 1} \right\rceil. \end{aligned}$$

So, the bound of Corollary 7.2 will be met if and only if

$$\left\lceil \frac{2^{m-1}\sum_{i=1}^{p}(2^{u_i}-1)}{2^m-1} \right\rceil = \sum_{i=1}^{p} 2^{u_i-1}.$$

This could be equivalently expressed as

$$\sum_{i=1}^{p} 2^{u_i-1} - 1 < \frac{2^{m-1}\sum_{i=1}^{p}(2^{u_i}-1)}{2^m-1} \leq \sum_{i=1}^{p} 2^{u_i-1}.$$

The right inequality holds if and only if

$$\begin{aligned} & 2^{m-1}\sum_{i=1}^{p}(2^{u_i}-1) \leq (2^m-1)\sum_{i=1}^{p} 2^{u_i-1} \\ \iff & \sum_{i=1}^{p} 2^{m-1+u_i} - p2^{m-1} \leq \sum_{i=1}^{p} 2^{m-1+u_i} - \sum_{i=1}^{p} 2^{u_i-1} \\ \iff & \sum_{i=1}^{p} 2^{u_i-1} \leq p2^{m-1}. \end{aligned}$$

But the last statement indeed holds as $u_i < m$ for $1 \leq i \leq p$.

The left inequality holds if and only if

$$\begin{aligned} & \left(\sum_{i=1}^{p} 2^{u_i-1} - 1\right)(2^m-1) < 2^{m-1}\sum_{i=1}^{p}(2^{u_i}-1) \\ \iff & \sum_{i=1}^{p} 2^{m-1+u_i} - \sum_{i=1}^{p} 2^{u_i-1} - 2^m + 1 < \sum_{i=1}^{p} 2^{m-1+u_i} - p2^{m-1} \\ \iff & 2^{m-1}(p-2) < \sum_{i=1}^{p} 2^{u_i-1} - 1. \end{aligned}$$

The last statement clearly holds if $p = 1$ or $p = 2$, and so we will focus on them. When $p = 1$, the feasibility condition (17) reduces to $u_1 \leq m$, which is indeed the case. When $p = 2$, the feasibility condition (17) reduces to $u_1 + u_2 \leq sm$. This is always satisfied for $s \geq 2$ and satisfied only if $u_1 + u_2 \leq m$ for $s = 1$. Thus for each fixed m, we get many infinite classes of optimal resilient functions. We summarize the above discussion as the following theorem.

THEOREM 9.4. *Let m be a fixed integer. Let $m > u_1 \geq u_2 \geq 1$. Then for the following values of n, the upper bound on resiliency given in Corollary 7.2 is tight and it can be attained by linear resilient functions.*

$$\begin{aligned} & n = s(2^m-1) - 2^{u_1} + 1 \text{ for } s \geq 1, \\ & n = s(2^m-1) - 2^{u_1} - 2^{u_2} + 2 \\ & \qquad \text{for } s = 1 \text{ if } u_1 + u_2 \leq m \text{ and for } s \geq 2\ . \end{aligned}$$

A similar analysis yields the parametric situations where the bound of Corollary 7.5 is met by the resilient functions constructed using the above technique. We omit the details and record the final result as the following theorem.

THEOREM 9.5. *Let m be a fixed integer. Let $m > u_1 \geq u_2 \geq u_3 \geq 1$ and let the number of $u_i = 1$ be even. Then for the following values of n, the upper bound on resiliency given in Corollary 7.5 is tight and it can be attained by linear resilient functions.*

$$n = s(2^m - 1) - 2^{u_1} + 1 \text{ for } s \geq 1,$$
$$n = s(2^m - 1) - 2^{u_1} - 2^{u_2} + 2$$
$$\text{for } s = 1 \text{ if } u_1 + u_2 \leq m \text{ and for } s \geq 2,$$
$$n = s(2^m - 1) - 2^{u_1} - 2^{u_2} - 2^{u_3} + 3$$
$$\text{for } s = 1 \text{ if } u_1 + u_2 \leq m, \text{ for } s = 2 \text{ if } u_1 + u_2 + u_3 \leq 2m$$
$$\text{and for } s \geq 3.$$

As an example, consider $m = 4$. Then the dimensions of the subspaces must be $1, 2$, or 3. The number of nonzero vectors in these subspaces are, respectively, $1, 3$, and 7. Taking one at a time, we get optimal resilient functions when $n \equiv 8, 12, 14 \bmod 15$. Taking two at a time (with possible repetition), we get optimal resilient functions when $n \equiv 1, 5, 7, 9, 11, 13 \bmod 15$. These resilient functions meet the bound of Corollary 7.2. Taking three at a time (with possible repetition), we get the additional optimal resilient functions corresponding to $n \equiv 2, 6, 10 \bmod 15$, which meet the bound of Corollary 7.5. Note that in this case, we are not allowed to use subspaces of dimension one.

It should be noted that originally the theory of anticodes was developed to construct optimal linear codes which meet the Griesmer bound [15]. But, we used the theory here to construct linear optimal resilient functions.

Analogous results can be found in [12, Prop. 2.2] and [5, Thm. 5]. However, they were obtained from first principles without using the theory of anticodes. The above discussion is intended to illustrate the application of the existing theory of anticodes in the construction of linear optimal resilient functions and is not meant to be exhaustive.

10. Concluding remarks and open problems. In this paper, upper and lower bounds on the largest value of t such that an (n, m, t)-resilient function exists are derived. The upper bounds are derived from linear programming techniques and the lower bounds come from constructions based on error-correcting codes. Although we have determined exactly the optimal value of t for $1 \leq m < n \leq 17$, it is evident from Table 2 that there are still many parameter situations where the upper bounds and the lower bounds differ. It would be interesting to determine the optimal values in those cases as well. It is possible that constructions from some other nonlinear codes, augmented with our constructions of new resilient functions from §4.3, might do the trick. For example, if we could construct a $(21, 12, 5)$-resilient function from a systematic code, it would imply that $(20, 11, 5)$-, $(19, 10, 5)$-, $(20, 12, 4)$-, $(19, 11, 4)$-, and $(18, 10, 4)$-resilient functions exist and are optimal.

We derived two explicit bounds on the size of the orthogonal arrays using linear programming techniques in §7. It was also proved in §8 that these bounds are as powerful as the LP bound itself for certain parameter situations. But we still need to use the original LP bound to get tighter bounds for other parametric situations. It would be interesting to determine whether a set of explicit bounds could be developed using the LP bounds such that for any parametric situation at least one of them is

as powerful as the LP bound itself. If this can be done, then the original LP bound could be completely eliminated.

By applying our new bounds, we have completed the determination of the optimal resiliency of resilient functions with $m = 3$ in §9. Most of the cases for $m = 4$ were also done. One question pertinent to the remaining cases is whether $(18, 4, 8)$- and $(19, 4, 9)$-resilient functions exist or not. Actually, an $(18, 4, 8)$-resilient function exists if and only if a $(19, 4, 9)$-resilient function exists in view of Theorem 6.2, and so we will restrict our attention to one of them. Even if an $(18, 4, 8)$-resilient function exists, we cannot immediately extend it to an infinite class of optimal resilient functions using Theorem 9.1 since it has to be a nonlinear resilient function. Note that an $(18, 4, 7)$-resilient function is the best possible linear resilient function for this parameter situation. On the other hand, if an $(18, 4, 8)$-resilient function does not exist, then the linear $(18, 4, 7)$-resilient function would be an optimal one. Unfortunately, once again, we cannot extend it to an infinite class of optimal resilient functions using Theorem 9.1 since it is not optimal by virtue of meeting one of the two bounds of Corollaries 7.2 and 7.5 (which happens to be 8 and 9, respectively, in this case).

Although we have confined our discussion throughout this paper to binary orthogonal arrays and resilient functions for the sake of simplicity, most of the techniques used in this paper are applicable mutatis mutandis to nonbinary orthogonal arrays and resilient functions. As an example, the following bound on nonbinary orthogonal arrays was proved by Bierbrauer [5] by an algebraic method based on characters of homocyclic groups.

THEOREM 10.1. *If there exists an* $OA_\lambda(t, n, v)$, *then*

$$M \geq v^n \left(1 - \frac{(v-1)n}{v(t+1)}\right).$$

Several other results about nonbinary resilient functions can be found in [14].

Appendix A. A property of Krawtchouk polynomials. In this appendix we shall prove Lemma 8.2, which was used in §8 to prove the power of the explicit bounds derived in the paper. First we will state it in a slightly different form.

THEOREM A.1. *Let* k *be an even integer, let* $k \leq n \leq 2k - 1$, *and let* $1 \leq i \leq n$. *Then*

$$P_k(i) \geq \binom{n}{k} \left[1 - \frac{2k}{n}\right].$$

The right-hand side can be written in a handier combinatorial form:

$$\binom{n}{k} \left[1 - \frac{2k}{n}\right] = \binom{n}{k} - 2\binom{n-1}{k-1} = \binom{n-1}{k} - \binom{n-1}{k-1}.$$

The number $P_k(i)$ can also be expressed in a combinatorial fashion in the following way [21, Chap. 5, p. 64]:

$$P_k(i) = \sum_y (-1)^{<x,y>}.$$

Here x is a fixed binary vector of length n and weight i and the sum is over all the binary vectors of length n and weight k. This invites the following combinatorial interpretation.

Let S be an n-element set. Let X be a fixed subset of S of size i. Denote by $g = g(n,k,i)$ the number of subsets Y of S of size k such that $|X \cap Y|$ is even. Similarly, denote by $u = u(n,k,i)$, the number of subsets Y of S of size k such that $|X \cap Y|$ is odd. Then clearly

$$g(n,k,i) + u(n,k,i) = \binom{n}{k},$$
$$g(n,k,i) - u(n,k,i) = P_k(i).$$

Using these relations the claim of Theorem A.1 can be equivalently expressed in terms of the function g.

THEOREM A.2. *Let k be an even integer, let $k \leq n \leq 2k-1$, and let $1 \leq i \leq n$. Then*

$$g(n,k,i) \geq \binom{n-1}{k}.$$

LEMMA A.3. *The statement of Theorem* A.2 *is true when $i = n$.*

Proof. When $i = n$, any k-subset intersects X in k elements. As k is even, the number of k-subsets intersecting X in an even number of elements is simply $\binom{n}{k}$, which is clearly greater than or equal to $\binom{n-1}{k}$. □

So we may assume $1 \leq i \leq n-1$ in the sequel. For these cases, instead of proving the claim of Theorem A.2, we shall prove the following stronger claim.

THEOREM A.4. *Let $k \leq n \leq 2k-1$ and let $1 \leq i \leq n-1$. Then*

$$g(n,k,i) \geq \binom{n-1}{k}.$$

Proof. The proof is by induction on all three of the parameters n, k, and i. As $i \leq n-1$, the set $S - X$ is nonempty. Let $x \in S - X$. Considering those subsets Y of size k which exclude and include x, we can form the following recurrence relation.

$$g(n,k,i) = g(n-1,k,i) + g(n-1,k-1,i).$$

Then using the induction hypothesis we get

$$\begin{aligned} g(n,k,i) &\geq \binom{n-2}{k} + \binom{n-2}{k-1} \\ &= \binom{n-1}{k}. \end{aligned}$$

As long as the recurrence relation is applicable, it gives exactly what we want. It is also easy to see that the recurrence relation is not applicable exactly when either $n = k$ or $n = 2k-1$. So we have to take care of these two limiting cases.

LEMMA A.5. *The statement of Theorem* A.4 *is true when $k = n$.*

Proof. When $k = n$, $g(n,k,i) = g(n,n,i)$ is 1 if i is even and 0 if i is odd. But $\binom{n-1}{n} = 0$. Hence the statement of Theorem A.4 is true when $k = n$. □

The case of $n = 2k-1$ is a little bit more laborious and will be done through the following sequence of lemmas.

Let $n = 2k$. If i is odd, then of each complementary pair of k-subsets, precisely one will intersect X in even cardinality. Thus we get the following lemma.

LEMMA A.6.

$$g(2k,k,i)=\frac{1}{2}\binom{2k}{k}=\binom{2k-1}{k} \text{ if } i \text{ is odd.}$$

Using the same trick in the case of odd n ($n=2k-1$), we get the following lemma.

LEMMA A.7.

$$g(2k-1,k-1,i)+g(2k-1,k,i)=\binom{2k-1}{k} \text{ for odd } i.$$

As $i \geq 1$, the fixed subset X is nonempty. Let $x \in X$. Considering those subsets Y of size k which exclude and include x, we can form the following recurrence relation.

$$\begin{aligned} g(n,k,i) &= g(n-1,k,i-1)+u(n-1,k-1,i-1) \\ &= \binom{n-1}{k-1}-g(n-1,k-1,i-1)+g(n-1,k,i-1). \end{aligned}$$

LEMMA A.8. *If i is even, then*

$$g(2k,k,i)=2g(2k-1,k,i-1).$$

Proof. By the second recurrence relation, we have

$$g(2k,k,i)=\binom{2k-1}{k-1}+g(2k-1,k,i-1)-g(2k-1,k-1,i-1).$$

As $i-1$ is odd, we can use Lemma A.7 to get

$$g(2k-2,k-1,i-1)=\binom{2k-1}{k}-g(2k-1,k,i-1).$$

Further simplification establishes the lemma. □

LEMMA A.9. *If i is even, then $g(2k,k,i) \geq 2\binom{2k-2}{k}$.*

Proof. By Lemma A.8 we have

$$g(2k,k,i)=2g(2k-1,k,i-1).$$

Using the first recursion, we get

$$g(2k,k,i)=2\{g(2k-2,k,i-1)+g(2k-2,k-1,i-1)\}.$$

Since $i-1$ is odd, we can use Lemma A.6 to get

$$2g(2k-2,k-1,i-1)=\binom{2k-2}{k-1}.$$

It follows that

$$g(2k,k,i)=\binom{2k-2}{k-1}+2g(2k-2,k,i-1).$$

We can now use induction on i to assume the validity of the claim of Theorem A.4 in the case of the last term (note that the parameter k also satisfies the desired conditions) and get

$$g(2k,k,i) \geq \binom{2k-2}{k-1} + 2\binom{2k-3}{k} = 2\binom{2k-2}{k}.$$

This is exactly what was claimed. □

The above lemma immediately yields the following corollary in view of Lemma A.8.

COROLLARY A.10. *The statement in Theorem* A.4 *is true in the case* $n = 2k-1$ *when* i *is odd.*

LEMMA A.11. *The statement in Theorem* A.4 *is true in the case* $n = 2k-1$ *when* i *is even.*

Proof. By the second recurrence relation, we have

$$g(2k-1,k,i) = \binom{2k-2}{k-1} + g(2k-2,k,i-1) - g(2k-2,k-1,i-1).$$

Since $i-1$ is odd, we can use Lemma A.6 to get

$$g(2k-2,k-1,i-1) = \binom{2k-3}{k-1}.$$

It follows that

$$g(2k-1,k,i) = \binom{2k-3}{k-1} + g(2k-2,k,i-1).$$

Once again, we can use induction on i to assume the validity of the claim of Theorem A.4 in the case of the last term and get

$$g(2k-1,k,i) \geq \binom{2k-3}{k-1} + \binom{2k-3}{k} = \binom{2k-2}{k}. \qquad \square$$

Finally, we need to take care of the case $i = 1$. This is to assure the basis of induction since it is done over i also.

LEMMA A.12. *The statement of Theorem* A.4 *is true when* $i = 1$.

Proof. When $i = 1$, clearly

$$g(n,k,i) = g(n,k,1) = \binom{n-1}{k}.$$

Thus the claim of Theorem A.4 is trivially satisfied. □

It would be nice to have a combinatorial proof of Theorem A.4 instead of the above lengthy proof by induction through multiple parameters. We leave this as an open problem.

Acknowledgments. We would like to thank V. Levenshtein for several helpful discussions and the referee for carefully reading the manuscript and making some helpful suggestions.

REFERENCES

[1] T. ATSUMI, *A study of orthogonal arrays from the point of view of design theory*, J. Combin. Theory A, 35 (1983), pp. 241–251.

[2] B. I. BELOV, V. N. LOGACHEV, AND V. P. SANDIMIROV, *Construction of a class of linear binary codes achieving the Varshamov-Griesmer bound*, Problems Info. Trans., 10 (1974), pp. 211–217.

[3] C. H. BENNETT, G. BRASSARD, AND A. K. EKERT, *Quantum cryptography*, Scientific American, 267 (1992), pp. 26–33.

[4] C. H. BENNETT, G. BRASSARD, AND J. M. ROBERT, *Privacy amplification by public discussion*, SIAM J. Comput., 17 (1988), pp. 210–229.

[5] J. BIERBRAUER, *Bounds on orthogonal arrays and resilient functions*, J. Combin. Designs, 3 (1995), pp. 179–183.

[6] A. E. BROUWER AND T. VERHOEFF, *An updated table of minimum-distance bounds for binary linear codes*, IEEE Trans. Inform. Theory, 39 (1993), pp. 662–677.

[7] K. A. BUSH, *Orthogonal arrays of index unity*, Ann. Math. Statist., 23 (1952), pp. 426–434.

[8] P. CAMION, C. CARLET, P. CHARPIN, AND N. SENDRIER, *On correlation-immune functions*, in Advances in Cryptology - CRYPTO '91, Springer-Verlag, Heidelberg, 1992, pp. 86–100.

[9] B. CHOR, O. GOLDREICH, J. HÅSTAD, J. FRIEDMAN, S. RUDICH, AND R. SMOLENSKY, *The bit extraction problem or t-resilient functions*, in Proc. 26th IEEE Symp. on Foundations of Computer Science, IEEE Computer Society, Piscataway, NJ, 1985, pp. 396–407.

[10] P. DELSARTE, *Bounds for unrestricted codes, by linear programming*, Philips Research Reports, 27 (1972), pp. 272–289.

[11] ———, *Four fundamental parameters of a code and their combinatorial significance*, Inform. and Control, 23 (1973), pp. 407–438.

[12] J. FRIEDMAN, *On the bit extraction problem*, in Proc. 33rd IEEE Symp. on Foundations of Computer Science, IEEE Computer Society, Piscataway, NJ, 1992, pp. 314–319.

[13] K. GOPALAKRISHNAN, D. G. HOFFMAN, AND D. R. STINSON, *A note on a conjecture concerning symmetric resilient functions*, Inform. Process. Lett., 47 (1993), pp. 139–143.

[14] K. GOPALAKRISHNAN AND D. R. STINSON, *Three characterizations of non-binary correlation-immune and resilient functions*, Designs, Codes and Cryptography, 5 (1995), pp. 241–251.

[15] J. H. GRIESMER, *A bound for error correcting codes*, IBM J. Res. Develop., 4 (1960), pp. 532–542.

[16] A. S. HEDAYAT, N. J. A. SLOANE, AND J. STUFKEN, *Orthogonal Arrays: Theory and Practice*, manuscript.

[17] A. S. HEDAYAT AND J. STUFKEN, *Two-symbol orthogonal arrays*, in Optimal Design and Analysis of Experiments, Y. Dodge, V. V. Federov, and H. P. Wynn, eds., North-Holland, Amsterdam, 1988, pp. 47–58.

[18] V. LEVENSHTEIN, *Krawtchouk polynomials and universal bounds for codes and designs in Hamming spaces*, IEEE Trans. Inform. Theory, 41 (1995), pp. 1303–1321.

[19] ———, *Bounds for self-complementary codes and their applications*, in CISM Courses and Lectures - Eurocode 92, Springer-Verlag, Berlin, 1993, pp. 159–171.

[20] ———, Private communication, 1994.

[21] J. H. VAN LINT, *Introduction to Coding Theory*, Springer-Verlag, Berlin, 1982.

[22] F. J. MACWILLIAMS AND N. J. A. SLOANE, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[23] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization - Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[24] C. R. RAO, *Factorial experiments derivable from combinatorial arrangements of arrays*, J. Royal Statist. Soc., 9 (1947), pp. 128–139.

[25] R. A. RUEPPEL, *Analysis and Design of Stream Ciphers*, Springer-Verlag, Berlin, 1986.

[26] E. SEIDEN AND R. ZEMACH, *On orthogonal arrays*, Ann. Math. Stat., 37 (1966), pp. 1355–1370.

[27] G. SOLOMON AND J. J. STIFFLER, *Algebraically punctured cyclic codes*, Inform. Control, 8 (1965), pp. 170–179.

[28] D. R. STINSON, *Resilient functions and large sets of orthogonal arrays*, Congr. and Numer., 92 (1993), pp. 105–110.

[29] D. R. STINSON AND J. L. MASSEY, *An infinite class of counterexamples to a conjecture concerning non-linear resilient functions*, J. Cryptology, 8 (1995), pp. 167–173.

[30] L. TEIRLINCK, *Large sets of disjoint designs and related structures*, in Contemporary Design Theory - A Collection of Surveys, J. H. Dinitz and D. R. Stinson, eds., John Wiley, New York, 1992, pp. 564–567.

SIAM J. DISCRETE MATH.
Vol. 9, No. 3, pp. 453–491, August 1996

008

SECOND-ORDER RIGIDITY AND PRESTRESS STABILITY FOR TENSEGRITY FRAMEWORKS*

ROBERT CONNELLY† AND WALTER WHITELEY‡

Abstract. This paper defines two concepts of rigidity for tensegrity frameworks (frameworks with cables, bars, and struts): prestress stability and second-order rigidity. We demonstrate a hierarchy of rigidity—first-order rigidity implies prestress stability implies second-order rigidity implies rigidity—for any framework. Examples show that none of these implications are reversible, even for bar frameworks. Other examples illustrate how these results can be used to create rigid tensegrity frameworks.

This paper also develops a duality for second-order rigidity, leading to a test which combines information on the self stresses and the first-order flexes of a framework to detect second-order rigidity. Using this test, the following conjecture of Roth is proven: a plane tensegrity framework, in which the vertices and bars form a strictly convex polygon with additional cables across the interior, is rigid if and only if it is first-order rigid.

Key words. tensegrity frameworks, rigid and flexible frameworks, stability of frameworks, static stress, first-order motion, second-order motion

AMS subject classifications. Primary, 52C25; Secondary, 70B15, 70C20

1. Introduction.

A fundamental problem in geometry is to determine when selected distance constraints, on a finite number of points, fix these points up to congruence, at least for small perturbations. We rephrase this as a problem in the rigidity of frameworks. From this point of view, we do the following:

(i) provide methods for recognizing when a given framework is rigid;
(ii) find ways of generating rigid frameworks;
(iii) explore the relationship among these methods;
(iv) solve a conjecture of Roth, in [29], concerning the rigidity of a specific class of frameworks in the plane.

Many of the concepts here are inspired by techniques used in structural engineering, such as the principle of least work and energy, but our treatment is independent of any such concepts. Broadly speaking our objective in this paper is to investigate the geometric properties of configurations of points in Euclidean space. However, our results clarify and justify mathematically some of the techniques used by structural engineers to analyze certain tensegrity structures. (See [5, 27, 28] for example.) Our techniques extend those of [21] and are related to the questions posed by Tarnai [31]. A summary of these results, for engineers, was presented in [13].

1.1. Terminology.

A *tensegrity framework* is an ordered finite collection of points in Euclidean space, called a *configuration*, with certain pairs of these points,

*Received by the editors April 13, 1992; accepted for publication (in revised form) October 10, 1995.

†Department of Mathematics, Cornell University, Ithaca, NY, 14853-7901 (connelly@math.cornell.edu). The work of this author was supported in part by National Science Foundation grant MCS-790251, Distinguished Visitorship, Centre de Recherches Mathematiques, Université de Montréal, and the Humboldt research award program.

‡Department of Mathematics and Statistics, York University, 4700 Keele Street, North York, Ontario M3J-1P3, Canada (whiteley@mathstat.yorku.ca). The work of this author was supported in part by grants from Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (Québec) and Natural Sciences and Engineering Research Council of Canada.

called *cables*, constrained not to get further apart; certain pairs, called *struts*, constrained not to get closer together; and certain pairs, called *bars*, constrained to stay the same distance apart. Together, struts, cables, and bars are called *members*. If each continuous motion of the points satisfying all the constraints is the restriction of a rigid motion of the ambient Euclidean space, then we say the tensegrity framework is *rigid*. See [9] and [29].

For the recognition problem (i) there has been much work done using the concept of first-order rigidity. A tensegrity framework is *first-order rigid* (or *infinitesimally rigid*) if the only smooth motion of the vertices, such that the first derivative of each member length is consistent with the constraints, has its derivative at time zero equal to that of the restriction of a congruent motion of Euclidean space. See §2.2 for more details. An equivalent dual concept says that a tensegrity framework is *statically rigid* if every equilibrium load can be resolved. See [10] or [29] for more details and a precise definition. Our working definition will be that of first-order rigidity as above.

A *stress* in a tensegrity framework is an assignment of a scalar to each member. It is called a *self stress* if the vector sum of the scalar times the corresponding member vector is zero at each vertex. It is called *proper* if the cable stresses are nonnegative and the strut stresses are nonpositive (with no condition on the bars). It is called *strict* if the stress in each cable and strut is nonzero.

1.2. First-order duality. The interplay between first-order motions and self stresses yields a test for first-order rigidity. Every first-order rigid tensegrity framework has a strict proper self stress, by a result of Roth and Whiteley [29]. We state the first-order stress test as follows: *There is a first-order flex of a framework which strictly changes the length of a strut or cable if and only if every proper self stress is zero on the given member.*

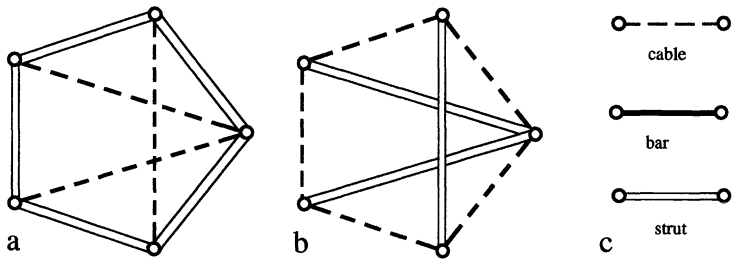


FIG. 1. *Two first-order rigid tensegrity frameworks* (a, b) *where cables and struts are reversed.*

In addition, if a first-order rigid bar framework has any nonzero self stress at all, one can change these members to cables or struts following the sign of the self stress to get another statically rigid tensegrity framework. In the spirit of (ii), this is a first-order method for generating examples of statically rigid tensegrity frameworks. See [29] and [35] for examples. Note that any first-order rigid tensegrity framework can have its cables and struts reversed to struts and cables, respectively, and it will remain first-order rigid. Figure 1 shows a pair of frameworks which are infinitesimally rigid in the plane.

1.3. Prestress stability and second-order rigidity. In this paper, we define two other classes of frameworks, those that are prestress stable and those that are second-order rigid. We call a tensegrity framework *prestress stable* if it has a proper strict self stress such that a certain energy function, defined in terms of the stress and defined for all configurations, has a local minimum at the given configuration, and this

minimum is a strict local minimum up to congruence of the whole framework. (See §3.3 for the precise formula for this energy function.)

Prestress stability is a concept we have borrowed from structural engineering. The "principle of least work" is the motivation behind the definition of our energy functions. If a certain configuration of a framework corresponds to a local minimum (modulo rigid motions) of an energy function, which is the sum of the energies of all the members, then it is clear that the framework is rigid. When the usual second derivative test detects such a minimum, this corresponds to prestress stability. Pellegrino and Calladine [28] describe certain matrix rank conditions that are necessary but not sufficient for prestress stability. However, their condition essentially ignores the basic positive definite conditions. See [5] for an improved version, though. For engineering calculations, the stress-strain relation in each member is given, and this information determines the corresponding energy function. On the other hand, for the simpler mathematical recognition problem (i), one is free to choose the member energy functions at will.

A tensegrity framework is *second-order rigid* if every smooth motion of the vertices, which does not violate any member constraint in the first and second derivative, has its first derivative trivial; i.e., its first derivative is the derivative of a one parameter family of congruent motions.

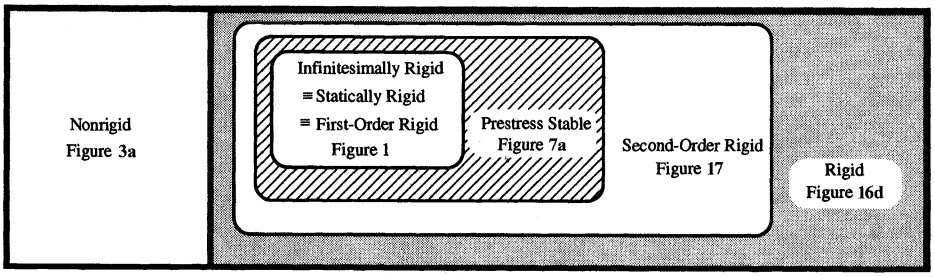


FIG. 2. *A diagram of the hierarchy of "rigidity" (numbers refer to illustrative examples).*

A series of basic results shows that for any tensegrity framework, first-order rigidity (i.e., infinitesimal rigidity) implies prestress stability, which implies second-order rigidity, which implies rigidity, and none of these implications can be reversed. See Figure 2, where the figure numbers refer to examples seen later in this paper that lie only in that region of the diagram. This extends the second-order rigidity results of [6] for bar frameworks and places prestress stability between first-order and second-order rigidity.

1.4. The second-order stress test. Information about a framework, or a class of frameworks, may come in various forms, and it can be useful to relate these different forms for the situation at hand. For example, to test the first-order rigidity of a tensegrity framework we may use both the self stresses and the first-order flexes, as in the first-order stress test.

We extend this first-order duality to the second-order situation. Regard any stress as the constant coefficients of a quadratic form on the space of all configurations as well as the space of first-order flexes. This is a "homogeneous" energy function. Suppose we have a fixed first-order flex of a given framework, and we wish to know when that first-order flex extends to some second-order flex. Our second-order stress test states the following: *A second-order flex exists if and only if for every proper self stress of the framework the quadratic form it defines is nonpositive when evaluated at the given*

first-order flex. Thus information about proper self stresses of a framework, as well as first-order flexes, can provide information about second-order rigidity. The proof amounts to observing that the (inequality and equality) constraints of second-order rigidity and our dual stress condition is a special case of the "Farkas alternative" (as used in linear programming duality).

It is also possible to sharpen the second-order stress test to provide necessary and sufficient conditions to detect when the second-order inequalities are strict. This sharpening is a generalization of the first-order stress test. The sharpened second-order stress test can be helpful not only in detecting second-order rigidity but also quite often in detecting when there is an actual continuous flex that has the cable and strut conditions slacken at the second-order.

1.5. Roth's conjecture. As an application of these methods we verify a conjecture of Roth about polygons in the plane in [29]. In their *Lectures on Lost Mathematics* [18, 19], Grünbaum and Shepard conjectured that if one has a framework $G(\mathbf{p})$ in the plane with the points as the vertices of a convex polygon, bars on the edges, and cables inside connecting certain pairs of the vertices (Figure 3a, c) in such a way that the framework is rigid in the plane (Figure 1a), then reversing the cables and bars to get $\hat{G}(\mathbf{p})$ (Figure 1b) preserves rigidity. (They also observed that starting with cables on the outside and bars inside does not necessarily preserve rigidity (Figure 3b, a).)

If Grünbaum and Shephard's polygonal frameworks are rigid because they are infinitesimally rigid, then it follows that the reversed framework is also infinitesimally rigid and therefore rigid. Roth's conjecture was that all rigid convex polygons with cables on the inside were indeed infinitesimally rigid. For example, Figure 3c shows a regular octagon with bars on the edges and fourteen cables on the inside. It is easy to check that this framework is not infinitesimally rigid. Thus Roth's conjecture implies that this framework is not rigid since if it were rigid, it would be infinitesimally rigid. The reader is invited to find the motion of the vertices in the plane directly. (See Remark 6.2.1.)

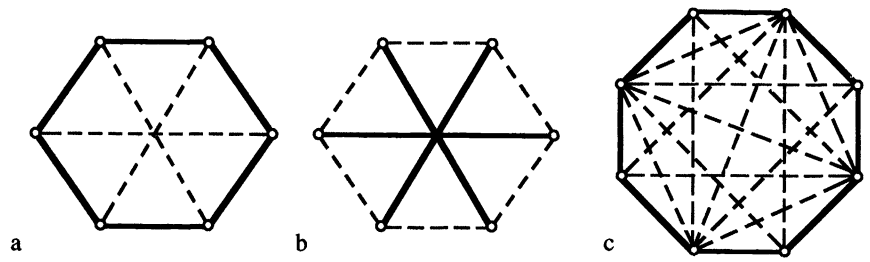


FIG. 3. *Framework* (b) *is rigid, but framework* (a), *where cables and struts are reversed, is not. Framework* (c) *is not first-order rigid and is not rigid, illustrating Roth's conjecture.*

Meanwhile, Connelly [7] showed that any proper self stress coming from one of Grünbaum and Shepard's polygonal frameworks $G(\mathbf{p})$ had an associated negative semidefinite quadratic form with nullity three. Equivalently the reversed framework $\hat{G}(\mathbf{p})$ had a positive semidefinite quadratic form with nullity three. Then it is easy to show that $\hat{G}(\mathbf{p})$ is rigid by showing that (globally) there is no other noncongruent configuration satisfying the bar and cable constraints. This global type of rigidity is somewhat different from infinitesimal rigidity. Neither infinitesimal rigidity nor global rigidity implies the other. However, the energy functions used to prove the global rigidity also imply prestress stability. Thus Grünbaum's conjecture was proved

and generalized in one direction, but Roth's conjecture, a generalization in another direction, was not proved.

1.6. The proof of Roth's conjecture. The idea behind our proof of Roth's conjecture is the following. We observe that the conditions for a strict second-order flex, in the second-order stress test, are satisfied by any one of Grünbaum and Shepard's frameworks $G(\mathbf{p})$, since any proper self stress defines a negative semidefinite quadratic form which is negative definite on any space of nontrivial first-order flexes. Thus if there is any nontrivial first-order flex, it will extend to a strict second-order flex which in turn implies that there will be a nontrivial continuous flex of the framework. Thus (the contrapositive of) Roth's conjecture is verified: if $G(\mathbf{p})$ is not infinitesimally rigid, then $G(\mathbf{p})$ is not rigid. In particular, if any one of Grünbaum and Shepard's frameworks $G(\mathbf{p})$ is rigid, the reversed framework $\hat{G}(\mathbf{p})$ is both infinitesimally rigid and globally rigid.

In an appendix we summarize a series of "replacement principles" which describe when and how one can switch between bars and cables or struts and preserve the various levels of rigidity or flexibility.

2. Review of tensegrity frameworks. Throughout this paper the word tensegrity is used to describe any framework with *cables*—each cable determines a maximum distance between two points, *struts*—each strut determines a minimum distance between two points, and *bars*—each bar determines a fixed distance between two points. Statically, cables can only apply tension and struts can only apply compression. We partition the edges of our graph into three disjoint classes — E_- for cables, E_0 for bars, and E_+ for struts, creating a *signed graph* $G = (V; E_-, E_0, E_+)$. In our figures, cables are indicated by dashed lines, struts by double thin lines, and bars by single thick lines (see Figure 1). General references for this chapter are [8, 9, 29, 35].

2.1. Rigidity.

DEFINITION 2.1.1. *A* tensegrity framework *in d-space $G(\mathbf{p})$ is a signed graph $(V; E_-, E_0, E_+)$, and an assignment $\mathbf{p} \in \mathbb{R}^{dv}$ such that each $\mathbf{p}_i \in \mathbb{R}^d$ corresponds to a vertex of G, where $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_v)$ is a* configuration. *The members in E_- are cables, the members in E_0 are bars, and the members in E_+ are struts. A* bar framework *is a tensegrity framework with no cables or struts; i.e., $E = E_0$.*

DEFINITION 2.1.2. *A tensegrity framework $G(\mathbf{p})$* dominates *the tensegrity framework $G(\mathbf{q})$, written $G(\mathbf{p}) \geq G(\mathbf{q})$, if*

$$\begin{aligned} |\mathbf{p}_i - \mathbf{p}_j| &\geq |\mathbf{q}_i - \mathbf{q}_j| \quad \text{when} \quad \{i,j\} \in E_-, \\ |\mathbf{p}_i - \mathbf{p}_j| &= |\mathbf{q}_i - \mathbf{q}_j| \quad \text{when} \quad \{i,j\} \in E_0, \\ |\mathbf{p}_i - \mathbf{p}_j| &\leq |\mathbf{q}_i - \mathbf{q}_j| \quad \text{when} \quad \{i,j\} \in E_+. \end{aligned}$$

A tensegrity framework $G(\mathbf{p})$ is rigid *in $\mathbb{R}^d$ if any of the following three equivalent conditions holds* [9] *or* [29]*:*

(a) *there is an $\varepsilon > 0$ such that if $G(\mathbf{p}) \geq G(\mathbf{q})$ and $|\mathbf{p} - \mathbf{q}| < \varepsilon$ then $\mathbf{p}$ is congruent to $\mathbf{q}$; or*

(b) *for every continuous path, or* continuous flex, *$\mathbf{p}(t) \in \mathbb{R}^{vd}$, $\mathbf{p}(0) = \mathbf{p}$, such that $G(\mathbf{p}) \geq G(\mathbf{p}(t))$ for all $0 \leq t \leq 1$, then $\mathbf{p}$ is congruent to $\mathbf{p}(t)$ for all $0 \leq t \leq 1$; or*

(c) *for every analytic path, or* analytic flex, *$\mathbf{p}(t) \in \mathbb{R}^{vd}$, $\mathbf{p}(0) = \mathbf{p}$, such that $G(\mathbf{p}) \geq G(\mathbf{p}(t))$ for all $0 \leq t \leq 1$, then $\mathbf{p}$ is congruent to $\mathbf{p}(t)$ for all $0 \leq t \leq 1$.*

2.2. First-order rigidity.

DEFINITION 2.2.1. *A* first-order flex, *or an* infinitesimal flex, *of a tensegrity framework* $G(\mathbf{p})$ *is an assignment* $\mathbf{p}' : V \to \mathbb{R}^n$, $\mathbf{p}'(v_i) = \mathbf{p}'_i$, *such that for each edge* $\{i,j\} \in E$ *(Figure* 4*),*

$$\begin{aligned}
(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}'_j - \mathbf{p}'_i) &\leq 0 \quad \textit{for cables} \quad \{i,j\} \in E_-,\\
(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}'_j - \mathbf{p}'_i) &= 0 \quad \textit{for bars} \quad \{i,j\} \in E_0,\\
(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}'_j - \mathbf{p}'_i) &\geq 0 \quad \textit{for struts} \quad \{i,j\} \in E_+.
\end{aligned}$$

The dot product of two vectors X, Y is indicated by XY or $X \cdot Y$.

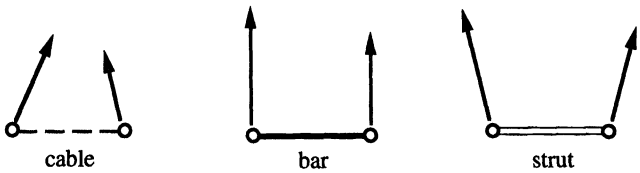


FIG. 4. *Some first-order motions (velocities) permitted by cables, bars, and struts.*

DEFINITION 2.2.2. *A first-order flex* $\mathbf{p}'$ *of a tensegrity framework* $G(\mathbf{p})$ *is* trivial *if there is a skew symmetric matrix* S *and a vector* $\mathbf{t}$ *such that* $\mathbf{p}'_i = S\mathbf{p}_i + \mathbf{t}$ *for all vertices* i.

DEFINITION 2.2.3. *A tensegrity framework* $G(\mathbf{p})$ *is* first-order rigid *(or* infinitesimally rigid*) if every first-order flex is trivial and* first-order flexible *otherwise.*

Let

$$\mathbf{p}^* = \begin{bmatrix} \mathbf{p}_1^* \\ \vdots \\ \mathbf{p}_v^* \end{bmatrix}$$

be regarded as a column vector in $\mathbb{R}^{dv}$, where each $\mathbf{p}_i^* \in \mathbb{R}^d$, $i = 1, \ldots, v$. Then $R(\mathbf{p})$ is the e-by-dv matrix defined by

$$R(\mathbf{p})\mathbf{p}^* = \begin{bmatrix} \vdots \\ (\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*) \\ \vdots \end{bmatrix}.$$

See [9] or [29]. $R(\mathbf{p})$ is called the *rigidity matrix* for the framework $G(\mathbf{p})$. Notice that a first-order flex of a bar framework is a solution to the linear equations

$$R(\mathbf{p})\mathbf{p}^* = 0.$$

Remark 2.2.1. A basic theorem of the subject says that *first-order rigidity for a tensegrity framework implies rigidity for the framework.* (See [9, 29].)

DEFINITION 2.2.4. *A first-order flex* $\mathbf{p}'$ *is an* equilibrium flex *if* $\mathbf{p}' \cdot \mathbf{q}' = 0$ *for all trivial first-order flexes* $\mathbf{q}'$.

Physically, a first-order flex is a velocity vector field associated with the configuration, and it turns out that an equilibrium flex is a vector field such that the linear and angular momentum is preserved (Figure 5).

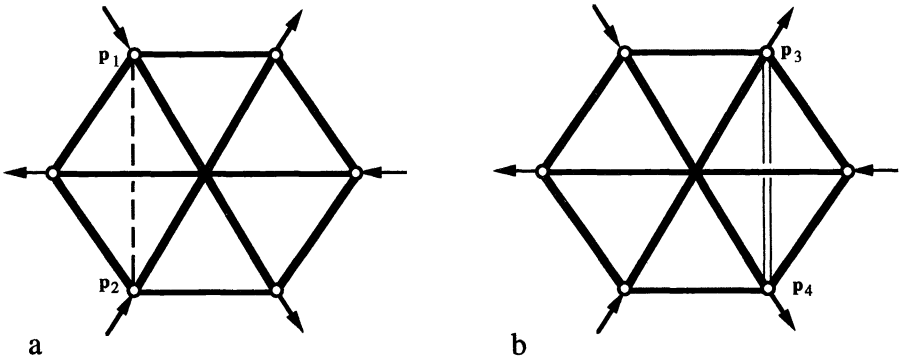


FIG. 5. *The cable* $(\mathbf{p}_1, \mathbf{p}_2)$ *in example* (a) *is unstressed because it can be shortened, while the strut* $(\mathbf{p}_3, \mathbf{p}_4)$ *in* (b) *is unstressed because it can be lengthened.*

2.3. Stresses.

DEFINITION 2.3.1. *A* stress ω *on a tensegrity framework* $G(\mathbf{p})$ *is an assignment of scalars* $\omega_{ij} = \omega_{ji}$ *to the edges of* G, *where* $\omega = (\ldots, \omega_{ij}, \ldots) \in \mathbb{R}^e$, *and* e *is the number of edges of* G.

A stress ω *on a tensegrity framework is a* self stress *if the following equilibrium condition holds at each vertex* i:

$$\sum_j \omega_{ij}(\mathbf{p}_j - \mathbf{p}_i) = 0,$$

where the sum is taken over all j *with* $\{i, j\} \in E$.

A self stress ω *is called a* proper self stress *if* (a) $\omega_{ij} \geq 0$ *for cables* $\{i, j\} \in E_-$ *and* (b) $\omega_{ij} \leq 0$ *for struts* $\{i, j\} \in E_+$.
There is no condition for a bar.

A proper self stress ω *is* strict *if the inequalities in* (a) *and* (b) *are strict.*

With this notation a self stress ω is a solution to the linear equations $\omega R(\mathbf{p}) = \mathbf{0}$, and ω is a proper self stress if each of the ω_{ij} corresponding to cables and struts have the proper sign. The following is shown in [35].

THEOREM 2.3.2 (first-order stress test). *Let* $G(\mathbf{p})$ *be a tensegrity framework, where* $\{i, j\}$ *is a fixed cable or strut. There is a first-order flex* $\mathbf{p}'$ *for* $G(\mathbf{p})$ *such that*

$$(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}'_i - \mathbf{p}'_j) \neq 0,$$

which means that a cable $\{\mathbf{p}_i, \mathbf{p}_j\}$ *is shortened or a strut* $\{\mathbf{p}_i, \mathbf{p}_j\}$ *is lengthened, to first-order, if and only if for every proper self stress* ω *for* $G(\mathbf{p})$ *has* $\omega_{ij} = 0$.

It is helpful to change the presentation of a stress. Let $\omega = (\ldots, \omega_{ij}, \ldots) \in \mathbb{R}^e$ be a stress for $G(\mathbf{p})$. Define a v-by-v symmetric matrix, the *reduced stress matrix* $\overline{\Omega}$, by setting the (i, j) entry to be

$$\overline{\Omega}_{ij} = \begin{cases} -\omega_{ij} & \text{if} \quad i \neq j, \\ \sum_k \omega_{ik} & \text{if} \quad i = j\ . \end{cases}$$

Denoting by p_{ik} the kth coordinate of $\mathbf{p}_i \in \mathbb{R}^d$, $k = 1, \ldots, d$, the expression

$$\sum_{k=1}^{d} \begin{bmatrix} p_{1k}, \ldots, p_{vk} \end{bmatrix} \overline{\Omega} \begin{bmatrix} p_{1k} \\ \vdots \\ p_{vk} \end{bmatrix}$$

is a quadratic form on the vd-dimensional space of coordinates of all points of the configuration. Reordering the coordinates by the order of the points yields

$$\left[\mathbf{p}_1^T, \ldots, \mathbf{p}_v^T\right]\Omega \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_v \end{bmatrix},$$

where $(\,)^T$ represents the transpose operation. This defines the *stress matrix* Ω, which is, up to permutation of the coordinates of $\mathbf{p}$, just k "copies" of $\overline{\Omega}$. It is easy to check that if $\mathbf{p}, \mathbf{q} \in \mathbb{R}^{vd}$, $\omega \in \mathbb{R}^e$, then

$$\omega R(\mathbf{p})\mathbf{q} = \mathbf{p}^T\Omega\mathbf{q} = \sum_{ij} \omega_{ij}(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{q}_i - \mathbf{q}_j). \tag{1}$$

Thus Ω is just the matrix of a bilinear form in the coordinates of $\mathbf{p}$ and $\mathbf{q}$, and it turns out that ω is a self stress if $\mathbf{p}^T\Omega = 0$.

3. Prestress stability.

3.1. The energy principle. If a cable is stretched, the energy in the cable increases. Similarly, if a strut is shortened, or a bar is changed in length, the energy increases. We put these together in the following energy function:

$$H^*(\mathbf{q}) = \sum_{ij} f_{ij}\left(|\mathbf{q}_j - \mathbf{q}_i|^2\right), \tag{2}$$

where

$$\begin{array}{lllll} & \text{for each cable} & \{i,j\}, & f_{ij} & \text{is strictly monotone increasing,} \\ (3) & \text{for each strut} & \{i,j\}, & f_{ij} & \text{is strictly monotone decreasing,} \\ & \text{for each bar} & \{i,j\}, & f_{ij} & \text{has a strict minimum at } |\mathbf{p}_j - \mathbf{p}_i|^2. \end{array}$$

See Figure 6.

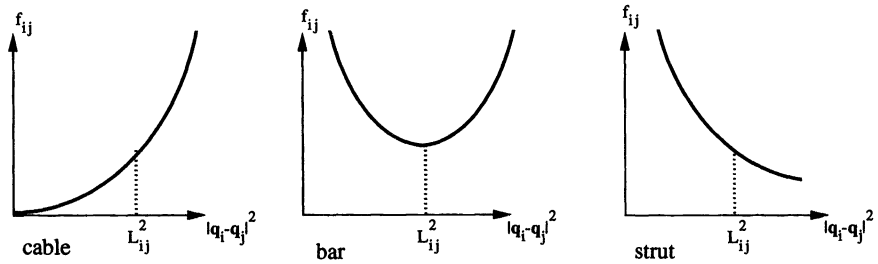


FIG. 6. *At the given configuration, the energy function is increasing on cables, a local minimum on bars, and decreasing on struts.*

We have the following Theorem 3.1.1.

THEOREM 3.1.1 (energy principle). *If such an H^* has a local minimum at $\mathbf{p}$ which is strict up to congruence in some neighborhood of $\mathbf{p}$ in $\mathbb{R}^{dv}$, then the framework $G(\mathbf{p})$ is rigid.*

Proof. Any nearby $\mathbf{q}$ with $G(\mathbf{q}) \leq G(\mathbf{p})$ will have $f_{ij}\left(|\mathbf{q}_j - \mathbf{q}_i|^2\right) \leq f_{ij}\left(|\mathbf{p}_j - \mathbf{p}_i|^2\right)$ for all members. Since this makes $H^*(\mathbf{q}) \leq H^*(\mathbf{p})$, we conclude that $\mathbf{q}$ is congruent to $\mathbf{p}$. This makes $G(\mathbf{p})$ rigid, by Definition 2.1.2(a) of rigidity. □

Remark 3.1.1. It turns out that any rigid tensegrity framework $G(\mathbf{p})$ will have "some" energy functions that make H^* a minimum at $\mathbf{p}$, but they would only satisfy certain "relaxed" conditions (2). Later it will be necessary to use conditions (2) as they stand.

3.2. Stiffness matrix, stress matrix decomposition. We apply the energy principle for functions $H^*(\mathbf{p})$ which have their minimum by the second derivative test. Let equation (2) define an energy function, where each f_{ij} is twice continuously differentiable and chosen so that the first derivative $f'_{ij}(|\mathbf{p}_j - \mathbf{p}_i|^2) = \omega_{ij}$ for each member, $\omega_{ij} \neq 0$ are scalars for all cables and struts, and the second derivative $f''(|\mathbf{p}_j - \mathbf{p}_i|^2) = c_{ij} > 0$ for all members. Note that (3) insures that ω is a strict and proper stress. We suppose that $\mathbf{p}$ is a fixed particular configuration.

To find a local minimum, the first step is to find a critical point. Note that $\mathbf{p}$ is a critical point for H^* if and only if the directional derivative at $\mathbf{p}$ is zero for all directions $\mathbf{p}^*$. Hence, we compute the directional derivative of this energy function in the direction $\mathbf{p}^*$ starting from $\mathbf{p}$.

Let $\mathbf{p}(t) = \mathbf{p} + t\mathbf{p}^*$. So $D_t(\mathbf{p}(t)) = \mathbf{p}^*$, where D_t represents differentiation with respect to t. We compute the derivative of $H^*(\mathbf{p}(t))$ with respect to t,

$$D_t\big(H^*(\mathbf{p}(t))\big) = \sum_{ij} f'_{ij}\big(|\mathbf{p}_j(t) - \mathbf{p}_i(t)|^2\big)\big[2(\mathbf{p}_j(t) - \mathbf{p}_i(t)) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*)\big].$$

The directional derivative is then the above function evaluated at $t = 0$.

$$D_t\big(H^*(\mathbf{p}(t))\big)\Big|_{t=0} = \sum_{ij} f'_{ij}\big(|\mathbf{p}_j - \mathbf{p}_i|^2\big)\big[2(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*)\big].$$

Since $f'_{ij}(|\mathbf{p}_j - \mathbf{p}_i|^2) = \omega_{ij}$, using (1) we have

$$D_t\big(H^*(\mathbf{p}(t))\big)\Big|_{t=0} = 2\sum_{ij} \omega_{ij}(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*) = 2\omega R(\mathbf{p})\mathbf{p}^*.$$

By the above calculation $\mathbf{p}$ is a critical point for H^* if and only if $2\omega R(\mathbf{p})\mathbf{p}^* = 0$ for all $\mathbf{p}^*$ if and only if $\omega R(\mathbf{p}) = 0$ if and only if ω is a self stress for $G(\mathbf{p})$.

If ω is a self stress on $G(\mathbf{p})$, we use the second derivative test to check whether H^* has a strict minimum at $\mathbf{p}$, up to congruences. For each direction $\mathbf{p}^*$ we calculate the second derivative along the path $\mathbf{p}(t)$ and then evaluate when $t = 0$ and $\mathbf{p}(0) = \mathbf{p}$.

$$\begin{aligned} D_t^2[H^*(\mathbf{p}(t))]\Big|_{t=0} &= \sum_{ij} f''_{ij}(|\mathbf{p}_j - \mathbf{p}_i|^2)[2(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*)]^2 \\ &\qquad + \sum_{ij} f'_{ij}(|\mathbf{p}_j - \mathbf{p}_i|^2)2|\mathbf{p}_j^* - \mathbf{p}_i^*|^2 \\ &= \sum_{ij} 4c_{ij}[(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*)]^2 \\ &\qquad + \sum_{ij} 2\omega_{ij}(\mathbf{p}_j^* - \mathbf{p}_i^*) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*). \end{aligned}$$

The constant c_{ij} is often called the *stiffness coefficient* for the member $\{i, j\}$ and in physics it is normally a function of the Young modulus of the material forming the member, the rest length of the member, and the cross-sectional area of the member.

The rigid congruences of $\mathbf{p}$ form a submanifold in the space of all configurations, and it is clear that H^* is constant on this set. Thus when $\mathbf{p}^*$ is a trivial infinitesimal flex of $G(\mathbf{p})$ it is easy to see that both the first and second derivative of H^* along a path in the direction of $\mathbf{p}^*$ are zero. (This can also be seen by a direct calculation.) We conclude with Proposition 3.2.1.

PROPOSITION 3.2.1. *The energy function H^* has a strict local minimum, up to congruence, if the quadratic form*

$$H(\mathbf{p}^*) = \sum_{ij} 2\omega_{ij}(\mathbf{p}_j^* - \mathbf{p}_i^*) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*) + \sum_{ij} 4c_{ij}[(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*)]^2,$$

regarded as a function of the coordinates of $\mathbf{p}^*$, satisfies $H(\mathbf{p}^*) \geq 0$ for all $\mathbf{p}^*$, and $H(\mathbf{p}^*) = 0$ if and only if $\mathbf{p}^*$ is a trivial infinitesimal flex of $G(\mathbf{p})$. In other words H is positive definite on any complement of the trivial infinitesimal flexes.

Note that each f_{ij} for each cable and strut is a monotone function with nonzero derivative. For that reason we know that the self stress is strict and proper. In other words the self stress ω is nonzero with the correct sign on each cable and strut.

What we have done is calculate the Hessian H as a quadratic form for the function H^*. Note, however, that H itself is not the sum of energy functionals of the members of G. Thus the energy principle does not apply directly to H but applies only because H^* is locally approximated by H.

We now rewrite the formula for H in terms of the rigidity matrix. Let C denote the dv-by-dv diagonal matrix with entries c_{ij}, where its rows and columns correspond to the members of G. If $\{i,j\}$ is a member of G, then the corresponding diagonal entry is c_{ij}.

$$\begin{aligned} H &= 2\omega R(\mathbf{p}^*)\mathbf{p}^* + 4(\mathbf{p}^*)^T R(\mathbf{p})^T C R(\mathbf{p})\mathbf{p}^* \\ &= 2(\mathbf{p}^*)^T \Omega \mathbf{p}^* + 4(\mathbf{p}^*)^T R(\mathbf{p})^T C R(\mathbf{p})\mathbf{p}^* \\ &= (\mathbf{p}^*)^T [2\Omega + 4R(\mathbf{p})^T C R(\mathbf{p})]\mathbf{p}^*. \end{aligned}$$

In structural engineering, the matrix $R(\mathbf{p})^T C R(\mathbf{p})$ is called the *stiffness matrix* of the framework, Ω is the *geometric stiffness matrix* or the *stress matrix*, and $2[\Omega + 2R(\mathbf{p})^T C R(\mathbf{p})]$ is the *tangential stiffness matrix*. The matrix $R(\mathbf{p})^T C R(\mathbf{p})$ is clearly positive semidefinite with the first-order flexes in its kernel. If the framework is infinitesimally rigid, then $\Omega = 0$ can be used in the above. The interesting cases for us occur when there are some nontrivial infinitesimal flexes and some nonzero self stresses.

Remark 3.2.1. The condition of Proposition 3.2.1 is a particular kind of stability which corresponds to the engineer's concept of first-order stiffness [30]. If we take gradients of this energy function, we find that if the force at the ith joint is $\mathbf{F}_i$, then this set of forces is resolved, at first-order, by a displacement $\mathbf{p}^*$ of the joints, where

$$\mathbf{F}_i = \Delta H = 2\sum_j \omega_{ij}(\mathbf{p}_j^* - \mathbf{p}_i^*) + 4\sum_j c_{ij}\big[(\mathbf{p}_j - \mathbf{p}_i) \cdot (\mathbf{p}_j^* - \mathbf{p}_i^*)\big](\mathbf{p}_j - \mathbf{p}_i).$$

Regarding the forces as one column vector $\mathbf{F} = (\mathbf{F}_1^T, \ldots, \mathbf{F}_v^T)^T$, we get

$$\mathbf{F} = \big[2\Omega + 4R(\mathbf{p})^T C R(\mathbf{p})\big]\mathbf{p}^*.$$

All equilibrium loads are resolved if and only if the matrix $[2\Omega + 4R(\mathbf{p})^T C R(\mathbf{p})]$ is invertible when restricted to the orthogonal complement of the trivial motions. In this

case, the deformation $\mathbf{p}^*$ resolves the load $[2\Omega + 4R(\mathbf{p})^T C R(\mathbf{p})]\mathbf{p}^*$. This is a feasible physical response of the structure, corresponding to positive work by the force, if and only if $\mathbf{p}^*$ is in the same direction as $\mathbf{F}$ or $\mathbf{p}^* \cdot \mathbf{F} \geq 0$, with equality only if $\mathbf{F} = 0$. This is a restatement of the fact that H is positive definite on a complement of the trivial motions.

If H is only positive semidefinite on a complement of the trivial infinitesimal motions, then there is a direction $\mathbf{p}^*$ for which there is no change in the energy up to the second derivative. It may turn out that there will still be third- or higher-order effects of a real energy which produce rigidity. However, if H is indefinite there is a direction $\mathbf{p}^*$ in which the energy strictly decreases.

Remark 3.2.2. Looking at this discussion in terms of the physics we can also understand the rule of the energy functions. If H strictly decreases in the direction $\mathbf{p}^*$, then any smooth energy function $\mathbf{H}^*$ with the equilibrium stresses ω_{ij} as the first derivatives and the c_{ij} as the second derivatives for each member will have a local maximum at $\mathbf{p}$ along the line $\mathbf{p} + t\mathbf{p}^*$. If released with this energy in the direction of $\mathbf{p}^*$, the framework will continue to move while seeking a smaller overall energy. It is also possible to interpret the behavior of the framework in terms of Lagrange multipliers.

3.3. Definition of prestress stability. Recall that if Q is a quadratic form on a finite-dimensional vector space, then there is a symmetric matrix A such that $Q(\mathbf{p}) = \mathbf{p}^T A \mathbf{p}$, where $\mathbf{p}$ is a vector written as coordinates with respect to some basis of the vector space. If $Q(\mathbf{p}) \geq 0$ for all vectors $\mathbf{p}$, then Q (or A) is called *positive semidefinite.* The *zero set* of Q is the set of vectors $\mathbf{p}$ such that $Q(\mathbf{p}) = 0$. If Q is positive semidefinite and the zero set consists of just the zero vector, then Q is called *positive definite.*

DEFINITION 3.3.1. *We say a tensegrity framework $G(\mathbf{p})$ is* prestress stable *if there is a proper self stress ω and nonnegative scalars c_{ij} (where $\{i,j\}$ is a member of G) such that the energy function regarded as quadratic form in the coordinates of $\mathbf{p}^*$*

$$H = \sum_{ij} \omega_{ij}(\mathbf{p}_i^* - \mathbf{p}_j^*)^2 + \sum_{ij} c_{ij}\left[(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i^* - \mathbf{p}_j^*)\right]^2$$

is positive semidefinite, $c_{ij} = 0$ when $\omega_{ij} = 0$ and $\{i,j\}$ is a cable or strut, and only the trivial infinitesimal flexes of $\mathbf{p}$ are in the kernel of H. In this case we say ω stabilizes *$G(\mathbf{p})$. The c_{ij} are called the stiffness coefficients as in §3.2. We have dropped the constants 2 and 4 that appeared in §3.2 for simplicity.*

Remark 3.3.1. If, for some member $\{i,j\}$, H has $\omega_{ij} = 0$ but $c_{ij} > 0$, then for the energy principle to apply, the member must be a bar. Imagine a single cable, which has no nonzero self stress. It is certainly not rigid, but it would be prestress stable with the zero self stress if we did not insist that $c_{ij} = 0$ when $\omega_{ij} = 0$. This is why the definition insists that each cable or strut which appears with a zero stress does not appear in the formula at all.

Thus if any $G(\mathbf{p})$ is prestress stable, stabilized at ω, then we might as well change $G(\mathbf{p})$ to have struts only for those members where $\omega_{ij} < 0$ and cables only for those members where $\omega_{ij} > 0$, deleting any cables or struts with a zero stress.

Note also that if we regard unstressed members as not being in G, then increasing any c_{ij} keeps H positive definite (if it already was) and so we may assume they are all equal to each other, assuming that we are only interested in recognizing the rigidity of the framework. Thus if ω stabilizes $G(\mathbf{p})$ with all cables and struts stressed, with

stiffness coefficients c_{ij}, then $\frac{\omega}{\max\{c_{ij}\}}$ stabilizes $G(\mathbf{p})$ with all the stiffness coefficients equal to one.

PROPOSITION 3.3.2. *If a tensegrity framework $G(\mathbf{p})$ is prestress stable for a self stress ω, then $G(\mathbf{p})$ is rigid.*

Proof. The positive definite property of H guarantees a strict local minimum modulo trivial first-order flexes of $\mathbf{p}$. Since $\omega_{ij} \neq 0$ for all the cables and struts of G that appear in H^*, we have the strictly monotone property of their energy functions, near $\mathbf{p}$. Thus the energy principle applies to show $G(\mathbf{p})$ is rigid. □

Remark 3.3.2. We will see, by Theorem 4.4.1, that if $G(\mathbf{p})$ is prestress stable, then $G(\mathbf{p})$ is second-order rigid and, by Theorem 4.3.1, $G(\mathbf{p})$ is rigid. However, the present proof is much simpler since it is a direct application of the energy principle. □

PROPOSITION 3.3.3. *If a tensegrity framework $G(\mathbf{p})$ is first-order rigid, then it is prestress stable.*

Proof. The underlying bar framework $\overline{G}(\mathbf{p})$ is certainly first-order rigid; thus $R(\mathbf{p})^T R(\mathbf{p})$ has only the trivial infinitesimal flexes in its kernel. In other words $R(\mathbf{p})^T R(\mathbf{p})$ is positive definite on the equilibrium infinitesimal motions perpendicular to the trivial infinitesimal motions in $\mathbb{R}^{vd}$. By [29] there is a proper self stress ω which is nonzero on each cable and strut. By choosing ω sufficiently small, then $\Omega + R(\mathbf{p})^T R(\mathbf{p})$ also will be positive definite on the equilibrium infinitesimal flexes restricted to the compact unit sphere. Thus $G(\mathbf{p})$ is prestress stable. □

Often it is convenient to assume that a proper stress ω for $G(\mathbf{p})$ is *strict*; i.e., $\omega_{ij} \neq 0$ for every cable or strut. If we are willing to consider subframeworks of $G(\mathbf{p})$, we need only consider strict self stresses for prestress stability.

Remark 3.3.3. Pellegrino and Calladine [28] use a different analysis of the rigidifying effect of a prestress. (See also [4].) Given a framework $G(\mathbf{p})$ with a self stress ω and a set of generators $\mathbf{p}'_1, \ldots, \mathbf{p}'_k$ for a complementary space of nontrivial first-order flexes, they add k new rows to the rigidity matrix $\omega R(\mathbf{p}'_1), \omega R(\mathbf{p}'_2), \ldots, \omega R(\mathbf{p}'_k)$. If this extended matrix $R^*(\mathbf{p}, \omega)$ has rank $vd - \frac{d(d+1)}{2}$, they say that the prestress ω "stiffens" the framework, modulo the positive definiteness of an unspecified matrix.

If $R^*(\mathbf{p}, \omega)$ does not have rank $vd - d(d+1)/2$ (assuming the vertices span $\mathbb{R}^d$), then there is a nontrivial first-order flex $\mathbf{p}' = \sum \alpha_i \mathbf{p}_j$ satisfying $\omega R(\mathbf{p}'_i)\mathbf{p}' = 0$ for all $i = 1, \ldots, k$. Thus $\omega R(\mathbf{p}')\mathbf{p}' = 0$ and $G(\mathbf{p})$ is certainly not prestress stable for this self stress ω. In fact, it is easy to see that their condition for stiffening is equivalent to requiring that the rank of $\alpha^2\Omega + R(\mathbf{p})^T R(\mathbf{p})$ be $vd - \frac{d(d+1)}{2}$ for all sufficiently small α (assuming that the affine span of $\mathbf{p}$ is at least $(d-1)$-dimensional). The matrix that they have in mind, which must be positive definite, must be equivalent to $\Omega + R(\mathbf{p})^T R(\mathbf{p})$ restricted to some space complementary to the space of trivial first-order flexes. If no positive definiteness is required, many mechanisms, such as collinear parallelograms, would be declared "stiff."

On the other hand, if there is a one-dimensional space of equilibrium first-order flexes, then we will see that prestress stability, the rank of $R^*(\mathbf{p}) = dv - \frac{d(d+1)}{2}$, and second-order rigidity will all coincide. It is interesting that in the paper [28], most examples have a one-dimensional space of equilibrium flexes. See [5] for corrections, as well as [22–26] for a discussion of the problem of how to do the second-order analysis.

3.4. Interpretation in terms of the stress matrix and quadratic forms. We now present some simple facts about quadratic forms that we will find useful later.

LEMMA 3.4.1. *Let Q_1 and Q_2 be two quadratic forms on a finite-dimensional real (inner product) vector space. Suppose that Q_2 is positive semidefinite with zero*

set K, and Q_1 is positive definite on K. Then there is a positive real number α such that $Q_1 + \alpha Q_2$ is positive definite.

Proof. Let X denote the compact set (a sphere) of unit vectors in the inner product space.

$$X = \{\mathbf{p} \mid \mathbf{p} \cdot \mathbf{p} = 1\}.$$

Recall that the zero set of Q_2 is

$$K = \{\mathbf{p} \mid Q_2(\mathbf{p}) = 0\}.$$

Let $K \cap X \subset N \subset X$ be an open neighborhood of K in X such that

$$Q_1(\mathbf{p}) > 0 \qquad \text{for all} \qquad \mathbf{p} \in N.$$

Such an open set N exists since $K \cap X$ is compact, Q_1 is positive on $K \setminus \{0\} \supset K \cap X$, and thus Q_1 restricted to $K \cup X$ must have a positive minimum m. Then take $N = \{\mathbf{p} \in X | Q_1(\mathbf{p}) > \frac{m}{2}\}$. For similar reasons there are real constants c_1, c_2 such that

$$\begin{aligned} Q_1(\mathbf{p}) &> c_1 \qquad \text{for all} \qquad \mathbf{p} \in X, \\ Q_2(\mathbf{p}) &\geq c_2 > 0 \qquad \text{for all} \qquad \mathbf{p} \in X \setminus N. \end{aligned}$$

Then we define $\alpha = \frac{|c_1|}{c_2}$. We calculate for $\mathbf{p} \in N \cap X$,

$$Q_1(\mathbf{p}) + \alpha Q_2(\mathbf{p}) \geq Q_1(\mathbf{p}) > 0.$$

For $\mathbf{p} \in X \setminus N$,

$$Q_1(\mathbf{p}) + \alpha Q_2(\mathbf{p}) = Q_1(\mathbf{p}) + \frac{|c_1|}{c_2} Q_2(\mathbf{p}) > c_1 + |c_1| \geq 0.$$

Thus $Q_1 + \alpha Q_2$ is positive on all of X, and hence it is positive definite. □

Remark 3.4.1. Note that Q_1 is allowed to be an indeterminate term on the whole vector space. It is only required to be positive definite on K. Note also, for any quadratic form given by a symmetric matrix A, where $Q(\mathbf{p}) = \mathbf{p}^T A \mathbf{p}$, certainly the kernel of A is contained in the zero set of Q. If $A\mathbf{p} = \mathbf{0}$, then $Q(\mathbf{p}) = \mathbf{p}^T A \mathbf{p} = 0$. However, the converse is not true unless A is positive semidefinite. In particular the converse is true when $A = R(\mathbf{p})^T C R(\mathbf{p})$, the stiffness matrix. Then

$$(\mathbf{p}^*)^T R(\mathbf{p})^T C R(\mathbf{p}) \mathbf{p}^* = \sum_{ij} c_{ij} \left[(\mathbf{p}_j^* - \mathbf{p}_i^*) \cdot (\mathbf{p}_j - \mathbf{p}_i) \right]^2 \geq 0,$$

and the kernel of A, assuming all the $c_{ij} > 0$, is the same as the zero set of its quadratic form. It is also clear from the above that the kernel of A is precisely the space of all first-order flexes of the corresponding bar framework.

For any tensegrity framework $G(\mathbf{p})$ we recall that $\overline{G}(\mathbf{p})$ is the corresponding bar framework with all members converted to bars. We denote

$$\overline{I} = I(\overline{G}(\mathbf{p})) = \left\{ \mathbf{p}' \in \mathbb{R}^{vd} \mid (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i' - \mathbf{p}_j') = 0, \quad \{i,j\} \text{ a member of } G \right\}.$$

In other words $\overline{I}$ is the space of first-order flexes of $\overline{G}(\mathbf{p})$, a linear subspace of $\mathbb{R}^{vd}$. Recall also that $T_{\mathbf{p}}$ is the space of trivial first-order flexes at $\mathbf{p}$. (So $T_{\mathbf{p}} \subset \overline{I}$.) Note

that if $G(\mathbf{p})$ has a strict proper self stress, then the first-order stress test implies that $I = \overline{I}$.

We now give a way of checking the prestress stability of a tensegrity framework which is useful for calculations later. Recall a stress ω is strict if it is nonzero on every cable and strut.

PROPOSITION 3.4.2. *A tensegrity framework $G(\mathbf{p})$ is prestress stable for the strict proper self stress ω if and only if the associated stress matrix Ω is positive definite on any subspace $K \subset \overline{I}$ complementary to $T_{\mathbf{p}}$.*

Proof. Assume that $[\Omega + R(\mathbf{p})^T C R(\mathbf{p})]$ is positive semidefinite with only $T_{\mathbf{p}}$ as the kernel, where C is a diagonal matrix with positive stiffness coefficients. Let $\mathbf{p}' \in \overline{I}$. Then $R(\mathbf{p})\mathbf{p}' = 0$ and so

$$0 \leq (\mathbf{p}')^T \big[\Omega + R(\mathbf{p})^T C R(\mathbf{p})\big]\mathbf{p}' = (\mathbf{p}')^T \Omega \mathbf{p}',$$

with equality if and only if $\mathbf{p}' \in T_{\mathbf{p}}$. Thus on K, Ω is positive definite.

Before proving the converse we remark that if $\mathbf{p}'$ is any trivial first-order flex at $\mathbf{p}$, then by Definition 2.2.2 there is a d-by-d (skew symmetric) matrix S and a vector $\mathbf{t} \in \mathbb{R}^d$ such that $\mathbf{p}'_i = S\mathbf{p}_i + \mathbf{t}$, $i = 1, \ldots, v$. Thus

$$\Omega\mathbf{p}' = \begin{bmatrix} \vdots \\ \sum_j \omega_{ij}(\mathbf{p}'_i - \mathbf{p}'_j) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \sum_j \omega_{ij}(S\mathbf{p}'_i - S\mathbf{p}'_j) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ S\sum_j \omega_{ij}(\mathbf{p}'_i - \mathbf{p}'_j) \\ \vdots \end{bmatrix} = 0.$$

Now suppose Ω is positive definite on K. Let $\mathbf{p}' \in \mathbb{R}^{vd}$ be arbitrary. Write $\mathbf{p}' = \mathbf{p}_T + \mathbf{p}_K + \mathbf{p}_E$, where $\mathbf{p}_T \in T_{\mathbf{p}}$, $\mathbf{p}_K \in K$, and $\mathbf{p}_e \in E$, the (orthogonal) complement of $\overline{I}$ in $\mathbb{R}^{vd}$. Since $\Omega\mathbf{p}_T = 0 = R(\mathbf{p})\mathbf{p}_T$, we have

$$(4) \quad (\mathbf{p}')^T \big[\Omega + R(\mathbf{p})^T C R(\mathbf{p})\big]\mathbf{p}' = (\mathbf{p}_K + \mathbf{p}_E)^T \big[\Omega + R(\mathbf{p})^T C R(\mathbf{p})\big](\mathbf{p}_K + \mathbf{p}_E).$$

By Remark 3.4.1, the kernel of $R(\mathbf{p})^T C R(\mathbf{p})$ is $\overline{I} = T_{\mathbf{p}} + K$. Now apply Lemma 3.4.1 to the inner product space $K + E$, where Q_1 is the quadratic form corresponding to Ω, and Q_2 is the quadratic form corresponding to $R(\mathbf{p})^T C R(\mathbf{p})$. The kernel of Q_2 is precisely K, and we have assumed that Q_1 is positive definite on K. Thus by possibly multiplying C by a positive constant we can assume that $\Omega + R(\mathbf{p})^T C R(\mathbf{p})$ is positive definite on $K + E$. By (4), this implies that $\Omega + R(\mathbf{p})^T C R(\mathbf{p})$ is positive semidefinite with kernel $T_{\mathbf{p}}$, and $G(\mathbf{p})$ is prestress stable. □

3.5. Examples of prestress stable frameworks. The following are examples of tensegrity frameworks that are prestress stable, but not first-order rigid. Thus the converse of Proposition 3.3.3 is false.

In Figure 7a there is a self stress such that the outside members have a positive self stress. A nontrivial first-order flex is given so that one can apply Proposition 3.4.2. Figure (7b) is stable by a result in [7] concerning spider webs. In Figure 7b it is the inside members we can choose to be positive. In both of these examples there is a strict proper self stress such that the indicated first-order flex is nonzero only on vertices of members that have a positive self stress, and the given first-order flex generates a complementary space to the trivial flexes in the space of all first-order

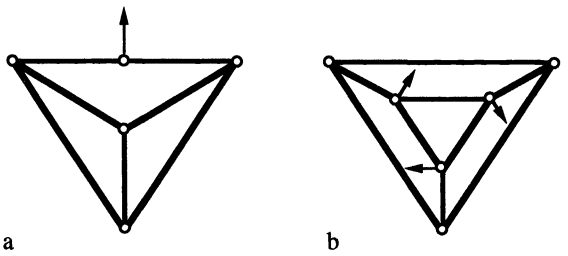


FIG. 7. *Examples of prestress stable bar frameworks that are not first-order rigid. (Nontrivial first-order motions are indicated).*

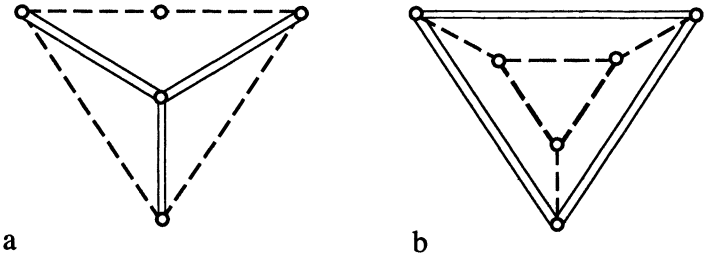


FIG. 8. *Corresponding examples of prestress stable tensegrity frameworks, where the bars have been replaced by cables and struts following the stabilizing self stress.*

flexes. Hence the stress matrix on this space is positive definite and the framework is prestress stable.

Following Remark 3.3.1, we can change appropriate members to be cables or struts and preserve prestress stability, as in Figure 8.

Can a framework be rigid but not prestress stable? Consider the next two examples. Note that any first-order rigid bar framework with no self stress at all must have **0** as its stabilizing self stress. But the example in Figure 9a has a self stress on part of the framework, and the bar can have no stress other than 0. It still is prestress stable.

However, the example of Figure 9b is not prestress stable, because the short horizontal cable and the horizontal strut are unstressed and the framework becomes nonrigid upon their removal. Nevertheless the framework is clearly still rigid. In fact, built with all bars, it is prestress stable.

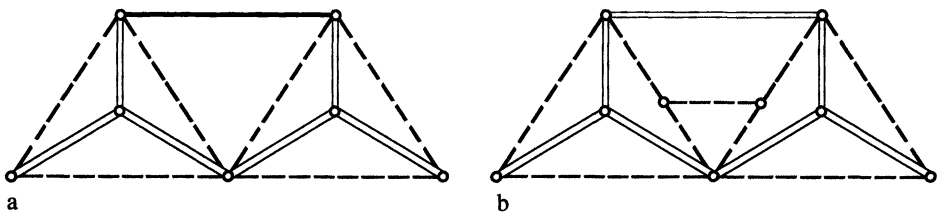


FIG. 9. *Example* (a) *is first-order rigid, and thus prestress stable. Example* (b) *is rigid but not prestress stable.*

Suppose we fix (or pin) certain vertices. For our purposes this can be accomplished by adding some first-order framework that contains these vertices and none of the other original vertices. If the original framework has only cables with a proper self stress (where the equilibrium condition only holds at the vertices that are not fixed), then we say that the framework is a *spider web*. There is a discussion of this in [9] and [36] as well as [7]. It is clear that spider webs are prestress stable. See Figure 10.

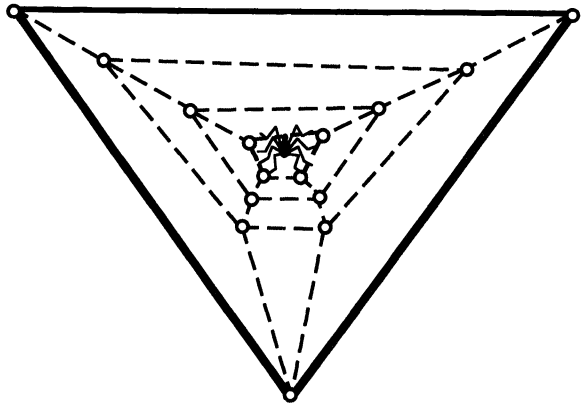


FIG. 10. *This example is a spider web and thus prestress stable.*

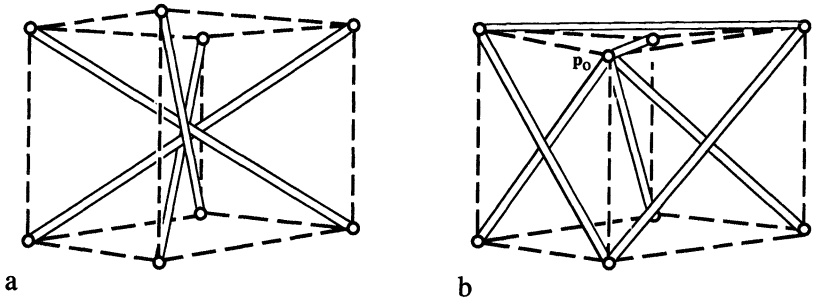


FIG. 11. *Two prestress stable tensegrity frameworks in three-space.*

In three-space there are many examples of prestress stable but not necessarily infinitesimally rigid tensegrity frameworks, such as in Figure 11.

Figure 11a is a regular cube with its main diagonals as struts and its edges as cables. Examples such as in 11b can be obtained by taking any convex polyhedron with a triangular face (in this case a cube with its near upper corner truncated), choosing a point $\mathbf{p}_0$ close to that face, and joining $\mathbf{p}_0$ to all the other vertices of the polyhedron with struts and making all the edges of the polyhedron cables except the triangle which is composed of struts. Again it turns out that there is a strict proper self stress ω, and Ω is positive semidefinite with only the affine motions in the kernel. This example is closely related to three-dimensional spider webs. (See [36].)

Another three-dimensional example can be obtained by taking a tetrahedron and putting struts on each of the six edges and some prestressed spider web on the inside of each triangular face, as in Figure 12.

Each face is prestress stable even in $\mathbb{R}^3$, so the sum of the energy functions is positive semidefinite with only the first-order flexes that are trivial on each face in the kernel. But since the tetrahedron itself is first-order rigid, flexes which are trivial on each face are trivial on the whole framework. The framework is prestress stable.

3.6. Roth's conjecture. Suppose $G(\mathbf{p})$ has its points as the vertices of a convex polygon in the plane. If the exterior edges of G are cables, and all of the other members are struts, say, then we call it a *cable-strut polygon* or a *c-s polygon.*

It follows from [7] that any *c-s* polygon that has a proper self stress $\omega \neq 0$ has Ω as positive semidefinite with only the affine motions in the kernel. It turns out that the affine motions are never a first-order flex of such a framework [36]. Thus such an

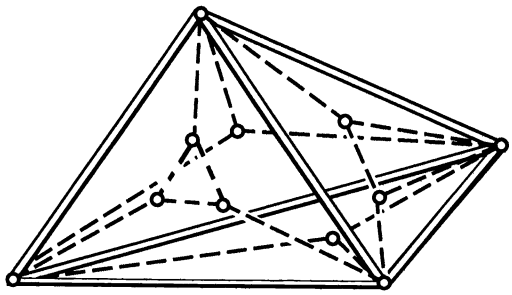


FIG. 12. *A prestress stable tetrahedron with cables in the faces and struts on the edges.*

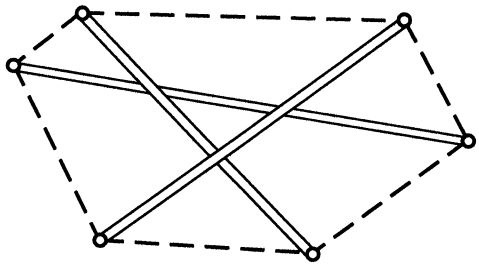


FIG. 13. *A cable-strut (c-s) polygon.*

ω also stabilizes $G(\mathbf{p})$, and thus $G(\mathbf{p})$ is prestress rigid. Note that such a *c-s* polygon need not be first-order rigid. In the case of Figure 13, the six vertices lie on an ellipse, and by a classical result (see [3] and [34]), the framework has a strict proper self stress and a nontrivial first-order flex. So this framework is prestress stable, but it is not first-order rigid.

On the other hand Roth conjectured that any rigid *b-c* polygon (bars on the outside, cables inside) was first-order rigid. In §6.2 we show that this is true. In Figure 14 we show three examples of nonrigid *b-c* hexagons with first-order flex indicated.

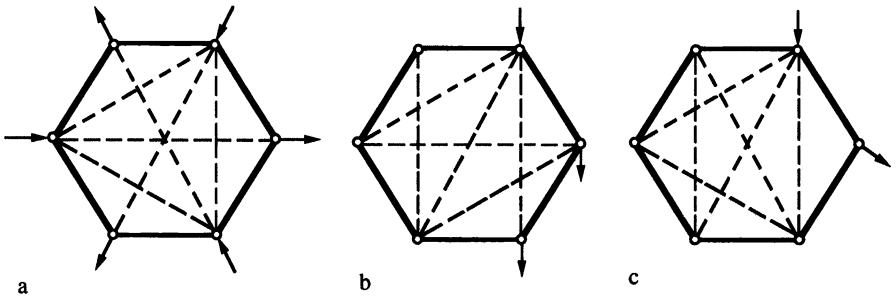


FIG. 14. *Examples of bar-cable (b-c) polygons, with nontrivial first-order motions indicated.*

For the three frameworks in Figure 14, the six vertices of each configuration form a regular hexagon. For the first two cases, there are certain other configurations for the same tensegrity graph (but still a convex *b-c* polygon) such that the framework is rigid. This is not true for the last case, though. The reader is invited to find the continuous nontrivial flex of each of these frameworks. But see §6.2 for a proof that the flex exists.

Following [29] we see that there are many cabling schemes that guarantee first-order rigidity, as in Figure 15.

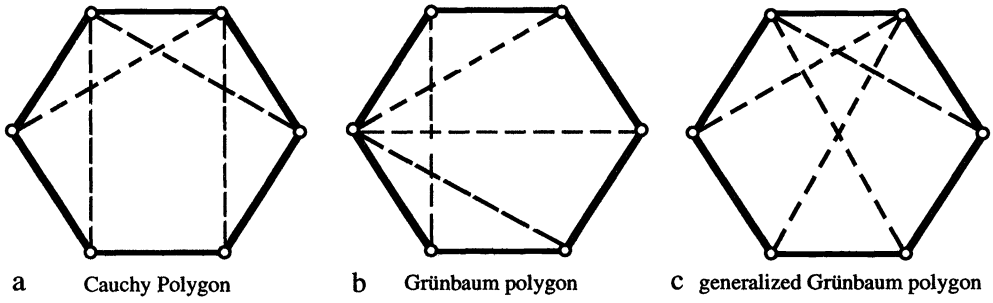


FIG. 15. *Examples of first-order rigid bar-cable polygons.*

Consider a *b-c* hexagon with four cables. One can see that either it contains one of the examples of Figure 15 and thus is always first-order rigid, or it is contained in one of the examples of Figure 14 and thus, at least for some convex configurations, it is not rigid.

4. Second-order rigidity for tensegrity frameworks.

4.1. The definition of second-order rigidity. Our definition of second-order rigidity for tensegrity frameworks comes from differentiating the equation $|\mathbf{p}_j(t) - \mathbf{p}_i(t)|^2 = L_{ij}^2$ twice. This generalizes the previous definition of second-order rigidity for bar frameworks in [7].

DEFINITION 4.1.1. *A* second-order flex $(\mathbf{p}', \mathbf{p}'')$ *for a tensegrity framework* $G(\mathbf{p})$ *is a solution to the following constraints, where* $\mathbf{p}'$ *and* $\mathbf{p}''$ *are configurations in* $\mathbb{R}^d$ *(each regarded as an associated pair of vectors* $\mathbf{p}_i'$ *and* $\mathbf{p}_i''$ *to each point* $\mathbf{p}_i$*).*

(a) *For* $\{i,j\}$ *a bar,* $(\mathbf{p}_i - \mathbf{p}_j)\cdot(\mathbf{p}_i' - \mathbf{p}_j') = 0$ *and* $|\mathbf{p}_i' - \mathbf{p}_j'|^2 + (\mathbf{p}_i - \mathbf{p}_j)\cdot(\mathbf{p}_i'' - \mathbf{p}_j'') = 0$.

(b) *For* $\{i,j\}$ *a cable, either* $(\mathbf{p}_i - \mathbf{p}_j)\cdot(\mathbf{p}_i' - \mathbf{p}_j') < 0$ *or* $(\mathbf{p}_i - \mathbf{p}_j)\cdot(\mathbf{p}_i' - \mathbf{p}_j') = 0$ *and* $|\mathbf{p}_i' - \mathbf{p}_j'|^2 + (\mathbf{p}_i - \mathbf{p}_j)\cdot(\mathbf{p}_i'' - \mathbf{p}_j'') \le 0$.

(c) *For* $\{i,j\}$ *a strut, either* $(\mathbf{p}_i - \mathbf{p}_j)\cdot(\mathbf{p}_i' - \mathbf{p}_j') > 0$ *or* $(\mathbf{p}_i - \mathbf{p}_j)\cdot(\mathbf{p}_i' - \mathbf{p}_j') = 0$ *and* $|\mathbf{p}_i' - \mathbf{p}_j'|^2 + (\mathbf{p}_i - \mathbf{p}_j)\cdot(\mathbf{p}_i'' - \mathbf{p}_j'') \ge 0$.

A tensegrity framework is second-order rigid *if all second-order flexes* $(\mathbf{p}', \mathbf{p}'')$ *have* $\mathbf{p}'$ *as a trivial first-order flex. Otherwise* $G(\mathbf{p})$ *is* second-order flexible.

Figure 16 shows second-order flexes of some tensegrity frameworks (double arrows for $\mathbf{p}''$, single arrow for $\mathbf{p}'$). The flex in Figure 16a is nontrivial for $\mathbf{p}'$. The flex in Figure 16b is trivial for $\mathbf{p}'$, but $(\mathbf{p}', \mathbf{p}'')$ is not the first and second derivative of a rigid motion of $\mathbf{p}$. The flex in Figure 16c is the derivative of a rigid motion of $\mathbf{p}$. Figure 16d shows a nontrivial second-order flex in a framework which is still rigid.

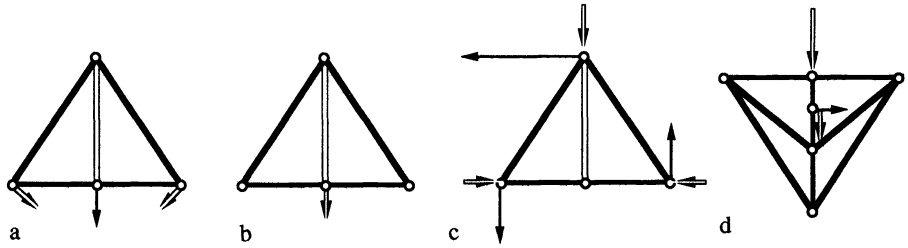


FIG. 16. *Examples of bar frameworks with first-order (single arrows) and second-order (double arrows) motions indicated.*

Since the second-order extension $\mathbf{p}$ is the solution of an inhomogeneous system of

equations and inequalities, we can add any solution $\mathbf{q}'$ of the corresponding homogeneous system to $\mathbf{p}''$.

PROPOSITION 4.1.2. *If $(\mathbf{p}', \mathbf{p}'')$ is a second-order flex of a tensegrity framework $G(\mathbf{p})$ and $\mathbf{q}'$ is any first-order flex of $G(\mathbf{p})$, then $(\mathbf{p}', \mathbf{p}'' + \mathbf{q}')$ is a second-order flex of $G(\mathbf{p})$.*

Proof. Assume that for each cable $\{i, j\}$ (respectively, each bar, strut) with $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i' - \mathbf{p}_j') = 0$,

$$(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i' - \mathbf{p}_j') + (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i'' - \mathbf{p}_j'') \leq 0 \quad \text{(respectively, } = 0, \geq 0)$$

and

$$(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{q}_i' - \mathbf{q}_j') \leq 0 \quad \text{(respectively, } = 0, \geq 0).$$

Therefore by adding these inequalities we obtain

$$(\mathbf{p}_i' - \mathbf{p}_j') \cdot (\mathbf{p}_i' - \mathbf{p}_j') + (\mathbf{p}_i - \mathbf{p}_j) \cdot [(\mathbf{p}_i'' + \mathbf{q}_i') - (\mathbf{p}_j'' + \mathbf{q}_j')] \leq 0 \quad \text{(respectively, } = 0, \geq 0).$$

These are the inequalities required for Definition 4.1.1. □

If we add a multiple of $\mathbf{p}'$ itself to any second-order extension $\mathbf{p}$ we can make the second-order extension also satisfy the second-order inequalities even for those members with $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i' - \mathbf{p}_j') \neq 0$.

4.2. Trivial higher-order motions. In the following $\mathbf{p}(t) = (\mathbf{p}_1(t), \ldots)$, $0 \leq t \leq 1$ will be an analytic path in the configuration space, so $\mathbf{p}_i(t) \in \mathbb{R}^d$, $i = 1, \ldots, v$. Following [6] we say that $\mathbf{p}(t)$ is a trivial flex if

$$\mathbf{p}(t) = T(t)\mathbf{p}(0) = (T(t)\mathbf{p}_1(0), T(t)\mathbf{p}_2(0), \ldots, T(t)\mathbf{p}_v(0)),$$

where $T(0) = I$, $T(t)$ is a rigid motion of $\mathbb{R}^d$, and $T(t)$ is an analytic function of t. In particular, this means that we can write $T(t)\mathbf{p}_i = A(t)\mathbf{p}_i + \mathbf{b}(t)$, $i = 1, \ldots, v$, where $A(t)$ is an orthogonal matrix, $\mathbf{b}(t) \in \mathbb{R}^d$, and all the coordinates are real analytic functions of t.

We next say that $\mathbf{p}', \mathbf{p}'', \ldots, \mathbf{p}^{(k)}$, each $\mathbf{p}^{(j)} \in \mathbb{R}^{dv}$ for $j = 1, \ldots, k$, is a *k-trivial* flex of $\mathbf{p}$ if there is a trivial flex $\mathbf{p}(t)$ such that

$$D_t^j \mathbf{p}(t)\Big|_{t=0} = \mathbf{p}^{(j)} \qquad \text{for } j = 1, 2, \ldots, k.$$

Recall that D_t^j represents the jth derivative with respect to t.

It is easy to check that if $\mathbf{p}', \ldots, \mathbf{p}^{(k)}$ is a k-trivial flex of any framework $G(\mathbf{p})$ in $\mathbb{R}^d$, then the analogue of equations (a) in Definition 4.1.1 holds for $j = 1, \ldots, k$, since clearly edge lengths are preserved up to any order k.

In fact we will give a fairly explicit description of k-trivial flexes. Although this description is long, it seems important to be precise, given the long history of confusion in this area.

We already know that 1-trivial flexes $\mathbf{p}'$ are given by

$$\mathbf{p}_i' = S\mathbf{p}_i + \mathbf{b}', \quad i = 1, \ldots, v,$$

where $S = -S^T$ is a d-by-d skew symmetric matrix and $\mathbf{b}' \in \mathbb{R}^d$. See [6] or [9]. In fact every orthogonal matrix A sufficiently close to the identity matrix I can be written as

$$A = e^S = 1 + S = \frac{1}{2}S^2 + \frac{1}{2 \cdot 3}S^3 + \ldots,$$

where S is a skew symmetric matrix, and the above infinite series converges. It is well known that the exponential map

$$S \mapsto e^S$$

takes the tangent space of the Lie group to orthogonal matrices, which is the Lie algebra of the Lie group, into the Lie group itself. This exponential map is a local analytic diffeomorphism near I, the identity. Thus any analytic path $A(t)$ with $A(0) = I$ pulls back to a path $S(t)$ in the Lie algebra, which is itself analytic. Thus

$$A(t) = e^{S(t)}.$$

On the other hand since $e^0 = I$ and thus $S(0) = 0$ we can write

$$S(t) = tS_1 + \frac{t^2}{2}S_2 + \frac{t^3}{2 \cdot 3}S_3 + \cdots,$$

where each $S_1, S_2, \ldots$ is skew symmetric. Thus

$$A(t) = I + \left(tS_1 + \frac{t^2}{2}S_2 + \frac{t^3}{2 \cdot 3}S_3 + \ldots\right) + \frac{1}{2}\left(tS_1 + \frac{t^2}{2}S_2 + \frac{t^3}{2 \cdot 3}S_3 + \ldots\right)^2 + \ldots.$$

Rearranging terms, which is possible since we have an absolutely convergent power series, we get

$$A(t) = I + tS_1 + \frac{t^2}{2}(S_2 + S_1^2) + \frac{t^3}{2 \cdot 3}\left(S_3 + \frac{3}{2}S_1S_2 + \frac{3}{2}S_2S_1 + S_1^3\right) + \ldots. \tag{5}$$

Thus each of the matrix coefficients of $\frac{t^j}{j!}$ gives a parametric description of the jth derivative of $A(t)$, and thus a description of a k-trivial flex of $\mathbf{p}$. In particular $\mathbf{p}'$, $\mathbf{p}''$ is a 2-trivial flex of $\mathbf{p}$ if and only if there are skew symmetric matrices S_1, S_2 and $\mathbf{b}', \mathbf{b}'' \in \mathbb{R}^d$ such that

$$\mathbf{p}' = S_1\mathbf{p} + (\mathbf{b}', \ldots, \mathbf{b}'),$$

$$\mathbf{p}'' = (S_2 + S_1^2)\mathbf{p} + (\mathbf{b}'', \ldots, \mathbf{b}'').$$

Later it will be convenient to be able to "cancel" initial parts of a kth order flex, with a k-trivial flex. Thus we state the following. See [6].

PROPOSITION 4.2.1. *Let $\mathbf{p}(t)$, $\mathbf{p}(0) = \mathbf{p}$, be an analytic path in configuration space such that*

$$\mathbf{p}^{(j)} = D_t^j[\mathbf{p}(T)]\Big|_{t=0}, \quad j = 0, 1, \ldots, k$$

is k-trivial. Then there is a rigid motion $T(t)$ of $\mathbb{R}^d$, analytic in t, such that $T(0) = I$ and

$$D_t^j[T(t)\mathbf{p}(t)]\Big|_{t=0} = 0, \qquad \text{for } j = 1, \ldots, k.$$

Proof. We proceed by induction on k. For $k = 0$, there is nothing to prove. So we assume

$$D_t^j[\mathbf{p}(t)]\Big|_{t=0} = \begin{cases} \mathbf{p} & \text{if} \quad j = 0, \\ 0 & \text{if} \quad j = 1, \ldots, k, \end{cases}$$

and we wish to find $T(t)$ such that the $(k+1)$st derivative is 0 as well, assuming that the first $k+1$ derivatives are k-trivial.

We restrict to the space spanned by $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_v$. By adding a translation we have, using (5),

$$\mathbf{b}^{(j)} = 0, \quad j = 1, \ldots, k+1,$$
$$S_j = 0, \quad j = 1, \ldots, k.$$

Note then that $D_t^{k+1}\mathbf{p}(t)|_{t=0} = S_{k+1}\mathbf{p}$. Define $T(t)$ by

$$T(t) = e^{\frac{-t^{k+1}}{(k+1)!}S_{k+1}} = I - \frac{t^{k+1}}{(k+1)!}S_{k+1} + \cdots .$$

We then observe, for all $j = 1, 2, \ldots,$

$$D_t^j[T(t)\mathbf{p}(t)] = \sum_{\ell=0}^{j} \binom{j}{\ell} D_t^\ell[T(t)] \cdot D_t^{j-\ell}[\mathbf{p}(t)]. \tag{6}$$

But

$$D_t^\ell T(t)\Big|_{t=0} = \begin{cases} 0 & \ell = 1, \ldots, k, \\ -S_{k+1} & \ell = k+1. \end{cases}$$

Thus

$$D_t^j[T(t)\mathbf{p}(t)]\Big|_{t=0} = \left\{ \begin{array}{ll} S_{k+1}\mathbf{p} - S_{k+1}\mathbf{p} = 0, & \text{for } j = k+1 \\ 0 & \text{for } j = 1, \ldots, k \end{array} \right\} = 0. \qquad \square$$

Remark 4.2.1. There are several ways of handling the problem of "normalizing" the first few derivatives. One way is the above technique; another way is to use "tie downs" as in [33] and discussed in [9]; a third way is to use the method described by [20]. (See also [9].) This normalizing is a nuisance but it is convenient to have for the argument used to show that second-order rigidity implies rigidity.

The following is an immediate consequence of the definition of k-trivial and the formula (5).

LEMMA 4.2.2. *Let* $\mathbf{p}', \ldots, \mathbf{p}^{(k)}$ *be such that* $\mathbf{p}' = \mathbf{p}'' \cdots = \mathbf{p}^{(k-1)} = 0$. *Then* $\mathbf{p}^{(k)}$ *is a* 1*-trivial flex at* $\mathbf{p}$ *if and only if* $\mathbf{p}', \mathbf{p}'', \ldots, \mathbf{p}^{(k)}$ *is* k*-trivial at* $\mathbf{p}$.

It is also useful to have the following.

LEMMA 4.2.3. *Let* $\mathbf{p}(t)$ *be any analytic path in configuration space such that* $\mathbf{p}' = D_t\mathbf{p}(t)\big|_{t=0}, \ldots, \mathbf{p}^{(k)} = D_t^k\mathbf{p}(t)\big|_{t=0}$. *Let* $T(t)$ *be any rigid motion of* $\mathbb{R}^d$, *analytic in* t; $T(0) = I$. *Then* $D_t[T(t)\mathbf{p}(t)]\big|_{t=0}, \ldots, D_t^k[T(t)\mathbf{p}(t)]\big|_{t=0}$ *is* k*-trivial at* $\mathbf{p}$ *if and only if* $\mathbf{p}', \ldots, \mathbf{p}^{(k)}$ *is* k*-trivial at* $\mathbf{p}$.

Proof. Since $T(t)$ is invertible it is enough to show that if $\mathbf{p}', \ldots, \mathbf{p}^{(k)}$ is k-trivial, then $D_t[T(t)\mathbf{p}(t)]\big|_{t=0}, \ldots, D_t^k[T(t)\mathbf{p}(t)]\big|_{t=0}$ is k-trivial. But then $\mathbf{p}^{(\ell)} = D_t^\ell[\overline{T}(t)]\big|_{t=0}$ for $\ell = 1, \ldots, k$ for some rigid motion $\overline{T}(t)$ of $\mathbb{R}^d$, analytic in t, $\overline{T}(0) = I$. But then clearly for $\ell = 1, \ldots, k$,

$$D_t^\ell[T(t)\mathbf{p}(t)]\Big|_{t=0} = D_t^\ell[T(t)\overline{T}(t)\mathbf{p}(t)]\Big|_{t=0},$$

by expanding both sides by the product rule (6). But $T(t)\overline{T}(t)$ is again a rigid analytic motion of $\mathbb{R}^d$ and we are done. $\square$

4.3. Second-order rigidity implies rigidity. We have generalized the notion of second-order flex (see [6] for bar frameworks) to general tensegrity frameworks. In the next theorem we will show that a nontrivial analytic flex of a tensegrity framework

gives rise to a second-order flex $(\mathbf{p}', \mathbf{p}'')$ whose first-order part $\mathbf{p}'$ is nontrivial. The natural idea is to take the first and second derivatives of the analytic flex evaluated at the starting point. Unfortunately, this may not work because the first derivative of the analytic flex may be trivial, and we may have to wait for some higher derivative to be nontrivial. For cables and struts we use the principle that the first nonvanishing derivative of the member length squared has the correct sign.

THEOREM 4.3.1. *If a tensegrity framework $G(\mathbf{p})$ is second-order rigid, then it is rigid.*

Proof. Assume $G(\mathbf{p})$ is not rigid. Then we will show that $G(\mathbf{p})$ is not second-order rigid by finding a second-order flex $(\mathbf{q}', \mathbf{q}'')$ such that $\mathbf{q}'$ is not 1-trivial at $\mathbf{p}$.

Since $G(\mathbf{p})$ is not rigid we know that $G(\mathbf{p})$ has a nonrigid analytic flex $\mathbf{p}(t)$ by Definition 2.1.2 (c). (See [6] or [9].) Define, for $\ell = 1, 2, \ldots,$

$$\mathbf{p}^{(\ell)} = D_T^{\ell}\mathbf{p}(t)\Big|_{t=0}.$$

Suppose for all $k = 1, 2, \ldots, \mathbf{p}', \ldots, \mathbf{p}^{(k)}$ is k-trivial. Then for any $\{i, j\}$, not just those members of G, for all $k = 1, 2, \ldots,$

$$D_t^k[|\mathbf{p}_i(t) - \mathbf{p}_j(t)|^2]\Big|_{t=0} = 0,$$

which implies that $|\mathbf{p}_i(t) - \mathbf{p}_j(t)|^2$ is constant in t, which implies that $\mathbf{p}(t)$ is a rigid analytic flex, contradicting the choice of $\mathbf{p}(t)$. Thus for some $k \geq 1$, $\mathbf{p}', \ldots, \mathbf{p}^{(k)}$ is not k-trivial.

Now let k be the smallest positive integer such that $\mathbf{p}', \ldots, \mathbf{p}^{(k)}$ is not k-trivial, fixing k. (If $k = 1$, life is especially easy.) Applying Proposition 4.2.1, we can alter $\mathbf{p}(t)$ so that not only is $\mathbf{p}', \ldots, \mathbf{p}^{(k-1)}$ $(k-1)$-trivial, but $\mathbf{p}' = \cdots = \mathbf{p}^{(k-1)} = 0$. (If $k = 1$ we do nothing.) Note that by Lemma 4.2.3, $0 = \mathbf{p}', \ldots, \mathbf{p}^{(k)}$ is still not k-trivial.

We observe that for any $\{i, j\}$,

$$\begin{aligned} D_t^k[|\mathbf{p}_i(t) - \mathbf{p}_j(t)|^2]\Big|_{t=0} &= \sum_{\ell=0}^{k} \binom{k}{\ell} (\mathbf{p}_i^{(\ell)} - \mathbf{p}_i^{(\ell)}) \cdot (\mathbf{p}_i^{(k-\ell)} - \mathbf{p}_j^{(k-\ell)}) \\ &= 2(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i^{(k)} - \mathbf{p}_j^{(k)}). \end{aligned}$$

Hence $\mathbf{p}^{(k)}$ is a first-order flex for $G(\mathbf{p})$. By Lemma 4.2.2, $\mathbf{p}^{(k)}$ is not 1-trivial. Now we define

$$\mathbf{q}' = \mathbf{p}^{(k)}.$$

We now proceed to find $\mathbf{q}''$. We still must pay attention to conditions analogous to first-order conditions. Recall that E_0 is the set of bars of G, E_- is the set of cables of G, and E_+ is the set of struts of G. For every $n = k, k+1, \ldots, 2k-1$, define

$$E_n = \left\{ \{i, j\} \in E_0 \cup E_- \cup E_+ \;\middle|\; \begin{array}{l} (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i^{(\ell)} - \mathbf{p}_j^{(\ell)}) = 0 \quad \text{for } \ell = 1, \ldots, n-1 \\ (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_j^{(n)} - \mathbf{p}_j^{(n)}) \neq 0 \end{array} \right\}.$$

Note that when $m = 1, \ldots, n$, and $\{i, j\} \in E_n$, or when $\{i, j\}$ is a bar,

$$\begin{aligned} D_t^m[|\mathbf{p}_i(t) - \mathbf{p}_j(t)|^2]\Big|_{t=0} &= \sum_{\ell=0}^{m} \binom{m}{\ell} (\mathbf{p}_i^{(\ell)} - \mathbf{p}_j^{(\ell)}) \cdot (\mathbf{p}_i^{(m-\ell)} - \mathbf{p}_j^{(m-\ell)}) \\ &= 2(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i^{(m)} - \mathbf{p}_j^{(m)}) \\ &\begin{cases} = 0 & \text{if } \{i, j\} \in E_0 \\ = 0 & \text{if } m = 1, 2, \ldots, n-1 \text{ and } \{i, j\} \in E_n \\ < 0 & \text{if } m = n \text{ and } \{i, j\} \in E_n \cap E_- \\ > 0 & \text{if } m = n \text{ and } \{i, j\} \in E_n \cap E_+ \end{cases}, \end{aligned}$$

since either $\mathbf{p}^{(\ell)} = 0$ or $\mathbf{p}^{(m-\ell)} = 0$ if $\ell = 1, \ldots, m-1 \leq 2k-1$. (Note that only cables or struts are in any E_n.) In other words, for just those members in E_n, $\mathbf{p}^{(n)}$ acts as a strict first-order flex of $G(\mathbf{p})$.

We will next find a sequence of real numbers $\varepsilon_1 >> \varepsilon_2 >> \varepsilon_{k-1} > 0$ and define

$$\mathbf{r}' = \mathbf{p}^{(k)} + \varepsilon_1 \mathbf{p}^{(k+1)} + \varepsilon_1 \mathbf{p}^{(k+2)} + \cdots + \varepsilon_{k-1} \mathbf{p}^{(2k-1)},$$

where $\varepsilon_i >> \varepsilon_{i+1}$ means that ε_{i+1} is chosen sufficiently small such that later inequalitjes will remain satisfied. We see that $\mathbf{r}'$ is also a (nontrivial) first-order flex of $G(\mathbf{p})$. In fact we require that for $\{i,j\}$ a member of G,

$$(7) \qquad (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{r}'_i - \mathbf{r}'_j) \left\{ \begin{array}{ll} < 0 & \text{if } \{i,j\} \in (E_k \cup \cdots \cup E_{2k-1}) \cap E_- \\ > 0 & \text{if } \{i,j\} \in (E_k \cup \cdots \cup E_{2k-1}) \cap E_+ \\ = 0 & \text{otherwise} \end{array} \right\}.$$

To see that this is possible we proceed by induction. Define for $n = k, k+1, \ldots, 2k+1$,

$$\mathbf{r}'(n) = \mathbf{p}^{(k)} + \varepsilon_1 \mathbf{p}^{(k+1)} + \cdots + \varepsilon_{n-k} \mathbf{p}^{(n)},$$

where $\mathbf{r}'(k) = \mathbf{p}^{(k)}$. We require that for $\{i,j\}$ a member of G,

$$(8) \qquad (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{r}'_i(n) - \mathbf{r}'_j(n)) \left\{ \begin{array}{ll} < 0 & \text{if } \{i,j\} \in (E_k \cup \cdots \cup E_n) \cap E_- \\ > 0 & \text{if } \{i,j\} \in (E_k \cup \cdots \cup E_n) \cap E_+ \\ = 0 & \text{otherwise} \end{array} \right\}.$$

But this is true for $n = k$, and if $\varepsilon_{n+1} > 0$ is chosen small enough we can satisfy (8) for $n+1$, assuming it is true for n.

In other words for every cable and strut of G, either $\mathbf{r}'$ is strict or $\mathbf{r}'$ and $\mathbf{p}', \ldots, \mathbf{p}^{(2k-1)}$ act as if $\{i,j\}$ were a bar for the first-order conditions.

We now choose a large real number $B > 0$ and define

$$\mathbf{q}'' = \frac{2}{\binom{2k}{k}} \mathbf{p}^{(2k)} + B\mathbf{r}'.$$

Recalling that $\mathbf{q}' = \mathbf{p}^{(k)}$, we claim that $(\mathbf{q}', \mathbf{q}'')$ is a second-order flex of $G(\mathbf{p})$ for B large enough. For $\{i,j\} \in E_k \cup \cdots \cup E_{2k-1}$ we calculate

$$\begin{aligned} &(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{q}''_i - \mathbf{q}''_j) + |\mathbf{q}'_i - \mathbf{q}'_j|^2 \\ &\qquad = \frac{2}{\binom{2k}{k}} (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i^{(2k)} - \mathbf{p}_j^{(2k)}) + B(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{r}'_i - \mathbf{r}'_j) + |\mathbf{q}'_i - \mathbf{q}'_j|^2 \\ &\qquad \left\{ \begin{array}{ll} > 0 & \text{if } \{i,j\} \in E_- \cap (E_k \cup \cdots \cup E_{2k-1}) \\ < 0 & \text{if } \{i,j\} \in E_+ \cap (E_k \cup \cdots \cup E_{2k-1}) \end{array} \right\} \end{aligned}$$

if B is chosen large enough by (7).

For $\{i,j\}$ a member of G but $\{i,j\} \notin E_k \cup \cdots E_{2k-1}$, then $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{r}'_i - \mathbf{r}'_j) = 0$ and $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_I^{(\ell)} - \mathbf{p}_j^{(\ell)}) = 0$ for $\ell = 1, \ldots, 2k-1$. Then

$$\begin{aligned} D_t^{2k}[|\mathbf{p}_i(t) - \mathbf{p}_j(t)|^2]\Big|_{t=0} &= \sum_{\ell=0}^{2k} \binom{2k}{\ell} (\mathbf{p}_i^{(\ell)} - \mathbf{p}_j^{(\ell)}) \cdot (\mathbf{p}_i^{(2k-\ell)} - \mathbf{p}_j^{(2k-\ell)}) \\ &= 2(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i^{(2k)} - \mathbf{p}_j^{(2k)}) + \binom{2k}{k} |\mathbf{p}_i^{(k)} - \mathbf{p}_j^{(k)}|^2 \\ &= \binom{2k}{k} [(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{q}_i - \mathbf{q}_j) + |\mathbf{q}'_i - \mathbf{q}'_j|^2] \\ &\left\{ \begin{array}{ll} \leq 0 & \text{if } \{i,j\} \in E_- \setminus (E_k \cup \cdots \cup E_{2k-1}) \\ = 0 & \text{if } \{i,j\} \in E_0 \setminus (E_k \cup \cdots \cup E_{2k-1}) \\ \geq 0 & \text{if } \{i,j\} \in E_+ \setminus (E_k \cup \cdots \cup E_{2k-1}). \end{array} \right. \end{aligned}$$

Thus in either case the second-order conditions are satisfied, $(\mathbf{q}', \mathbf{q}'')$ is a second-order flex of $G(\mathbf{p})$, and $\mathbf{q}'$ is not 1-trivial. □

Remark 4.3.1. The general outline for the above proof is the same as in [6], except that the cables and struts can cause complications. Differentiating the edge length condition allows us to detect any cable or strut, but its occurrence causes an appropriate sign somewhere from the level k to $2k$. The intermediate levels from $k+1$ to $2k-1$ must be introduced into the $(\mathbf{q}', \mathbf{q}'')$ carefully.

4.4. Prestress stability and second-order rigidity. We observe that prestress stability is stronger than second-order rigidity.

THEOREM 4.4.1. *If a tensegrity framework $G(\mathbf{p})$ is prestress stable, then it is second-order rigid.*

Proof. Let ω be the prestress that stabilizes $G(\mathbf{p})$. Then from the comments following the definition of prestress stability, ω also stabilizes that subframework of $G(\mathbf{p})$ consisting of the same bars and all cables and struts with $\omega_{ij} \neq 0$. Thus, without any loss of generality, we can assume that ω is strict. For example, $\omega_{ij} \neq 0$ on all the cables and struts of G.

Suppose $(\mathbf{p}', \mathbf{p}'')$ is a second-order flex of $G(\mathbf{p})$, where $\mathbf{p}'$ is not a trivial first-order flex. We wish to find a contradiction. By the first-order stress test we see that $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}'_i - \mathbf{p}'_j) = 0$ for all members of G (because $\omega_{ij} \neq 0$ on all cables and struts). Thus, by the second-order condition,

$$|\mathbf{p}'_i - \mathbf{p}'_j|^2 + (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}''_i - \mathbf{p}''_j) \begin{cases} \leq 0 & \{i,j\} = \text{ cable} \\ = 0 & \{i,j\} = \text{ bar} \\ \geq 0 & \{i,j\} = \text{ strut} \end{cases}$$

for all members $\{i,j\}$ of G. In any case, since ω is proper for all members $\{i,j\}$ of G,

$$\omega_{ij}|\mathbf{p}'_i - \mathbf{p}'_j|^2 + \omega_{ij}(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i - \mathbf{p}_j) \leq 0.$$

But since $\mathbf{p}'$ is a nontrivial first-order flex of $G(\mathbf{p})$ and since ω is a stabilizing self stress for $G(\mathbf{p})$,

$$\omega R(\mathbf{p}')\mathbf{p}' = \sum_{ij} \omega_{ij}|\mathbf{p}'_i - \mathbf{p}'_j|^2 = (\mathbf{p}')^T \Omega \mathbf{p}' > 0$$

by Proposition 3.4.2. Recalling that $\omega R(\mathbf{p}) = 0$,

$$\begin{aligned} 0 < \omega R(\mathbf{p}')\mathbf{p}' &= \omega R(\mathbf{p}')\mathbf{p}' + \omega R(\mathbf{p})\mathbf{p}'' \\ &= \sum_{ij} \omega_{ij}(\mathbf{p}'_i - \mathbf{p}'_j)^2 + \omega_{ij}(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}''_i - \mathbf{p}''_j) \leq 0, \end{aligned}$$

a contradiction. Thus $G(\mathbf{p})$ is second-order rigid. □

Remark 4.4.1. It turns out that there are tensegrity frameworks that are not prestress stable for any prestress yet are still second-order rigid. For instance, the example of Figure 9b has this property. The two "tetrahedral" blocks are prestress stable. Therefore, all second-order flexes are trivial on these blocks. Any such second-order flex must extend to a rotation about the common point 0. However, this violates either the strut or the cable condition on the unstressed connecting members at first-order.

However, in §5.3 we will see that if the space of first-order flexes or the space of proper self stresses is one-dimensional, then second-order rigidity and prestress

stability are the same. This will also help us to find examples of bar frameworks which are second-order rigid but not prestress stable in §5.3.

5. The stress test.

5.1. Duality from linear algebra. We now formulate some well-known principles of duality in linear algebra which we will later interpret as a "stress" test for second-order rigidity. These duality principles are a special case of the duality principles used in linear programming.

In the following, let A be a d-by-e real matrix, where we write A in block form

$$A = \begin{bmatrix} A_0 \\ A_+ \end{bmatrix},$$

where A_0 and A_+ are some designated subsets of the rows of A. In our applications A will correspond to the rigidity matrix, A_0 will correspond to the rows indexed by the bars of G, and A_+ will correspond to the rows indexed by the struts and cables of G, with the strut rows multiplied by -1. However, for the general statements in this section we will not need any special properties of A.

We can now restate the first-order stress test in this somewhat more general context. We use the notation $[x_1, x_2, \ldots] < 0$ for vectors to mean $x_i < 0$ for all $i = 1, 2, \ldots$.

PROPOSITION 5.1.1. *There is a column vector* $\mathbf{x} \in \mathbb{R}^d$ *such that*

$$\begin{aligned} A_0\mathbf{x} &= 0, \\ A_+\mathbf{x} &< 0, \end{aligned}$$

if and only if for all row vectors $\mathbf{y} \in \mathbb{R}^e$, $\mathbf{y} = [\mathbf{y}_0, \mathbf{y}_+]$, *such that*

$$\mathbf{y}_0 A_0 + \mathbf{y}_+ A_+ = 0,$$

$$\mathbf{y}_+ \geq 0;$$

then $\mathbf{y}_+ = 0$.

This is a special case of the duality principle for homogeneous linear equalities, for instance, as found in [32, Thm. 6]. Note that the "only if" implication is easy, since if $A_0\mathbf{x} = 0$, $A_+\mathbf{x} < 0$, $\mathbf{y}_0 A_0 + \mathbf{y}_+ A_+ = 0$, $\mathbf{y}_+ \geq 0$, then $0 = \mathbf{y}_0 A_0\mathbf{x} + \mathbf{y}_+ A_+\mathbf{x} \leq 0$. If $\mathbf{y}_+ \neq 0$ this gives a strict inequality and thus a contradiction. The other implication implicitly or explicitly uses the principle of "hyperplane separation." See, for example, [17, p. 10].

An important point is the strictness of the inequalities. However, in the following we instead concentrate on the duality principle itself, putting aside the strictness properties for the moment.

Let

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_+ \end{bmatrix} \in \mathbb{R}^e$$

be a column vector.

PROPOSITION 5.1.2. *There is a column vector* $\mathbf{x} \in \mathbb{R}^d$ *such that*

$$\begin{aligned} A_0\mathbf{x} &= \mathbf{b}_0, \\ A_+\mathbf{x} &\leq \mathbf{b}_+, \end{aligned}$$

if and only if for all row vectors $\mathbf{y} \in \mathbb{R}^e$, $\mathbf{y} = [\mathbf{y}_0, \mathbf{y}_+]$, *such that*

$$\mathbf{y}_0 A_0 + \mathbf{y}_+ A_+ = 0,$$
$$\mathbf{y}_+ \geq 0;$$

then $\mathbf{y}_0\mathbf{b}_0 + \mathbf{y}_+\mathbf{b}_+ \geq 0$.

Note that again the "only if" implication is easy since if $A_0\mathbf{x} = \mathbf{b}_0$, $A_+\mathbf{x} \leq \mathbf{b}_+$, $\mathbf{y}_0 A_0 + \mathbf{y}_+ A_+ = 0$, and $\mathbf{y}_+ \geq 0$ then $0 = \mathbf{y}_0 A_0\mathbf{x} + \mathbf{y}_+ A_+\mathbf{x} \leq \mathbf{y}_0\mathbf{b}_0 + \mathbf{y}_+\mathbf{b}_+$, and again the "if" implication follows from the hyperplane separation principle.

In the terminology of linear programming this proposition is an asymmetric form of duality in the special case when the primal problem has the constant 0 objective function. See [15] for instance for a discussion of various such forms of the Farkas alternative as well as a proof.

We now sharpen this proposition to obtain an equivalent dual reformulation to determine when we get strict inequality. We fix A and $\mathbf{b}$.

PROPOSITION 5.1.3. *There is a column vector* $\mathbf{x} \in \mathbb{R}^d$ *such that*

$$A_0\mathbf{x} = \mathbf{b}_0,$$
$$A_+\mathbf{x} < \mathbf{b}_+,$$

if and only if for all row vectors $\mathbf{y} \in \mathbb{R}^e$, $\mathbf{y} = [\mathbf{y}_0, \mathbf{y}_+]$, *such that*

$$\mathbf{y}_0 A_0 + \mathbf{y}_+ A_+ = 0,$$
$$\mathbf{y}_+ \geq 0;$$

then $\mathbf{y}_0\mathbf{b}_0 + \mathbf{y}_+\mathbf{b}_+ \geq 0$ *with equality if and only if* $\mathbf{y}_+ = 0$.

Proof. Again, the only if implication follows easily. $A_0\mathbf{x} = \mathbf{b}_0$, $A_+\mathbf{x} < \mathbf{b}_+$, $\mathbf{y}_0 A_0 + \mathbf{y}_+ A_+ = 0$, $\mathbf{y}_+ \geq 0$ imply that $0 = \mathbf{y}_0 A_0\mathbf{x} + \mathbf{y}_+ A_+\mathbf{x} \leq \mathbf{y}_0\mathbf{b}_0 + \mathbf{y}_+\mathbf{b}_+$ and we have strict inequality, and a contradiction, if and only if $\mathbf{y}_+ \neq 0$.

To show the converse we use the two previous propositions. Assume that the condition on the $\mathbf{y}$ vector holds. By Proposition 5.1.2 we know that there is an $\mathbf{x} \in \mathbb{R}^d$ such that $A_0\mathbf{x} = \mathbf{b}_0$, $A_+\mathbf{x} \leq \mathbf{b}_+$. If all of the $\mathbf{b}_+$ inequalities are strict we are done. If not, throw out those strict inequalities from A to get the condition $A_0\mathbf{x} = \mathbf{b}_0$, $A_+\mathbf{x} = \mathbf{b}_+$. Similarly, we can throw out the corresponding set of variables in the $\mathbf{y}$ vector. Now it is still true that if $\mathbf{y}_0 A_0 + \mathbf{y}_+ A_+ = 0$, then $\mathbf{y}_0 A_0\mathbf{x} + \mathbf{y}_+ A_+\mathbf{x} = \mathbf{y}_0\mathbf{b}_0 + \mathbf{y}_+\mathbf{b}_+ = 0$. Thus Proposition 5.1.1 applies, and there is a (small) $\mathbf{x}_\varepsilon \in \mathbb{R}^e$ such that $A_0\mathbf{x}_\varepsilon = 0$, $A_+\mathbf{x}_\varepsilon < 0$. Then $A_0(\mathbf{x} + \mathbf{x}_\varepsilon) = \mathbf{b}_0$, $A_+(\mathbf{x} + \mathbf{x}_\varepsilon) < \mathbf{b}_+$. If $\mathbf{x}_\varepsilon$ is small enough then $\mathbf{x} + \mathbf{x}_\varepsilon$ still satisfies those inequalities that were thrown out as well. □

5.2. Interpretation as the stress test. We now specialize the results of §5.1 to the case of the rigidity matrix. Let $G(\mathbf{p})$ be a tensegrity framework in $\mathbb{R}^d$, and let $R(\mathbf{p})$ be its d-by-e rigidity matrix. Let $R_0(\mathbf{p})$ denote those rows (regarded as a smaller matrix) corresponding to the bars of G. Let $R_+(\mathbf{p})$ denote the matrix obtained from those rows of $R(\mathbf{p})$ corresponding to cables and struts of G, with the rows corresponding to struts multiplied by -1.

Suppose ω is a proper stress for $G(\mathbf{p})$, so $\omega R(\mathbf{p}) = 0$. We then have a corresponding $[\omega_0, \omega_+]$, where ω_0 corresponds to the stresses on the bars and ω_+ to the stresses on the cables and struts, but with the opposite sign for struts only. Thus ω being proper translates into $\omega_+ \geq 0$, and being a self stress means $\omega_0 R(\mathbf{p}) + \omega_+ R_+(\mathbf{p}) = 0$.

In this terminology $\mathbf{p}'$ is a first-order flex if

$$R_0(\mathbf{p})\mathbf{p}' = 0,$$
$$R_+(\mathbf{p})\mathbf{p}' \leq 0,$$

and $\mathbf{p}'$, $\mathbf{p}''$ is a second-order flex if in addition

$$R_0(\mathbf{p}')\mathbf{p}' + R_0(\mathbf{p})\mathbf{p}'' = 0,$$
$$R_+(\mathbf{p}')\mathbf{p}' + R_+(\mathbf{p})\mathbf{p}'' \leq 0,$$

where an inequality need only hold when the corresponding inequality, in the first-order system, is an equality. Recall that $\mathbf{p}''$ (or $\mathbf{p}'$) is strict for $\{i,j\}$ a cable or strut if the corresponding inequality is strict.

We now have our strict second-order duality result.

COROLLARY 5.2.1 (the second-order stress test). *A first-order flex $\mathbf{p}'$ of $G(\mathbf{p})$ extends to a second-order flex $(\mathbf{p}', \mathbf{p}'')$ if and only if for all proper self stresses ω for $G(\mathbf{p})$, with stress matrix Ω,*

$$(\mathbf{p}')^T\Omega\mathbf{p}' \leq 0.$$

Furthermore, $\mathbf{p}''$ can be chosen to be strict on each cable and strut $\{i,j\}$ where $\mathbf{p}'$ is not strict, if and only if for all proper self stresses $\omega, \mathbf{p}'\Omega\mathbf{p}' = 0$ implies $\omega_{ij} = 0$, for each such $\{i,j\}$.

Proof. We apply Proposition 5.1.3, where

$$\mathbf{p} = \mathbf{x},$$

$$R_0(\mathbf{p}) = A_0, \qquad -R_0(\mathbf{p}')\mathbf{p}' = \mathbf{b}_0,$$
$$R_+(\mathbf{p}) = A_+, \qquad -R_+(\mathbf{p}')\mathbf{p}' = \mathbf{b}_+,$$
$$\omega_0 = \mathbf{y}_0, \quad \omega_+ = \mathbf{y}_+.$$

We may assume, without loss of generality, that $R(\mathbf{p})\mathbf{p}' = 0$, since any cable or strut where $\mathbf{p}'$ is strict can be disregarded as a cable or strut for the second-order conditions.

The second-order conditions translate into the hypothesis of Proposition 5.1.2, and the conclusion translates into the condition that ω is a proper self stress. Then

$$0 \leq \mathbf{y}_0\mathbf{b}_0 + \mathbf{y}_+\mathbf{b}_+ = -\omega_0 R_0(\mathbf{p}')\mathbf{p}' - \omega_+ R_+(\mathbf{p}')\mathbf{p}' = -\omega R(\mathbf{p}')\mathbf{p}' = -(\mathbf{p}')^T\Omega\mathbf{p}'$$

is the condition desired. The strictness follows from Proposition 5.1.3. □

We can simplify matters even further when G consists only of bars. This is our second-order duality result for bar frameworks.

COROLLARY 5.2.2 (second-order stress test for bars). *A first-order flex $\mathbf{p}'$ of a bar framework $G(\mathbf{p})$ extends to a second-order flex if and only if for all self stresses ω for $G(\mathbf{p})$, with stress matrix Ω,*

$$(\mathbf{p}')^T\Omega\mathbf{p}' = 0.$$

Remark 5.2.1. In the appendix, we show that we can always replace a framework (with a strict proper self stress) by an equivalent bar framework and use this to check second-order rigidity. However, it seems simpler to use Corollary 5.2.1 directly, rather than introduce so many extraneous members.

5.3. Second-order rigidity and prestress stability. When does second-order rigidity imply prestress stability? We begin with cases when the set of self stresses or the set of equilibrium first-order flexes is one-dimensional, the natural first cases to consider.

Note that for a fixed tensegrity framework $G(\mathbf{p})$, the proper self stress and first-order flexes each form a cone with the origin as the cone point.

PROPOSITION 5.3.1. *If a tensegrity framework $G(\mathbf{p})$ is second-order rigid with either a one-dimensional cone of equilibrium first-order flexes or a one-dimensional cone of proper self stresses, then $G(\mathbf{p})$ is prestress stable.*

Proof. Suppose $\mathbf{p}'$ is any nontrivial equilibrium first-order flex of $G(\mathbf{p})$ generating the one-dimensional cone of all equilibrium first-order flexes. If for all proper self stresses ω with stress matrix Ω we have

$$t^2(\mathbf{p}')^T\Omega\mathbf{p}' = (t\mathbf{p}')^T\Omega t\mathbf{p}' \leq 0$$

for all first-order flexes $t\mathbf{p}'$ of $G(\mathbf{p})$ (t a real scalar), then by Corollary 5.2.1 $\mathbf{p}'$ extends to a second-order flex $(\mathbf{p}', \mathbf{p}'')$ of $G(\mathbf{p})$, which contradicts $G(\mathbf{p})$ being second-order rigid. Thus for some proper self stress ω, $(\mathbf{p}')^T\Omega\mathbf{p}' > 0$, ω stabilizes $G(\mathbf{p})$ (by Proposition 3.4.2) and $G(\mathbf{p})$ is prestress stable.

Suppose ω is a proper nonzero self stress in the one-dimensional cone of proper self stresses. Suppose there is a nontrivial first-order flex $\mathbf{p}'$ such that $(\mathbf{p}')^T\Omega\mathbf{p}' \leq 0$. If $-\omega$ is not a proper stress, then $t\omega$, $t \geq 0$, are the only proper self stresses for $G(\mathbf{p})$. Then by Corollary 5.2.1 again $G(\mathbf{p})$ would not be second-order rigid, contradicting the hypothesis. Therefore either $(\mathbf{p}')^T\Omega\mathbf{p}' > 0$ for all nontrivial first-order flexes $\mathbf{p}'$ *or* $-\omega$ is a proper self stress. If $-\omega$ is a proper self stress, then $(\mathbf{p}')^T\Omega\mathbf{p}' = 0$ and again $\mathbf{p}'$ would extend to a second-order flex. Thus $(\mathbf{p}')^T(\pm\Omega)\mathbf{p}' > 0$ and $\pm\omega$ stabilizes $G(\mathbf{p})$. □

We have already seen an example of tensegrity framework in the plane, Figure 9b, which is easily seen to be second-order rigid, directly from the definition, but is not prestress stable for any proper self stress. Here we present another example, but one which is a bar framework in three-space. It also serves as an example of how to calculate using the stress test.

If we have any bar framework $G(\mathbf{p})$, let

$$\mathbf{p}'(1), \ldots, \mathbf{p}'(n)$$

denote a basis for a space of nontrivial first-order flexes of $G(\mathbf{p})$. Let

$$\Omega(1), \ldots, \Omega(m)$$

denote a basis for the space of self stresses of $G(\mathbf{p})$. If $G(\mathbf{p})$ is prestress stable, some linear combination of the stress matrices must be positive definite on the space generated by the first-order flexes $\mathbf{p}'(1), \ldots, \mathbf{p}'(n)$. From Corollary 5.2.2, the second-order stress test for bar frameworks, $G(\mathbf{p})$ is not second-order rigid if and only if all of the stress matrices have a common nonzero vector on which they evaluate to be 0 in this same space generated by $\mathbf{p}'(1), \ldots, \mathbf{p}'(n)$. When $n = 2$, both of these criteria can be checked with certain easily calculated expressions.

Example 5.3.1. We define a specific example in three-space. Let $G(\mathbf{p})$ be the following bar framework in three-space with the following seven vertices:

$$\begin{array}{lll} \mathbf{p}_1 = (0,0,0), & \mathbf{p}_4 = (1,0,0), & \mathbf{p}_7 = (1,0,0), \\ \mathbf{p}_2 = (0,1,0), & \mathbf{p}_5 = (1,1,0), & \\ \mathbf{p}_3 = (0,0,1), & \mathbf{p}_6 = (0,1,0), & \end{array}$$

and the following bars:

$$\{1,2\},\{1,3\},\{1,4\},\{1,6\},\{1,7\},$$
$$\{2,3\},\{2,5\},\{2,7\},$$
$$\{3,4\},\{3,5\},$$
$$\{4,5\},\{4,6\},$$
$$\{5,6\},\{5,7\},$$
$$\{6,7\}.$$

See Figure 17a, where although $\mathbf{p}_2 = \mathbf{p}_6$ and $\mathbf{p}_4 = \mathbf{p}_7$ we have separated them slightly so that the framework can be more easily understood.

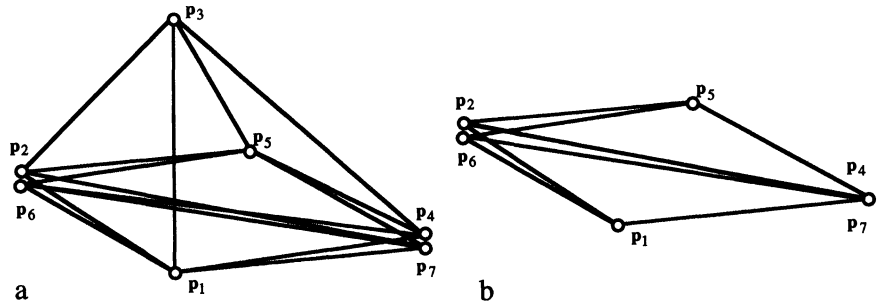


FIG. 17. *A bar framework* (a) *in* 3*-space that is second-order rigid but not prestress stable. A portion of the planar base is shown in* (b).

Note that this framework is made of the framework of Figure 17b and its symmetric copy, with appropriate identifications. Then $\mathbf{p}_3$ is added along the z-axis.

We consider those first-order flexes $\mathbf{p}'(k)$, where $\mathbf{p}'_1(k) = \mathbf{p}'_2(k) = \mathbf{p}'_3(k) = 0$, which clearly determines a complement of the space of trivial flexes, since $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ determines a bar triangle. Since $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_7)$ and $(\mathbf{p}_2, \mathbf{p}_5, \mathbf{p}_7)$ are bar triangles in the same plane, sharing a common bar $\mathbf{p}_1$, $\mathbf{p}_2$, any first-order flex $\mathbf{p}'$ must have

$$(\mathbf{p}_1 - \mathbf{p}_5) \cdot (\mathbf{p}'_1 - \mathbf{p}'_5) = 0.$$

In other words, $\{1,5\}$ is an "implied bar." Thus the first-order rigid tetrahedron $(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_5)$ is implied and $\mathbf{p}'_5 = 0$. Similarly $\mathbf{p}'_4 = 0$, from the implied tetrahedron $(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_5, \mathbf{p}_4)$. See [9]. So the following first-order flexes are a basis for a complementary space:

$$\mathbf{p}'_i(1) = \left\{ \begin{array}{ll} (0,0,1) & \text{if} \quad i = 6 \\ (0,0,0) & \text{otherwise} \end{array} \right\},$$

$$\mathbf{p}'_i(2) = \left\{ \begin{array}{ll} (0,0,1) & \text{if} \quad i = 7 \\ (0,0,0) & \text{otherwise} \end{array} \right\}.$$

We can also find two independent stresses for $G(\mathbf{p})$:

$$\omega_{ij}(1) = \left\{ \begin{array}{lll} 1 & \text{if} & \{i,j\} = \{2,7\},\{5,6\}, \quad \text{or} \quad \{1,6\} \\ -1 & \text{if} & \{i,j\} = \{2,5\},\{6,7\}, \quad \text{or} \quad \{2,1\} \\ 0 & \text{otherwise} & \end{array} \right\},$$

$$\omega_{ij}(2) = \left\{ \begin{array}{ll} 1 & \text{if} \quad \{i,j\} = \{1,7\},\{5,7\}, \quad \text{or} \quad \{4,6\} \\ -1 & \text{if} \quad \{i,j\} = \{1,4\},\{6,7\}, \quad \text{or} \quad \{4,5\} \\ 0 & \text{otherwise} \end{array} \right\}.$$

These are easy to see by looking at Figure 17b. Notice that $e = 15 = 3v - 6$ and that the space of first-order nontrivial (equilibrium) flexes must be of dimension 2, so the dimension of the space of self stresses is 2 as well. Thus $\omega(1)$ and $\omega(2)$ generate all the self stresses.

We next calculate the stress matrices $\Omega(1)$, $\Omega(2)$ corresponding to $\omega(1)$, $\omega(2)$, relative to the vectors $\mathbf{p}'(1)$, $\mathbf{p}'(2)$. Note that for $a = 1,2$, $b = 1,2$, $k = 1,2$,

$$\mathbf{p}'(a)^T \Omega(k) \mathbf{p}'(b) = \Omega_{ab}(k) = \sum_{ij} \omega_{ij}(k)(\mathbf{p}'_i(a) - \mathbf{p}'_j(a)) \cdot (\mathbf{p}'_i(b) - \mathbf{p}'_j(b)).$$

Then

$$\Omega(1) = \begin{bmatrix} \sum_i \omega_{6i}(1) & -\omega_{67}(1) \\ -\omega_{76}(1) & \sum_i \omega_{7i}(1) \end{bmatrix} = \begin{bmatrix} 1 & +1 \\ +1 & 0 \end{bmatrix},$$

$$\Omega(2) = \begin{bmatrix} \sum_i \omega_{6i}(2) & -\omega_{67}(2) \\ -\omega_{76}(3) & \sum_i \omega_{7i}(2) \end{bmatrix} = \begin{bmatrix} 0 & +1 \\ +1 & 1 \end{bmatrix}.$$

To see if any linear combination of these is positive definite, we calculate, for any real λ_1, λ_2,

$$\det[\lambda_1 \Omega(1) + \lambda_2 \Omega(2)] = \det \begin{bmatrix} \lambda_1 & \lambda_1 + \lambda_2 \\ \lambda_1 + \lambda_2 & \lambda_2 \end{bmatrix} = -\lambda_1^2 - \lambda_1 \lambda_2 - \lambda_2^2,$$

which is a negative definite quadratic form itself, since $(-1)^2 - 4(-1)(-1) = -3 < 0$. Thus for each choice of $(\lambda_1, \lambda_2) \neq (0,0)$, $\det[\lambda_1 \Omega(1) + \lambda_2 \Omega(2)] < 0$, which implies that none of the forms $\lambda_1 \Omega(1) + \lambda_2 \Omega(2)$ are positive definite, and thus no stress $\lambda_1 \omega(1) + \lambda_2 \omega(2)$ can serve as a stable prestress.

On the other hand recall that the second-order stress test for bar frameworks, Corollary 5.2.2, says that a first-order flex $\mathbf{p}'$ will extend to a second-order flex if and only if $\mathbf{p}'$ is in the zero set of all the proper self stresses (regarded as quadratic forms) of $G(\mathbf{p})$. If some $\mathbf{p}'$ does extend, then so does $\mathbf{p}'$ plus any trivial first-order flex, and so we can assume that $\mathbf{p}'$ is in the space spanned by $\mathbf{p}'(1)$ and $\mathbf{p}'(2)$. Thus $G(\mathbf{p})$ will be second-order rigid if and only if $\Omega(1)$ and $\Omega(2)$ (and thus all $\lambda_1 \Omega(1) + \lambda_2 \Omega(2)$) have a common zero. The zeros of $\Omega(1)$ (as a quadratic form) are scalar multiples of

$$(0,1) \quad \text{or} \quad (2,-1).$$

For $\Omega(2)$ we get

$$(1,0) \quad \text{or} \quad (-1,2).$$

None of the above four vectors are scalar multiples of any of the others, so $G(\mathbf{p})$ is second-order rigid but not prestress stable. □

Remark 5.3.1. If the dimension of the space of nontrivial (equilibrium) first-order flexes I is two, then it is easy to determine when the framework is prestress stable. Calculate a basis of stress matrices restricted to I. Choosing an orthonormal basis for

I, we see that there are at most three such distinct stress matrices (even if the space of self stresses is higher-dimensional), since they correspond to symmetric 2×2 matrices. If there are just two such matrices, one can perform an analysis similar to Example 5.3.1. If there are three such stress matrices, independent over I, there always is a stabilizing self stress.

Also for Example 5.3.1 it is possible to vary the points by a small amount, keeping all the points except $\mathbf{p}_3$ in a plane, and obtain many other examples of second-order rigid but not prestress stable bar frameworks in three-space. □

Note that the underlying graph of the example above is a triangulated sphere. Figure 18a shows a realization of this graph as a triangulated convex surface. By [6], this realization is also second-order rigid. In fact it is prestress stable as well. All members adjacent to $\mathbf{p}_6$ and $\mathbf{p}_7$ have a positive stress in the stabilizing self stress. This brings up the question: are all triangulations of a convex polyhedron in three-space, with edges as bars, prestress stable? In [6] it is only shown that such frameworks are second-order rigid. The answer is yes and will be shown elsewhere.

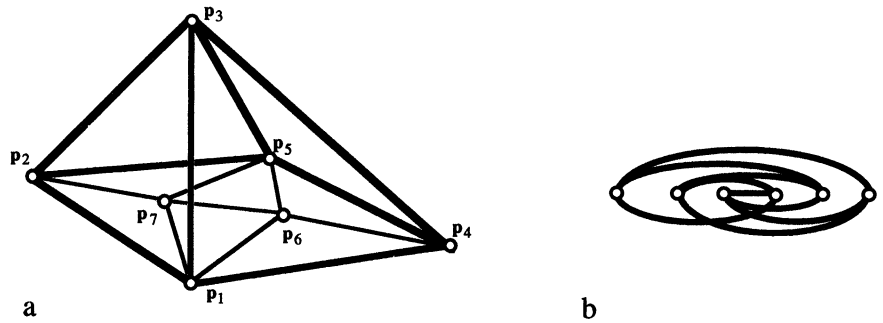


FIG. 18. *Example* (a), *with the same graph as Figure* 17a, *is prestress stable in three-space. Example* (b) *is second-order rigid in the plane but not prestress stable.*

In the plane, it turns out that if we take the bipartite graph $K_{3,3}$ with its six points on the line and $\{1, 2, 3\}$, $\{4, 5, 6\}$ as the partition (Figure 18b), then this framework $K_{3,3}(\mathbf{p})$ is second-order rigid but not prestress stable. We omit this nontrivial calculation. It also turns out that $K_{3,3}(\mathbf{p})$ is a mechanism in R^3.

In [26], as well as in [27, p. 50], there is another example of a second-order rigid but not prestress stable bar framework in the plane. The calculation for that example turns out to be quite simple.

5.4. Applications in *b-c* polygons. Here we apply the second-order stress test to a special class of frameworks $G(\mathbf{p})$: the *b-c* polygons of §3.5. That is, $G(\mathbf{p})$ is a convex polygon in the plane with bars as edges and only cables on the inside.

PROPOSITION 5.4.1. *Let* $\mathbf{p}'$ *be any nontrivial first-order flex of* $G(\mathbf{p})$, *a b-c polygon in the plane. Then* $\mathbf{p}'$ *extends to a strict second-order flex* $(\mathbf{p}', \mathbf{p}'')$ *of* $G(\mathbf{p})$.

Proof. Let Ω be any stress matrix coming from a proper self stress ω of $G(\mathbf{p})$. Since $\mathbf{p}'$ is nontrivial it is easy to check that $\mathbf{p}'$ is not an affine image of $\mathbf{p}$. By [7]

$$(\mathbf{p}')^T \Omega \mathbf{p}' < 0,$$

since for the reversed polygon (with struts on the inside) the corresponding matrix $(-\Omega)$ is positive semidefinite. Thus by Corollary 5.2.1, $\mathbf{p}'$ extends to a strict second-order flex $(\mathbf{p}', \mathbf{p}'')$. □

Remark 5.4.1. We will use this result in the next section to prove Roth's conjecture about *b-c* polygons. It is interesting (although painful) to calculate $\mathbf{p}''$ directly for the $\mathbf{p}'$ as indicated in Figure 14.

Note, however, with this result alone we can prove a weak form of Roth's conjecture. Namely, if $\mathbf{p}'$ is a nontrivial first-order flex of a strut-cable polygon (the outside edges are struts), then an argument similar to the one above can be used to find a strict second-order flex $(\mathbf{p}', \mathbf{p}'')$ as well. Then $\mathbf{p}(t) = \mathbf{p} + t\mathbf{p}' + \frac{1}{2}t^2\mathbf{p}''$ is a flex at $G(\mathbf{p})$ as required.

The only problem left in the stronger form of Roth's conjecture is to find a way of handling the bars. We will treat this in §6.

5.5. Interpretation for triangulated spheres. For any triangulated sphere $G(\mathbf{p})$ in $\mathbb{R}^3$, there is a natural correspondence between first-order flexes $\mathbf{p}'$ of $G(\mathbf{p})$ (modulo trivial first-order flexes) and self stresses ω of $G(\mathbf{p})$. In fact for each edge $\{i, j\}$ there is a dihedral angle θ_{ij}, which itself "varies" and thus there is a θ'_{ij} defined as the derivative of θ_{ij}. Then

$$\omega_{ij} = \frac{\theta'_{ij}}{|\mathbf{p}_i - \mathbf{p}_j|}, \quad \{i, j\} \quad \text{and edge at} \quad G$$

serves as a self stress for $G(\mathbf{p})$. Conversely, given a self stress it is possible to define a first-order flex $\mathbf{p}'$ with θ'_{ij} as above. See [16] or [14] for a discussion of this.

Thus using Corollary 5.2.1 we can state the dual condition for a second-order flex.

COROLLARY 5.5.1. *A first-order flex $\mathbf{p}'$ of a triangulated sphere $G(\mathbf{p})$ in $\mathbb{R}^3$ extends to a second-order flex if and only if for every θ'_{ij} (coming from a possibly different first-order flex) we have*

$$\sum_{ij} \frac{\theta'_{ij}}{|\mathbf{p}_i - \mathbf{p}_j|} |\mathbf{p}'_i - \mathbf{p}'_j|^2 = 0.$$

5.6. Interpretation in terms of packings. For the rigidity of packings as in [11] or [12] we see that the associated framework has certain vertices pinned and all the members are struts. For any proper self stress ω, $\omega_{ij} \leq 0$ for all $\{i, j\}$ struts, and thus such a $G(\mathbf{p})$ has for all $\mathbf{p}'$, a first-order flex,

$$(\mathbf{p}')^T \Omega \mathbf{p}' = \sum_{ij} \omega_{ij} (\mathbf{p}'_i - \mathbf{p}'_j)^2 \leq 0,$$

since we can take $\mathbf{p}'$ to be 0 on the pinned vertices. In fact, we get strict inequality assuming G is connected and $\mathbf{p}' \neq 0$. Thus there is a strict second-order flex $(\mathbf{p}', \mathbf{p}'')$, and it is easy to see that such a $G(\mathbf{p})$ is rigid if and only if it is first-order rigid. This was observed directly in [11].

6. Extending second-order flexes.

6.1. The general result. Some second-order flexes extend to continuous flexes of the framework. For example, if a second-order flex shortens all cables and lengthens all struts and there are no bars, then it is clear that we can complete these first two derivatives to a real analytic path. We describe a less restrictive situation where we can still extend the second-order flex. This result is an extension of and motivated by some of the results in [1, 2]. A bar framework $G(\mathbf{p})$ is called *independent* if the only self stress for $G(\mathbf{p})$ is the zero self stress.

PROPOSITION 6.1.1. *Let $G(\mathbf{p})$ be any independent bar framework with a second-order flex $(\mathbf{p}', \mathbf{p}'')$ in $\mathbb{R}^d$. Then there is an analytic flex $\mathbf{p}(t)$ of $G(\mathbf{p})$ with*

$$\begin{aligned} \mathbf{p}(0) &= \mathbf{p}, \\ D_t[\mathbf{p}(t)]\Big|_{t=0} &= \mathbf{p}', \\ D_t^2[\mathbf{p}(t)]\Big|_{t=0} &= \mathbf{p}''. \end{aligned}$$

Proof. Let

$$M_{G(\mathbf{p})} = \left\{ \mathbf{q} \in \mathbb{R}^{dv} \mid |\mathbf{q}_i - \mathbf{q}_j| = |\mathbf{p}_i - \mathbf{p}_j|, \quad \{i,j\} \quad \text{a member of} \quad G \right\}$$

be the set of all configurations equivalent to $\mathbf{p}$. By [1] or [29], since $G(\mathbf{p})$ is independent, $M_{G(\mathbf{p})}$ is a smooth analytic manifold of dimension at least $d(d+1)/2$ in a neighborhood of $\mathbf{p}$ (when the dimension of the affine span of $\mathbf{p}$ is at least $d-1$), and we may naturally identify the tangent space $T_{\mathbf{p}}$ of $M_{G(\mathbf{p})}$ at $\mathbf{p}$ with the first-order flexes of $G(\mathbf{p})$.

Let $h : T_{\mathbf{p}} \to M_{G(\mathbf{p})}$ be a smooth analytic map such that the following hold:

(a) On a neighborhood of $\mathbf{p}$ in $T_{\mathbf{p}}$, h is a real analytic diffeomorphism onto a neighborhood of $\mathbf{p}$ in $M_{G(\mathbf{p})}$.

(b) Identifying the tangent space of $T_{\mathbf{p}}$ with itself, $h(\mathbf{p}) = \mathbf{p}$ and $dh_{\mathbf{p}}(\mathbf{p}') = \mathbf{p}'$ for all $\mathbf{p}' \in T_{\mathbf{p}}$, where $dh_{\mathbf{p}}$ is the differential of h at $\mathbf{p}$.

For instance, the exponential map has these properties.

Let $\mathbf{q}(t) = \mathbf{p} + t\mathbf{p}' + \frac{1}{2}t^2\mathbf{q}$ be a smooth analytic path in $T_{\mathbf{p}}$ where we see that $\mathbf{q}(0) = \mathbf{p}$. $D_t[\mathbf{q}(t)]\big|_{t=0} = \mathbf{p}'$ and $D_t^2[\mathbf{q}(t)]\big|_{t=0} = \mathbf{q}''$, which will be determined later. Then define $\mathbf{p}(t) = h(\mathbf{q}(t))$, which is a smooth analytic flex of $G(\mathbf{p})$ with $\mathbf{p}(0) = \mathbf{p}$. Also

$$D_t[\mathbf{p}(t)] = D_t[h(\mathbf{q}(t))] = dh_{\mathbf{q}(t)}(D_t[\mathbf{q}(t)]) \tag{9}$$

and

$$D_t[\mathbf{p}(t)]\Big|_{t=0} = dh_{\mathbf{p}}(\mathbf{p}') = \mathbf{p}',$$

by condition (b). Since $\mathbf{p}(t) \in M_{G(\mathbf{p})}$, $\mathbf{p}(t)$ is an analytic flex of $G(\mathbf{p})$ and thus the second derivatives of the squares of the edge lengths are zero. Restating this in terms of the matrix $R(\mathbf{p})$ we get

$$R(D_t[\mathbf{p}(t)])D_t[\mathbf{p}(t)] + R(\mathbf{p}(t))D_t^2[\mathbf{p}(t)] = 0.$$

Evaluating when $t = 0$, we get

$$R(\mathbf{p}')\mathbf{p}' + R(\mathbf{p})\mathbf{r}'' = \mathbf{0},$$

where $\mathbf{r}'' = D_t^2[\mathbf{p}(t)]\big|_{t=0}$.

We must choose $\mathbf{q}''$ such that $\mathbf{r}'' = \mathbf{p}''$ the preassigned vector. Recalling that $(\mathbf{p}', \mathbf{p}'')$ is a second-order flex of $G(\mathbf{p})$ we see that for any $\mathbf{q}''$,

$$R(\mathbf{p}')\mathbf{p}' + R(\mathbf{p})\mathbf{p}'' = \mathbf{0} = R(\mathbf{p}')\mathbf{p}' + R(\mathbf{p})\mathbf{r}''.$$

So

$$R(\mathbf{p})(\mathbf{p}'' - \mathbf{r}'') = \mathbf{0}. \tag{10}$$

Differentiating (9) again we obtain

$$D_t^2[\mathbf{p}(t)] = D_t[dh_{\mathbf{q}(t)}]D_t[\mathbf{q}(t)] + dh_{\mathbf{q}(t)}D_t^2[\mathbf{q}(t)].$$

Applying the chain rule to each entry of the matrix $dh_{\mathbf{q}(t)}$, we see that $D_t[dh_{\mathbf{q}(t)}]\big|_{t=0}$ depends only on $\mathbf{p}'$ and not on $\mathbf{q}''$. Let $\mathbf{s}''$ be the value of the second derivative of $\mathbf{p}(t)$ (that is $\mathbf{r}''$) when $\mathbf{q}'' = \mathbf{0}$, evaluated when $t = 0$. In other words

$$\mathbf{s} = D_t[dh_{\mathbf{q}(t)}]D_t[\mathbf{q}(t)]\big|_{t=0}$$

is independent of the choice of $\mathbf{q}''$. We must then solve the linear equation

$$\mathbf{p}'' = \mathbf{s}'' + dh_{\mathbf{p}}(\mathbf{q}'') = \mathbf{s}'' + \mathbf{q}''$$

for $\mathbf{q}''$. Recall that $dh_{\mathbf{p}}$ is the identity map by condition (b). By (10), $(\mathbf{p}', \mathbf{s}'')$ is a second-order flex of $G(\mathbf{p})$ and $\mathbf{p}'' - \mathbf{s}''$ is a first-order flex of $R(\mathbf{p})$ and thus is in $T_{\mathbf{p}}$. Thus we can define $\mathbf{q}'' = \mathbf{p}'' - \mathbf{s}''$. Then $\mathbf{p}(t)$ as defined is the desired flex. □

Remark 6.1.1. One way of looking at the above proof is to think of adding some curvature via $\mathbf{q}''$ to the curve $\mathbf{q}(t)$ to cancel the curvature in $M_{G(\mathbf{p})}$ in order to achieve the given second derivative $\mathbf{p}''$.

For our purposes we do not need $\mathbf{p}(t)$ to be real analytic. It only needs to be twice differentiable. In the spirit of Definition 2.1.2 (c) we stated things in this more general form.

It seems natural that there also should be a generalization of this result involving any number of derivatives.

6.2. Roth's conjecture. We can now prove Roth's conjecture in its full generality.

PROPOSITION 6.2.1. *A convex b-c polygon in the plane is rigid if and only if it is first-order rigid.*

Proof. Let $\mathbf{p}'$ be any nontrivial first-order flex of a *b-c* polygon $G(\mathbf{p})$. If $G(\mathbf{p})$ has no nonzero proper self stress, then by the first-order stress test, we can choose $\mathbf{p}'$ such that $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}'_i - \mathbf{p}'_j) < 0$ for all cables $\{i, j\}$. If ω is any proper nonzero self stress for $G(\mathbf{p})$, then, by [7], the associated stress matrix Ω is negative semidefinite with only the affine motions in the kernel. It is easy to check, due to the convex nature of the polygon and the cabling, that $(\mathbf{p}')^T\Omega\mathbf{p}' \neq 0$. Thus $(\mathbf{p}')^T\Omega\mathbf{p}' < 0$. Thus by the second-order stress test, Corollary 5.2.1, $\mathbf{p}'$ extends to a second-order flex $(\mathbf{p}', \mathbf{p}'')$ that is strict on all cables. But the subframework $G_0(\mathbf{p})$ of $G(\mathbf{p})$ consisting of just the bars is just a convex polygon and so is independent. Thus Proposition 6.1.1 applies to show that there is a flex $\mathbf{p}(t)$ of $G(\mathbf{p})$, such that $D_t[\mathbf{p}(t)]\big|_{t=0} = \mathbf{p}'$ and $D_t^2[\mathbf{p}(t)]\big|_{t=0} = \mathbf{p}''$. But since $(\mathbf{p}', \mathbf{p}'')$ is strict on all cables, $\mathbf{p}(t)$ is a nontrivial continuous flex of $G(\mathbf{p})$ as well. Thus $G(\mathbf{p})$ is not rigid, and the result is proved. □

COROLLARY 6.2.2. *If a convex b-c polygon in the plane with v vertices has less than $v - 2$ cables, then it is not rigid.*

Proof. At least $v-2$ cables are needed to make the tensegrity framework infinitesimally rigid. □

Remark 6.2.1. When one is attempting to show directly that a particular convex *b-c* polygon is not rigid in the plane, one might be tempted to force some of the stressed cables to be bars in order to decrease the "degrees of freedom" and simplify the calculation. For example, the framework $G(\mathbf{p})$ of Figure 19 is not rigid. It is

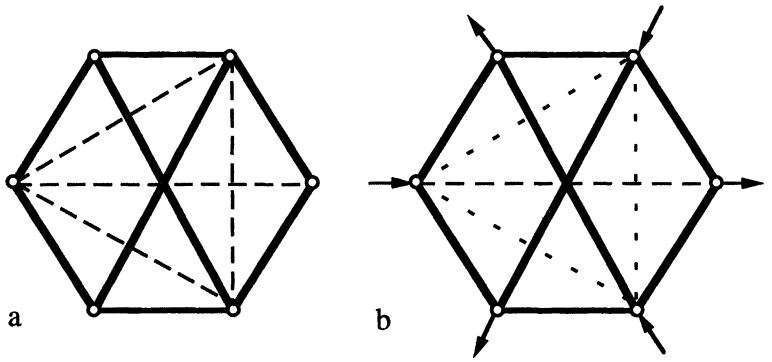


FIG. 19. *A flexible framework in the plane* (a), *with the corresponding first-order flex* (b).

obtained by forcing two of the cables of Figure 14a to be bars. (If the horizontal cable is changed to a strut (or bar), then the framework becomes rigid.)

The subframework $G_0(\mathbf{p})$, consisting of just the bars, is independent: $G(\mathbf{p})$ has a first-order flex, and from first-order considerations, as in [1], $G_0(\mathbf{p})$ has a continuous flex. The length of the horizontal cable cannot increase under this flex (by [7]), and the other cable lengths decrease strictly in their first derivative. Thus we obtain a continuous flex of $G(\mathbf{p})$.

However, one must be careful in deciding which of the cables to force to be bars. For example, consider the framework of Figure 20a.

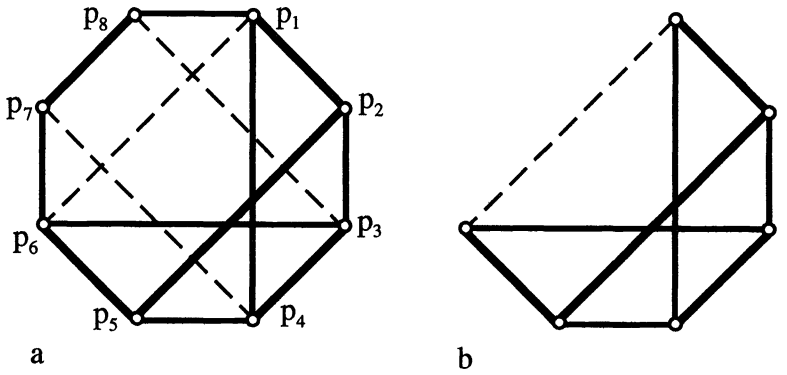


FIG. 20. *Framework* (a) *is rigid because subframework* (b) *is rigid.*

We have changed three of the stressed cables of Figure 3c to bars, and we consider only $\{1,6\}$, $\{4,7\}$, and $\{3,8\}$ from the rest of the cables of Figure 3c. There is a proper stress ω involving only members among the pairs of the first six vertices, since $\mathbf{p}_1,\ldots,\mathbf{p}_6$ lie on a circle. See Figure 20b for this *c-b* subframework, which is rigid by [7].

Considering $\mathbf{p}_1,\mathbf{p}_3,\mathbf{p}_4$, and $\mathbf{p}_6$ as a pinned rigid subset, then the vertices $\mathbf{p}_1$,$\mathbf{p}_3$,$\mathbf{p}_4$,$\mathbf{p}_6$,$\mathbf{p}_7$,$\mathbf{p}_8$ determine an infinitesimally rigid framework. So the whole framework $G(\mathbf{p})$ in Figure 20a is rigid.

On the other hand, in $G(\mathbf{p})$, the subframework determined by just the bars is independent, as with our proof of Roth's conjecture. Also, all the proper self stresses of Figures 3c are proper self stresses of $G(\mathbf{p})$ as well, and there is a nontrivial first-order flex $\mathbf{p}'$ of Figure 3c that is a nontrivial first-order flex of $G(\mathbf{p})$. Why can't we then conclude, as we did with our proof of Roth's conjecture, that $G(\mathbf{p})$ is flexible by

extending $\mathbf{p}'$ to s strict second-order flex $(\mathbf{p}', \mathbf{p}'')$ of $G(\mathbf{p})$ (using the strict second-order stress test, Corollary 5.2.1)? The reason we cannot apply the second-order stress test is because we must consider *all* proper self stresses of $G(\mathbf{p})$, not just those coming from Figure 3c. In fact, $G(\mathbf{p})$ has more self stresses to consider due to the added bars. When the added stresses *are* considered, it turns out that $G(\mathbf{p})$ is even prestress stable.

The moral of the story is that if one wants to add interior bars, as in Figure 19, and keep the framework flexible, then one must not only be careful that the added bars keep the bar subframework independent and do not destroy the infinitesimal flexes, but also be sure that the new bars do not introduce any new proper self stresses to the whole framework.

A. Appendix on replacement principles.

A.1. From bar frameworks to cables and struts. Recall from §2.3 that if a bar framework is infinitesimally rigid, with a nonzero self stress, we can replace some of the bars with cables and struts, following the signs of the self stress. (See [29].)

Similarly, from §3.4, if a bar framework is prestress stable with a nontrivial self stress ω, then we are able to replace some of the bars with cables or struts, following the signs of this self stress ω.

For a second-order rigid bar framework, we have no such replacement principle. If the framework is not prestress stable, we must check the signs of all stresses used to block the cone of first-order flexes. If these all agree on a specific sign, then the corresponding bar can be replaced, while preserving second-order rigidity.

For a framework which is rigid, by some other test, we know of no general replacement principle. In the following, we show how to replace cables and struts with bars.

A.2. Equivalent bar frameworks for prestress stability. Given a tensegrity framework with cables and struts, we can always replace all members with bars. This replacement will, of course, preserve any rigidity in the framework. In fact it may increase the rigidity, turning a nonrigid framework into a rigid framework, a second-order rigid framework into a prestress stable framework, or a prestress stable framework into a first-order rigid framework. We would like a more delicate replacement principle which leaves the rigidity, prestress stability, or second-order rigidity unchanged.

We associate a special bar framework with a tensegrity framework, which does not depend on fixing a self stress. Suppose some framework $G(\mathbf{p})$ has a cable $\{i, j\}$ with $\mathbf{p}_i \neq \mathbf{p}_j$. We can then replace the cable by two bars $\{i, k\}$ and $\{k, j\}$ and place $\mathbf{p}_k$ on the open line segment between $\mathbf{p}_i$ and $\mathbf{p}_j$ to get a framework as in Figure 21a.

Similarly, a strut $\{i, j\}$ can be replaced by two bars $\{i, k\}$ and $\{k, j\}$, but now we insist that $\mathbf{p}_k$ be on the line through $\mathbf{p}_i$ and $\mathbf{p}_j$ but outside the closed line segment between $\mathbf{p}_i$ and $\mathbf{p}_j$ as in Figure 21b.

We call the above processes *splitting a cable* and *splitting a strut*, respectively. It is clear that such splittings do not change the rigidity of a framework in $\mathbb{R}^d$ for $d \geq 2$, but any such splitting creates a framework that is not first-order rigid. In fact we can split all the cables and struts of $G(\mathbf{p})$ to create what we shall call an *equivalent bar framework* $\hat{G}(\hat{\mathbf{p}})$, where $\hat{G}(\hat{\mathbf{p}})$ is rigid if and only if $G(\mathbf{p})$ is rigid.

We next look at the relation between splitting members and prestress stability. Suppose ω is a proper self stress for the tensegrity framework $G(\mathbf{p})$, and $\{i, j\}$ is a

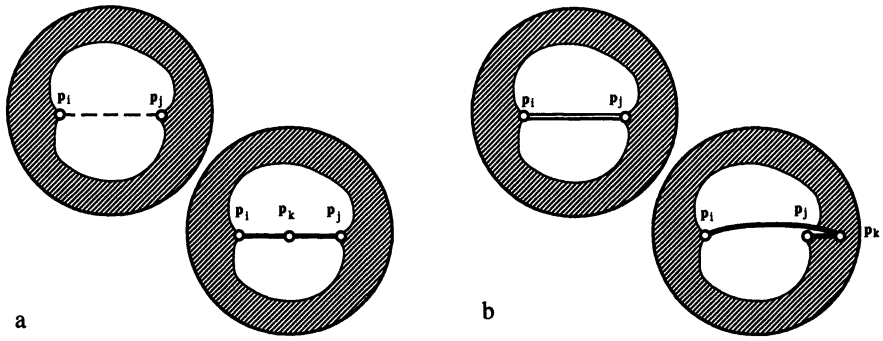


FIG. 21. *Replacing a cable* (a) *and a strut* (b) *with "equivalent" pairs of bars preserving prestress stability.*

cable or strut for G. Suppose $G(\mathbf{p})$ is split along $\{i,j\}$ at $\mathbf{p}_k$. Define

$$\hat{\omega}_{ik} = \omega_{ij}\frac{|\mathbf{p}_j - \mathbf{p}_i|}{|\mathbf{p}_i - \mathbf{p}_k|},$$

$$\hat{\omega}_{jk} = \left\{ \begin{array}{ll} \omega_{ij}\dfrac{|\mathbf{p}_j - \mathbf{p}_i|}{|\mathbf{p}_i - \mathbf{p}_k|} & \text{if} \quad \omega_{ij} > 0 \\ -\omega_{ij}\dfrac{|\mathbf{p}_j - \mathbf{p}_i|}{|\mathbf{p}_i - \mathbf{p}_k|} & \text{if} \quad \omega_{ij} < 0 \end{array} \right\},$$

and $\hat{\omega}_{\ell m} = \omega_{\ell m}$ for $\{\ell, m\} \neq \{i,k\}$ and $\{\ell, m\} \neq \{j,k\}$, where $\mathbf{p}_j$ is between $\mathbf{p}_i$ and $\mathbf{p}_k$ when $\omega_{ij} < 0$. It is easy to check that $\hat{\omega}$ defined above is a self stress for $\hat{G}(\hat{\mathbf{p}})$, the split framework.

PROPOSITION A.2.1. *Let $G(\mathbf{p})$ be any tensegrity framework with a proper strict self stress ω. Let $\hat{G}(\hat{\mathbf{p}})$ be the framework split along any cable or strut. Then ω stabilizes $G(\mathbf{p})$ if and only if $\hat{\omega}$ stabilizes $\hat{G}(\hat{\mathbf{p}})$.*

Proof. By Proposition 3.4.2 we need only consider a space of first-order flexes $\mathbf{p}'$ of $G(\mathbf{p})$ that are complementary to the trivial first-order flexes and then evaluate them on the form determined by Ω. A similar statement holds for $\hat{G}(\hat{\mathbf{p}})$. By the first-order stress test since $\omega_{ij} \neq 0$ we know that $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}'_i - \mathbf{p}'_j) = 0$. By adding a trivial first-order flex to $\mathbf{p}'$ we may consider some space of first-order flexes of $G(\mathbf{p})$ that has $\mathbf{p}'_i = \mathbf{p}'_j = 0$. Similarly for $\hat{G}(\hat{\mathbf{p}})$ we may consider only those first-order flexes $\hat{\mathbf{p}}'$ that are the direct sum of $\mathbf{p}'$ on the vertices of $G(\mathbf{p})$ and $\mathbf{p}'_k$ which is perpendicular to $\mathbf{p}_i - \mathbf{p}_j$, where k is the splitting vertex.

We evaluate $\hat{\Omega}$ at $\hat{\mathbf{p}}'$:

$$\begin{aligned} (\hat{\mathbf{p}}')^T\hat{\Omega}\hat{\mathbf{p}}' &= \sum_{\ell m} \hat{\omega}_{\ell,m}|\hat{\mathbf{p}}'_\ell - \hat{\mathbf{p}}'_m|^2, \\ &= \sum_{\ell m} |\mathbf{p}'_\ell - \mathbf{p}'_m|^2 + \omega_{ik}|\mathbf{p}'_k|^2 + \omega_{jk}|\mathbf{p}'_k|^2, \\ &= (\mathbf{p}')^T\Omega\mathbf{p}'. \end{aligned}$$

It is easy to check (even for struts) that $\omega_{ik} + \omega_{jk} > 0$. Thus $\hat{\Omega}$ is positive definite on its complementary space of nontrivial first-order flexes if and only if Ω is positive definite on its corresponding space. □

COROLLARY A.2.2. *Let $G(\mathbf{p})$ be any tensegrity framework and $\hat{G}(\hat{\mathbf{p}})$ the equivalent bar framework obtained by splitting all the cables and struts of nonzero length.*

Then $G(\mathbf{p})$ is prestress stable with a strict proper self stress if and only if $\hat{G}(\hat{\mathbf{p}})$ is prestress stable.

Thus, if we wish, we can "reduce" the problem of when a framework is prestress stable to the case when all the members are bars.

A.3. Equivalent bar frameworks for second-order rigidity. If we have a tensegrity framework with no strict proper self stress, such as in Figure 9b, then replacing this with the equivalent pair of bars can destroy second-order rigidity (but not rigidity). For example, a second-order flex is indicated on Figure 22.

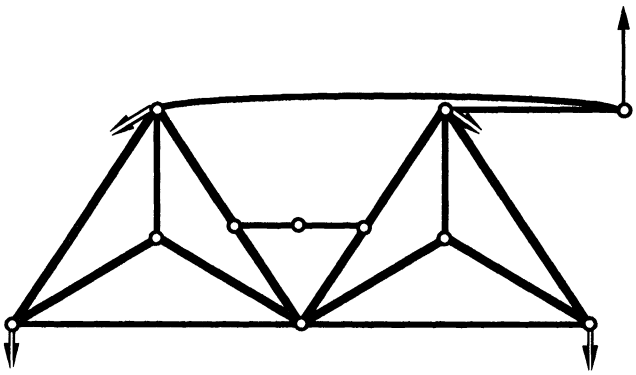


FIG. 22. *A second-order flexible (but rigid) bar framework equivalent to a second-order rigid tensegrity framework.*

If we restrict ourselves to tensegrity frameworks in which all cables and struts have nonzero coefficients in some self stress, then we can switch to the equivalent bar framework to check second-order rigidity. (Recall that for a tensegrity framework a proper self stress is strict if $\omega_{ij} \neq 0$ for every cable or strut.) We omit the proof, which is not difficult. See §5.2 and the second-order stress test.

PROPOSITION A.3.1. *A tensegrity framework with a strict proper self stress is second-order rigid if and only if the equivalent bar framework is second-order rigid.*

Acknowledgments. This work was provoked by the stimulating exchanges at the workshops on tensegrity frameworks and rigidity of triangulated surfaces held at the Université de Montréal during February 1987. We thank all of the participants, with particular thanks to Tibor Tarnai, Zsolt Gaspar, Tim Havel, and Ben Roth. We also thank Gerard Laman for several corrections to an earlier draft.

REFERENCES

[1] L. ASIMOW AND B. ROTH, *The rigidity of graphs*, Trans. Amer. Math. Soc., 245 (1978), pp. 279–289.

[2] ———, *The rigidity of graphs* II, J. Math. Anal. Appl., 68 (1979), pp. 171–190.

[3] E. BOLKER AND B. ROTH, *When is a bipartite graph a rigid framework?* Pacific J. Math., 90 (1980), pp. 27–44.

[4] C. R. CALLADINE, *Buckminister Fuller's "tensegrity" structures and Clerk Maxwell's rules for the construction of stiff frames*, Internat. J. Solids Structures, 14 (1978), pp. 161–172.

[5] C. R. CALLADINE AND S. PELLEGRINO, *First-order infinitesimal mechanisms*, Internat. J. Solids Structures, 27 (1991), pp. 505–515.

[6] R. CONNELLY, *The rigidity of certain cables frameworks and the second-order rigidity of arbitrarily triangulated convex surfaces*, Adv. Math., 37 (1980), pp. 272–299.

[7] ———, *Rigidity and energy*, Invent. Math., 66 (1982), pp. 11–33.

[8] R. CONNELLY, *Basic concepts of rigidity*, in The Geometry of Rigid Structures, H. Crapo and W. Whiteley, eds., manuscript.
[9] ———, *Basic concepts of infinitesimal rigidity*, in The Geometry of Rigid Structures, H. Crapo and W. Whiteley, eds., manuscript.
[10] ———, *Basic concepts of static rigidity*, in The Geometry of Rigid Structures, H. Crapo and W. Whiteley, eds., manuscript.
[11] ———, *Rigid circle and sphere packings Part* I: *Finite packings*, Structural Topology, 14 (1988), pp. 43–60.
[12] ———, *Rigid circle and sphere packings Part* II: *Infinite packings with finite motion*, Structural Topology, 16 (1991), pp. 57–76.
[13] R. CONNELLY AND W. WHITELEY, *The stability of tensegrity frameworks*, J. Space Structures, 7 (1992), pp. 153–163.
[14] H. CRAPO AND W. WHITELEY, *Stresses in frameworks and motions of panel structures: A projective geometry introduction*, Structural Topology, 6 (1982), pp. 42–82.
[15] J. FRANKLIN, *Methods of Mathematical Economics*, Springer-Verlag, New York, 1980.
[16] H. GLÜCK, *Almost all simply connected surfaces are rigid*, in Geometric Topology, Lecture Notes in Math 438, Springer-Verlag, Berlin, New York, 1975, pp. 225–239.
[17] B. GRÜNBAUM, *Convex Polytopes*, Interscience, New York, 1967.
[18] B. GRÜNBAUM AND G. C. SHEPARD, *Lectures on Lost Mathematics*, Mimeographed notes, University of Washington, Seattle, 1975.
[19] ———, *Lectures on Lost Mathematics*, Mimeographed notes, University of Washington, Seattle, (reissued for the special section on rigidity at the 760th meeting of the A.M.S.), 1978
[20] N. H. KUIPER, *Spheres polyhedriques flexible dans* E^3, *d'ápres Robert Connelly*, Seminaire Bourbaki, 541 (1978), pp. 1–22.
[21] E. KÖTTER, *Über die Möglichkeit, n Punkte in der Ebene oder im Raume durch weniger als* $2n-3$ *oder* $3n-6$ *Stäbe von ganz unveränderlicher Länge universchieblich miteinander zu verbinden*, Festschrift Heinrich Müller-Breslau, 1912, pp. 61–80.
[22] E. N. KUZNETSOV, *Underconstrained structural systems*, Internat. J. Solids Structures, 24 (1988), pp. 153–163.
[23] ———, *On immobile kinematic chains and a fallacious matrix analysis*, J. Appl. Mech., Brief Notes, 56 (1989), pp. 222–224.
[24] ———, *Letter to the editor*, Internat. J. Solids Structures, 27 (1991), pp. 517–519.
[25] ———, *Systems with infinitesimal mobility: Part* I–*Matrix analysis and first-order mobility*, J. Appl. Mech., 58 (1991), pp. 513–519.
[26] ———, *Systems with infinitesimal mobility: Part* II–*Compound and higher-order infinitesimal mechanisms*, J. Appl. Mech., 58 (1991), pp. 520–526.
[27] ———, *Underconstrained Structural Systems*, Springer-Verlag, Berlin, New York, 1991.
[28] S. PELLEGRINO AND C. R. CALLADINE, *Matrix analysis of statically and kinematically indeterminate frameworks*, Internat. J. Solids Structures, 32 (1986), pp. 409–428.
[29] B. ROTH AND W. WHITELEY, *Tensegrity frameworks*, Trans. Amer. Math. Soc., 265 (1981), 419–446.
[30] J. SZABO AND L. KOLLÁR, *Structural Design of Cable-Suspended Roofs*, Akademiai Kiado, Budapest, 1984.
[31] T. TARNAI, *Problems concerning spherical polyhedra and structural rigidity*, Structural Topology, 4 (1980), pp. 61–66.
[32] A. W. TUCKER, *Dual systems of homogeneous linear relations*, in Linear Inequalities and Related Systems , H. W. Kuhn and A. W. Tucker, eds., Annals of Math Studies 38, 1956, pp. 3–18.
[33] N. WHITE AND W. WHITELEY, *The algebraic geometry of stresses in frameworks*, SIAM J. Alg. Disc. Meth., 8 (1983), pp. 1–32.
[34] W. WHITELEY, *Infinitesimal motions of a bipartite framework*, Pacific J. Math., 110 (1984), pp. 233–255.
[35] ———, *Tensegrity frameworks*, in The Geometry of Rigid Structures, H. Crapo and W. Whiteley, eds., manuscript.
[36] ———, *Second-order rigidity and global rigidity*, in The Geometry of Rigid Structures, H. Crapo and W. Whiteley, eds., manuscript.

SIAM J. DISCRETE MATH.
Vol. 9, No. 3, pp. 492–509, August 1996

009

APPROXIMATION ALGORITHMS FOR THE k-CLIQUE COVERING PROBLEM*

OLIVER GOLDSCHMIDT†, DORIT S. HOCHBAUM‡, COR HURKENS§, AND GANG YU¶

Abstract. The problem of covering edges and vertices in a graph (or in a hypergraph) was motivated by a problem arising in the context of the component assembly problem. The problem is as follows: given a graph and a clique size k, find the minimum number of k-cliques such that all edges and vertices of the graph are covered by (included in) the cliques. This paper provides a collection of approximation algorithms for various clique sizes with proven worst-case bounds. The problem has a natural extension to hypergraphs, for which we consider one particular class. The k-clique covering problem can be formulated as a set covering problem. It is shown that the algorithms we design, which exploit the structure of this special set covering problem, have better performance than those derived from direct applications of general purpose algorithms for the set covering. In particular, these special classes of set covering problems can be solved with better worst-case bounds and/or complexity than if treated as general set covering problems.

Key words. approximation algorithms, clique covering, set covering, worst-case analysis

AMS subject classifications. 05C65, 05C85, 05C90

1. Introduction.

The problem of *k-clique covering* (CC_k) is defined on a graph (or a hypergraph) and a given clique size k. The aim is to use the least number of cliques—or subsets of k vertices—so that each vertex and each edge is contained in at least one such clique. When all edges are covered then all vertices incident to these edges are also covered, so the issue of covering vertices in addition to edges is of interest only in graphs with isolated vertices.

The k-clique covering problem's objective value differs from the *clique covering number* of a graph (frequently used in literature related to perfect graphs) in that the clique covering number of a graph is a partitioning of the edges of the graph into a minimum number of complete subgraphs. Hence, for each clique used in the clique covering number problem, all edges of the clique are present in the graph. As such, each edge must belong to exactly one clique in the cover. In our problem, each clique covers all edges contained in it. Contrary to the clique covering number, not every edge of the clique must be present in the graph. Furthermore, an edge may be covered by more than one clique.

One problem addressed in the literature that is related to the 3-clique covering problem is the partitioning of a graph into triangles. The feasibility decision problem—whether the edges of a graph can be partitioned into triangles—was proved NP-complete by Holyer [12]. Therefore, finding the minimum number of triangles to cover the edges of a graph is NP-hard.

A general definition of the (CC_k) problem for hypergraphs is as follows. A hypergraph $H = (V, F)$ is defined by a vertex set $V = \{1, \ldots, n\}$ and a hyperedge set $F \subseteq 2^V$. A clique of size k in a hypergraph H is a subset of vertices $K \subseteq V$ such

* Received by the editors July 19, 1993; accepted for publication (in revised form) October 10, 1995. This research was supported in part by Office of Naval Research grant N00014-91-J-1241.

† Department of Mechanical Engineering, University of Texas at Austin, Austin, TX 78712-1063.

‡ Department of Industrial Engineering and Operations Research, University of California at Berkeley, Berkeley, CA 94720.

§ Department of Mathematics and Computing Science, Eindhoven University of Technology, the Netherlands.

¶ Department of Management Science and Information Systems, University of Texas at Austin. Austin, TX 78712-1175.

that the hyperedges representing all subsets of K exist in H and $|K| = k$. In the case $|f| = 2$, for all $f \in F$, each hyperedge is an edge containing a pair of vertices, and the hypergraph is a graph. The k-clique covering (CC_k) problem is to find the minimum number of cliques of size no more than k such that all hyperedges of H are covered by (included in) the cliques.

The k-clique covering problem can be viewed as a special case of the set covering problem. In this context, approximation algorithms that apply to the set covering problem are also applicable to the k-clique covering problem. The drawback of using the set covering problem is that the input to the set covering problem includes all possible subsets of size k and hence is exponential in k. We propose an approach that exploits the special structure of the clique covering problem and delivers better worst-case bounds than the ones using the set covering.

The main results in this paper are approximation algorithms for the problem (CC_k) on graphs and on a class of hypergraphs. We present approximation algorithms based on the formulation of the problem as a set covering problem and algorithms specifically derived for the clique cover problem. The latter algorithms have better worst-case performance. Consequently, these classes of set covering problems can be solved with better worst-case bounds and/or complexity than if treated as general set covering problems.

The (CC_k) problem has many practical applications in flexible manufacturing systems and component assembly in the semiconductor industry. In [9], Goldschmidt, Hochbaum, and Yu provide a detailed description of some related applications.

In related literature, Tang and Denardo [22] developed a branch-and-bound procedure for solving the (CC_k) problem. The lower bound is generated by a so-called sweeping procedure and it can be arbitrarily bad. Several heuristic procedures have been proposed and tested for generating feasible solutions. These heuristics are similar to bin packing heuristics in that they select a starting seed hyperedge for a new bin (clique) and sequentially fill it according to some precedence rule. Tang and Denardo [22] and Whitney and Gaul [23] propose different rules for selecting the next hyperedge to add to the current clique. In these papers, no analysis or worst-case bounds have been provided for the heuristic solutions.

Let z^H and z^* be, respectively, the number of cliques derived by an approximation algorithm and the minimum number of cliques of a (CC_k) instance. An algorithm is a δ-approximation algorithm if, for any family of instances of the problem, $z^H \leq \delta z^* + o(z^*)$. We call δ the *worst-case bound* of the algorithm, and the algorithm a *δ-approximation algorithm.*

Our results include approximation algorithms for the following cases. For the triangle covering problem, the 3-clique covering, we present two approximation algorithms with worst-case bounds of 1.4 and 1.5, respectively. For the 4-clique covering our algorithm is a $2\frac{1}{3}$-approximation algorithm. For the general k-clique covering we describe an algorithm that is a $(\frac{k}{2} + \frac{k}{k-1})$-approximation algorithm. For this case the *greedy heuristic* for set covering gives a bound of $\mathcal{H}(\binom{k}{2}) \approx 2 \log k$ with running time $O(n^k)$. $\mathcal{H}(d)$ is a harmonic series defined as $\mathcal{H}(d) = \sum_{j=1}^{d} \frac{1}{j}$. The harmonic series is asymptotically equal to the natural logarithm of d (plus the Euler constant). For the problem on hypergraphs, when each hyperedge to be covered is of size $k-1$, we describe two approximation algorithms and demonstrate that they compare favorably with algorithms derived from setting the problem as a set covering problem.

In §2, we discuss the reduction of the k-clique covering problem to the set covering problem. Once reduced to the set covering problem, the greedy heuristic [4] for set

TABLE 1
δ-approximation algorithms.

Algorithm	Case	Running time	δ
H_1	$(3, \leq 2)$	$O(m^{2.5})$	1.4
H_2		$O(m)$	1.5
H_3	$(4, \leq 2)$	$O(m)$	7/3
H_4	$(k, \leq 2)$	$O(m)$	$\frac{k}{2} + \frac{k}{k-1}$
H_5	$(k, k-1)$	$O(m^3 k)$	$\mathcal{H}(k) - \frac{1}{6}$
H_6		$O(m^{2.5})$	$\lceil \frac{k}{2} \rceil$

covering becomes applicable. Throughout this paper, we refer to the greedy heuristic as the *greedy*. For a set covering problem on a hypergraph with n vertices and with m elements to be covered, elements that are edges and vertices in a graph or hyperedges in a hypergraph, the *greedy* is a $\min\{k \log 2, \log m\}$-approximation algorithm of complexity $O(\binom{n}{k} \min(k!, m))$.

Section 3.1 includes the approximation algorithms for graphs with clique sizes 3 or 4. Section 3.2 contains two approximation algorithms for solving the problem on hypergraphs in which each hyperedge has exactly $(k-1)$ vertices.

Table 1 summarizes the results of §3. Case (k, q) denotes an instance for which each hyperedge has cardinality q, and case $(k, \leq q)$, an instance for which a hyperedge consists of at most q vertices.

In §4 we conclude with some future research directions.

2. Reduction to the set covering problem. The *set covering* problem (SC) is defined for a universal set of m elements, $I = \{1 \ldots m\}$, and a collection of p sets $S_i \subset I$, $i = 1 \ldots p$. The problem is to find a smallest cardinality collection of sets, the union of which is I.

Not only is the set covering problem NP-hard, but it was recently established that an approximation algorithm for the problem with better than logarithmic bound is impossible unless all NP problems are solvable in subexponential time [18]—a fairly unlikely prospect.

The set covering problem may be generalized to a problem of covering sets with sets, as opposed to covering elements with sets. A set S is said to be covered by a set $\bar{S}$ if $S \subseteq \bar{S}$. The (CC_k) is a class of instances of covering sets with sets when the covering sets S_i are cliques of size k and the elements are vertices and edges (hyperedges) of a graph (hypergraph). As such it is a special case of the set covering problem with all sets of fixed size k. As noted before, this set covering problem is NP-hard since covering edges with a minimum number of triangles is a special case.

The (CC_k) has a shorter input representation than the set covering. Let the k-clique covering problem be defined on a hypergraph with n vertices. Since any subset of k vertices must be considered as a potential clique, $p = \binom{n}{k}$ sets are listed as part of the input. On the other hand, for the input of (CC_k) only the size limit, k, is specified in addition to the graph (or hypergraph).

In order to reduce (CC_k) to (SC), we consider the union of the hyperedges (that are sets of vertices) as the universal set. Any subset of vertices of size k can be potentially used in a cover. So all such subsets are enumerated, and for each one we list all the hyperedges and vertices that are contained in such a subset. This creates a set covering problem with the number of sets p equal to $\binom{n}{k}$, where n is the number

of vertices in the graph, and m is the number of elements equal to the number of hyperedges and isolated vertices.

There are two known approximation algorithms available for (SC). One bounds the worst-case error by the maximum number of sets that an element belongs to [10]. The other is the *greedy* with worst-case error bound equal to the harmonic series $\mathcal{H}(d)$, where d is the largest set size [16, 17, 4]. The running time of the *greedy* is $O(n \log n)$, where n is the number of sets. With the recent result of Lund and Yanakakis [18], the *greedy* is, up to a constant factor, a best possible approximation algorithm for the set covering problem, as it cannot be improved unless all NP problems are in DTIME$[n^{\text{poly }(\log n)}]$. The *greedy* is also ideally suited to the input of (CC_k), as the largest set size is the maximum number of hyperedges covered by a k-clique, 2^k. Consequently, this reduction makes an $\mathcal{H}(2^k)$-approximation algorithm readily available. There are $\binom{n}{k}$ potential covering sets. It takes $O(mk)$ to identify the hyperedges covered by a k-clique. At each iteration, we scan the list of cliques for the one that covers the most hyperedges. At each iteration, at least one hyperedge gets covered and thus removed. So the *greedy* iterates at most m times. Therefore, the running time of the *greedy* on an instance of (CC_k) is $O(\binom{n}{k}m^2k)$.

In a companion paper [9], we demonstrated an improvement to the *greedy* called the *modified greedy* heuristic, or simply the *modified greedy*. The *modified greedy* is an $\mathcal{H}(d) - \frac{1}{6}$-approximation algorithm. Its running time is $O(n \log n + m^{2.5})$ on a general set covering problem with n sets and m elements, and it follows that its running time on an instance of (CC_k) is $O(\binom{n}{k}m + m^{2.5})$. This improvement is achieved by solving, in addition to the greedy procedure, a matching problem in a graph. The *modified greedy* is useful for instances of set covering with largest set size being bounded by a small value, as in the clique covering problem.

The other approximation algorithms described here have either faster running times or better bounds, or both, for various special cases.

3. Approximation algorithms. In this section, we present approximation algorithms for the k-clique covering problem. We first examine the (CC_k) problem on graphs for the cases $k = 3$ and $k = 4$, and then for arbitrary k. We then describe two approximation algorithms for covering a hypergraph with hyperedges all of size $(k-1)$ with k-cliques.

For the clique covering problem on graphs, all vertices that are not isolated get covered when all edges are covered. Only isolated vertices may require additional cliques to be covered. For simplicity's sake, we are going to present the algorithms for covering edges only and comment separately about the adjustments required to cover isolated vertices as well. In all cases this adjustment does not modify the worst-case bound.

We use the common notation for a graph $G = (V, E)$, $|V| = n$ and $|E| = m$. In case G has isolated vertices, m denotes the number of edges and isolated vertices.

3.1. Covering graphs with k-cliques. The 2-clique covering problem on a graph is trivially solvable: each edge is covered by a separate clique and the isolated vertices are covered two per clique. For $k = 3$, the problem is NP-hard. To see this, notice that the number of triangles required to cover the edges of the graph is at least $\lceil \frac{m}{3} \rceil$. When m is a multiple of 3, this is achievable only if the edges of G can be partitioned into triangles. Recognizing whether a graph has a partition into triangles was shown to be NP-complete by Holyer [12].

The rest of this section is organized as follows. We first analyze the cases $k = 3$ and $k = 4$. For the case $k = 3$, we provide two algorithms, a 1.4-approximation algo-

rithm called (H_1) and a 1.5-approximation algorithm called (H_2), with running times $O(m^{3/2})$ and $O(m)$, respectively. While (H_1) has a better worst-case bound, (H_2) has a faster running time. These are the first known approximation algorithms for covering a graph with triangles. For the case of $k = 4$, we present a $\frac{7}{3}$-approximation algorithm, (H_3), with running time $O(m)$. In the last subsection, we describe approximation algorithms for arbitrary k.

3.1.1. The triangle covering problem. Both algorithms presented here are adaptations of the greedy approach.

Algorithm (H_1) runs in two phases. In the first phase, a maximal number of edge-disjoint triangles of G are covered. To find a maximal number of edge-disjoint triangles, we use the procedure by Itai and Rodeh [15] for determining whether a graph contains a triangle. The complexity of this procedure is $O(m^{3/2})$. We adapt it to find a maximal number of edge-disjoint triangles without increasing the complexity. The edges of these edge-disjoint triangles are covered and deleted from the graph.

In phase 2 there is no remaining triangle in the graph. It is possible to solve the 3-clique covering problem in a triangle-free graph in linear time. In a graph that contains no triangles, each covering triangle covers either a 2-chain (two adjacent edges) or a single edge. It is known [19] that if the number of edges m of a connected graph is even, then one can cover the edges with exactly $\frac{m}{2}$ 2-chains. Masuyama and Ibaraki [19] provide a linear time algorithm for solving this problem optimally. If a connected component of the triangle-free graph has an odd number of edges, then exactly one triangle is necessary to cover a single edge of the component while all others cover 2-chains. We call a triangle covering a single edge a 1-*triangle*.

Note that if the input graph G does not contain triangles or if the triangles are edge disjoint, then the problem can be solved optimally in $O(m^{3/2})$ time.

The algorithm is given below.

ALGORITHM (H_1).

Input: Graph $G = (V, E)$.

Phase 1: Find a maximal collection of edge-disjoint triangles T. Set $T^H = T$. Let E_T be the edges of T, $E \leftarrow E \setminus E_T$.

Phase 2: While $G = (V, E)$ contains a nontrivial component $G_i = (V_i, E_i)$, **do**

Cover E_i with a set T_i of $\left\lfloor \frac{|E_i|}{2} \right\rfloor$ 2-chains and possibly a 1-triangle (if $|E_i|$ is odd). Set $T^H \leftarrow T^H \cup T_i$ and $E \leftarrow E \setminus E_i$.

end

Output T^H.

End of (H_1)**.**

In case graph G contains a set of isolated vertices to be covered, the following adjustment is applied to (H_1). At the termination of phase 2, we assign isolated vertices, one for each 1-triangle. In phase 3, the remaining isolated vertices are covered three per triangle, with possibly one or two vertices in the last triangle used. T^H includes these additional triangles covering the isolated vertices.

LEMMA 1. *The complexity of algorithm* (H_1) *is* $O(m^{3/2})$.

Proof. Phase 1 of the algorithm requires a maximal collection of edge-disjoint triangles. The algorithm of Itai and Rodeh finds a triangle, if one exists, by constructing a breadth-first-search tree and inspecting the nontree edges. That procedure runs in

$O(m^{3/2})$. We use this procedure, and whenever a triangle is identified, its edges are removed from the graph and the data structure is updated in constant time. The procedure is then continued with no backtracking required. It is therefore possible to find a maximal collection of triangles with the same complexity as for finding a single triangle.

In phase 2, we find a maximum packing of 2-chains in a triangle-free graph G. This can be done in linear time by using the following procedure, which is an adaptation of the one by Masuyama and Ibaraki [19]. The procedure is presented for one connected component of G, $G_i = (V_i, E_i)$. Consider any spanning tree in that component. All nontree edges are appended to the tree, each with a new vertex assigned to one of its endpoints. This creates a tree on $|E_i|$ edges and $|E_i|+1$ nodes. The tree is suspended from any node, say 1, called the root node. While the tree contains more than one edge, consider any pair of leaf nodes that share the same parent node and cover the two edges incident to these leaf nodes with a 2-chain. Remove these two edges from the tree. If no such pair exists, then there is a node of degree 2 adjacent to a leaf node. The two edges incident to that node are packed in a 2-chain and removed from the tree. This operation is repeated until the tree contains at most one edge. The collection of 2-chains thus created forms a maximum packing.

If a single edge remains in the tree it is covered in phase 2 by a 1-triangle. The complexity of identifying a maximum packing is linear, as can be shown by a straightforward inductive argument. The total number of triangles required to cover each component G_i is $\lceil |E_i|/2 \rceil$.

The complexity of algorithm (H_1) is therefore dominated by phase 1 and is $O(m^{3/2})$, as stated. □

We claim that algorithm (H_1) is a $\frac{7}{5}$-approximation algorithm.

THEOREM 1. *The number of triangles delivered by* (H_1), $|T^H|$, *is at most* $\frac{7}{5}$ *times the smallest number of triangles covering the graph.*

Proof. Let z^H denote the number of triangles found by the heuristic. Let T^* denote a triangle-covering of G of minimum cardinality z^*. Note that in our context, a triangle is defined as a graph $\Delta = (V_\Delta, E_\Delta)$, with at most three vertices and at most three edges. A triangle with three edges is called a *proper* triangle. Without loss of generality we may assume that for each edge $e \in E$ there is exactly one triangle $\Delta \in T^*$ for which $e \in E_\Delta$.

Let the graph remaining at the end of phase 1 be $G' = (V', E')$. Let s denote the number of components in G' with an odd number of edges. Let W denote the set of isolated vertices in the original graph G.

Case 1. Assume that $s \leq |W|$.

Then $z^H = \frac{|E|-|E'|}{3} + \frac{|E'|+s}{2} + \left\lceil \frac{|W|-s}{3} \right\rceil$, where, for $i = 1, 2, 3$, the ith term results from phase i of the heuristic. Obviously we have

$$z^* \geq \left\lceil \frac{|E| + |W|}{3} \right\rceil,$$

as any triangle in T^* contains at most three edges from E, at most three vertices from W, or one edge from E and one vertex from W. Similarly,

$$z^* \geq \frac{|E'| + s}{2} + \left\lceil \frac{|W| - s}{3} \right\rceil,$$

as any triangle contains at most two edges from E', one edge from E', and one vertex from W, or at most three vertices from W. At least s triangles contain one edge from

E'. Combining the bounds we find

$$\begin{aligned} z^H &= \frac{|E|-|E'|}{3} + \frac{|E'|+s}{2} + \left\lceil \frac{|W|-s}{3} \right\rceil \\ &= \left\lceil \frac{|E|+|W|}{3} \right\rceil + \frac{|E'|+s}{6} + \frac{1}{3}\left\lceil \frac{|W|-s}{3} \right\rceil \\ &\quad + \left(\frac{|E|+|W|}{3} - \left\lceil \frac{|E|+|W|}{3} \right\rceil \right) + \left(\frac{2}{3}\left\lceil \frac{|W|-s}{3} \right\rceil - \frac{|W|-s}{3} \right) \\ &\leq \frac{4z^*+1}{3}. \end{aligned}$$

Note that equality holds only in the special case when $|W|-s=1$, $|E|$ and $|W|$ are multiples of 3, and both bounds on z^* are binding. For $z^* \leq 4$ it is easily verified that $z^H \leq \frac{4}{3}z^*$. For $z^* \geq 5$ we have $z^H \leq \frac{4z^*+1}{3} \leq \frac{7}{5}z^*$.

Case 2. Assume that $s > |W|$.

Then $6z^H = 2|E| + |E'| + 3s$.

In order to give an upper bound on this expression we derive three inequalities. Let s_i denote the number of components in E' with exactly i edges. Then $s - s_1 - s_3 - s_5 - s_7$ is the number of components with an odd number of at least nine edges. Hence $9(s - s_1 - s_3 - s_5 - s_7) \leq |E'| - s_1 - 3s_3 - 5s_5 - 7s_7$, which is rearranged to

$$9s - 8s_1 - 6s_3 - 4s_5 - 2s_7 - |E'| \leq 0. \tag{1}$$

The following two inequalities represent different lower bounds on the minimum number of triangles in a triangle cover. The first one is by counting degrees. A vertex v of degree $\delta(v)$ must occur in at least $\lceil \frac{1}{2}\delta(v) \rceil$ triangles. Define $\bar{V} \subseteq V$ to be the set of vertices v of even degree for which there exists at least one triangle $\Delta \in T^*$ such that $v \in V_\Delta$ and $\#(e \in E_\Delta, e \ni v) \leq 1$. Such a vertex $v \in \bar{V}$ appears in at least $\frac{1}{2}\delta(v)+1$ triangles. Note that $W \subseteq \bar{V}$. These considerations lead to the following:

$$\begin{aligned} 6z^* &\geq \sum_{\Delta \in T^*} \sum_{v \in V_\Delta} 2 = \sum_{v \in V} \sum_{\Delta \in T^*,\ V_\Delta \ni v} 2 \\ &\geq 2|W| + \sum_{v \in V,\ \delta(v)\text{odd}} (\delta(v)+1) + \sum_{v \in V \setminus \bar{V},\ \delta(v)\text{even}} \delta(v) + \sum_{v \in \bar{V} \setminus W} (\delta(v)+2). \end{aligned}$$

Here the first inequality uses the fact that a triangle contains at most three vertices, and the second one exploits the degrees of vertices. The result is reformulated as

$$6z^* \geq 2|W| + 2|E| + \sum_{v\ :\ \delta(v)\text{odd}} 1 + \sum_{v \in \bar{V} \setminus W} 2. \tag{2}$$

In phase 1 of the heuristic the parity of the degrees of the vertices does not change as only proper triangles are deleted from the graph. So odd degree vertices in G are odd-degree vertices in G'. Note that in G' each component of size 1 or 3 contains at least two vertices of odd degree. Hence these components give a total contribution of $2s_1 + 2s_3$ in the third term of inequality (2). A component of size 5 or 7 that contains an odd-degree vertex must contain at least two of these, and so it also gives a contribution of 2 in the third term of (2). If a component has only vertices of even degree, and at least one of these is in $\bar{E}$, then such a component contributes at least 2 in the fourth term in (2). It follows that components of size 5 or 7 that do not

contribute 2 to (2) can only have vertices of even degree, and none of these vertices can belong to $\bar{V}$.

The second lower bound on z^* is derived by counting triangles in T^* around each component in G'. Let, for $i = 0, 1, 2$, T_i^* denote the set of triangles in T^* that cover i edges of E'. For a component C of G', with vertex set $V(C)$ and edge set $E(C)$, and for $i = 1, 2$, let $T_i^*(C)$ denote the set of triangles $\Delta \in T^*$, with $|E(C) \cap E_\Delta| = i$. Note that $|E(C)| = |T_1^*(C)| + 2|T_2^*(C)|$. Furthermore, note that a triangle in T^* can share edges with at most one component of G', as these components are pairwise vertex disjoint.

A triangle $\Delta \in T_0^*$ with $V_\Delta = \{a, b, c\}$ is assigned to components C_a, C_b, C_c, for one third each, where C_x denotes the component containing x.

A triangle $\Delta \in T_2^*$ is assigned completely to the component it shares two edges with.

A triangle $\Delta \in T_1^*$ sharing one edge with component C is completely assigned to C, if $|T_1^*(C)| = 1$. On the other hand, if $|T_1^*(C)| > 1$, then Δ is assigned to C for a fraction of $\frac{29}{36}$, and to C' for the remaining $\frac{7}{36}$, where C' is the component containing the third vertex of Δ. (Note that C' may be equal to C, and it is also possible that C' contains an isolated vertex, or that C' does not exist. In the latter cases, the fraction $\frac{7}{36}$ is considered lost.)

Let $z^*(C)$ denote the total number of triangles assigned to C. Then $z^* \geq \sum z^*(C)$, where the sum is taken over all components C in G'. We estimate the value of $z^*(C)$ by distinguishing between the following cases.

[A] If $|T_1^*(C)| = 0$, then $|E(C)|$ is even and $z^*(C) \geq |T_2^*(C)| = \frac{1}{2}|E(C)|$.

[B] If $|T_1^*(C)| = 1$, then $|E(C)|$ is odd and $z^*(C) \geq 1 + |T_2^*(C)| = \frac{1}{2}(|E(C)| + 1)$.

[C] If $|T_1^*(C)| > 1$, and $|E(C)|$ is even, then $z^*(C) \geq |T_2^*(C)| + \frac{29}{36}|T_1^*(C)| = \frac{1}{2}(|E(C)| - |T_1^*(C)|) + \frac{29}{36}|T_1^*(C)| > \frac{1}{2}|E(C)|$.

[D] If $|T_1^*(C)| > 1$, and $|E(C)|$ is odd, then $z^*(C) \geq |T_2^*(C)| + \frac{29}{36}|T_1^*(C)| = \frac{1}{2}(|E(C)| - |T_1^*(C)|) + \frac{29}{36}|T_1^*(C)| > \frac{1}{2}(|E(C)| + 1) + \frac{1}{3}$, as $|T_1^*(C)| \geq 3$.

Let $D5$ and $D7$ denote the set of components C of size 5 and 7, respectively, for which the premise of [D] applies, that is, for which $|T_1^*(C)| > 1$ and $|E(C)|$ is odd.

Let $B5$ denote the set of components C of size 5, for which $|T_1^*(C)| = 1$, $V(C) \cap \bar{V} = \emptyset$, and $\delta(v)$ is even, for all $v \in V(C)$. Such a component C forms a 5-cycle, and each vertex in C has degree two in each triangle of T^* in which it appears. Moreover, $E(T_2^*(C)) - E(C) = \{a, b\}$ and $E(T_1^*(C)) - E(C) = \{c, d\}$ with 4 distinct edges a, b, c, d which form a 4-cycle. Let T_a denote the proper triangle containing edge a found by the heuristic in phase 1. Let the other two edges in T_a be $\{a', a''\}$. By symmetry between a and b we may assume without loss of generality that a' and a'' are not in $E(T^*(C))$. If a' or a'' is in $E(T_0^*)$, then we have $z^*(C) \geq 3 + \frac{1}{3}$. If not, then they must both belong to $E(T_1^*(C'))$ for some other component C'. But then $|T_1^*(C')| > 1$, and so C has been assigned at least $2 \times \frac{7}{36} > \frac{1}{3}$ of a triangle. Again we find $z^*(C) \geq 3 + \frac{1}{3}$. We conclude that $z^*(C) \geq 3 + \frac{1}{3} = \frac{1}{2}(|E(C)| + 1) + \frac{1}{3}$, for each $C \in B5 \cup D5$.

Let $B7$ denote the set of components C of size 7, for which $|T_1^*(C)| = 1$, $V(C) \cap \bar{V} = \emptyset$, and $\delta(v)$ is even, for all $v \in V(C)$. Such a component C must be a 7-cycle. Let a be any edge from $E(T_2^*(C)) \setminus E(C)$, and let T_a denote the proper triangle containing edge a found by the heuristic in phase 1. Let the other two edges in T_a be $\{a', a''\}$. If a' and a'' have both ends in $V(C)$, then one of them is in $E(T_0^*)$. If one of a', a'' is in $E(T_0^*)$, then C has been assigned at least $\frac{1}{3}$ of a T_0^*-triangle, so $z^*(C) \geq 4 + \frac{1}{3}$. It is easily verified that if the above does not apply, then $E(T_2^*(C)) \setminus E(C)$ contains

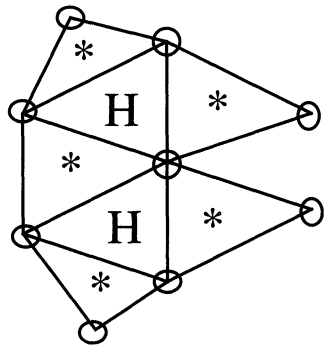


FIG. 1. *Illustration of the worst-case bound for Algorithm* (H_1).

an edge a, for which both edges a' and a'' are not in $E(T_0^*) \cup E(T^*(C))$. Hence they both belong to $E(T_1^*(C'))$ for some other component C'. We proceed as in the case for $B5$ and find that $z^*(C) \geq 4 + \frac{1}{3} = \frac{1}{2}(|E(C)| + 1) + \frac{1}{3}$, for $C \in B7 \cup D7$.

Combining these results we find

$$2z^* \geq \sum_C 2z^*(C) \geq |E'| + s + \sum_{C \in B5 \cup D5} \frac{2}{3} + \sum_{C \in B7 \cup D7} \frac{2}{3}. \tag{3}$$

Taking a linear combination of the three inequalities, with weight $\frac{1}{5}$ for inequality (1), weight 1 for inequality (2), and weight $\frac{6}{5}$ for inequality (3), we find

$$\begin{aligned} 6z^H &= 2|E| + |E'| + 3s \\ &\leq 0.2 \times (9s - 8s_1 - 6s_3 - 4s_5 - 2s_7 - |E'|) \\ &\quad + 1.0 \times \left(2|W| + 2|E| + \sum_{v:\delta(v) \text{ odd}} 1 + \sum_{v \in \bar{V} \setminus W} 2 \right) \\ &\quad + 1.2 \times \left(|E'| + s + \sum_{C \in B5 \cup D5} \frac{2}{3} + \sum_{C \in B7 \cup D7} \frac{2}{3} \right) \\ &\leq 0.2 \times 0 + 1.0 \times 6z^* + 1.2 \times 2z^* = 8.4z^*, \end{aligned}$$

which concludes our proof. □

Figure 1 illustrates the tightness of the bound of Algorithm (H_1): stars within triangles indicate an optimal covering, H's within triangles indicate the triangles selected in phase 1. Five more triangles are needed in phase 2 to cover the peripheral edges.

We now propose an alternative algorithm, (H_2), that runs in linear time and delivers a solution that is at most 1.5 times the optimum. (H_2) should be selected when running time is an important consideration.

ALGORITHM (H_2).

Input: Graph $G = (V, E)$. $T^H = \emptyset$.

Phase 1: For each nontrivial component $G_i = (V_i, E_i)$ of $G = (V, E)$ with $|E_i| \equiv 3$ modulo 6, **do**

Select a vertex $v \in V_i$, and find a vertex $w \in V_i$ at maximum distance from v. If there is a proper triangle Δ containing w, set $T^H \leftarrow T^H \cup \Delta$ and $E \leftarrow E \setminus E_\Delta$.

end

Phase 2: While $G = (V, E)$ contains a nontrivial component $G_i = (V_i, E_i)$, **do**

cover E_i with a set T_i of $\left\lfloor \frac{|E_i|}{2} \right\rfloor$ 2-chains and possibly a 1-triangle (if $|E_i|$ is odd). Set $T^H \leftarrow T^H \cup T_i$ and $E \leftarrow E \setminus E_i$.

end

Output T^H.

End of (H_2)**.**

In case the input graph contains a set of isolated vertices to be covered, the same adjustment applied to (H_1) is applied to (H_2).

THEOREM 2. *(H_2) is a linear time 1.5-approximation algorithm.*

Proof. Phase 1 of Algorithm (H_2) is linear in m since it requires for each component one breadth-first search starting from vertex v and one starting from w. Phase 2 is identical to phase 2 of Algorithm (H_1) and its running time is likewise linear (see the proof of Lemma 1).

In case there are no isolated vertices it suffices to prove the bound of 1.5 for each nontrivial component of G. The reason is that no triangle in any optimal cover can cover edges from different components of G. Hence, assume without loss of generality that the input graph G is connected. Let $z^H = |T^H|$ and z^* be, respectively, the number of cliques used by Algorithm (H_2) and the optimal number of cliques used to cover the edges and vertices of G. Then

$$z^H = \left\lfloor \frac{|E|+1}{2} \right\rfloor \text{ and } z^* \geq \left\lceil \frac{|E|}{3} \right\rceil.$$

It follows that

$$z^H \leq \frac{|E|+1}{2} = \frac{3}{2}\frac{|E|}{3} + \frac{1}{2} \leq \frac{3}{2}\left\lceil \frac{|E|}{3} \right\rceil + \frac{1}{2} \leq \frac{3}{2}z^* + \frac{1}{2}.$$

Note that if $|E| \neq 6k+3$, then equality cannot hold throughout, and so we find $z^H \leq 1.5z^*$. In case $|E| = 6k+3$, phase 1 of the algorithm tries to detect a triangle containing a vertex w. If such a triangle is not found, then we know that $z^* \geq \frac{|E|}{3} + 1$, and again equality does not hold throughout. In the case when a triangle is detected, its edges are deleted from E. Note that the resulting graph is still connected. Hence the heuristic will find in phase 2 a number of triangles equal to $\frac{|E|-3}{2}$. As a result $z^H = 1 + \frac{3}{2}\frac{|E|-3|}{3} \leq \frac{3}{2}\frac{|E|}{3}$, which settles this case.

Next we consider the case when there are isolated vertices to be covered. Let W be the set of isolated vertices, and let s be the number of odd components in graph G, at the start of phase 2. Obviously, if $|W| \leq s$, then the heuristic finds the same number of triangles as without the isolated vertices. It follows immediately that again $z^H \leq 1.5z^*$. We only need consider the case when $|W| \geq s+1$. Let t and u denote the number of components in graph G for which the number of edges is 1 modulo 3 and 2 modulo 3, respectively. Let E denote the set of edges of G at the start of phase 1, and E' the set of edges at the start of phase 2. Then the bound for z^* can be sharpened to

$$z^* \geq \frac{|E| - |E'|}{3} + \frac{|E'| + 2t + u}{3} + \frac{|W| - t}{3}.$$

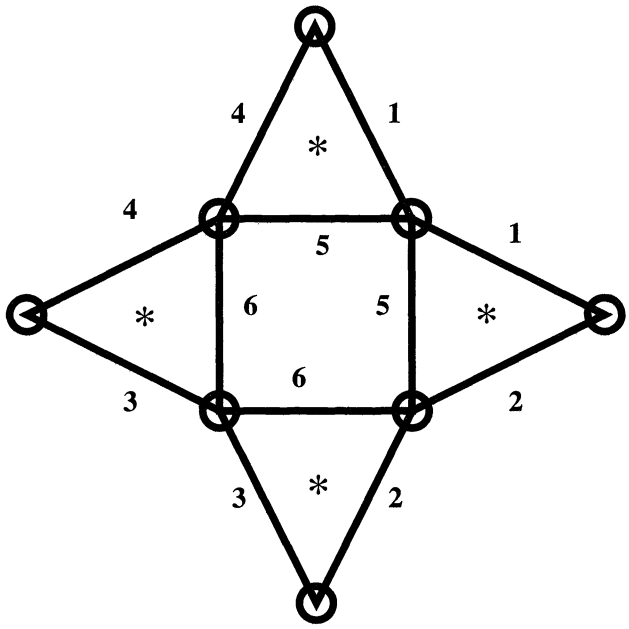


FIG. 2. *Illustration of the worst-case bound for Algorithm* (H_2).

As in the proof of Theorem 1 we derive

$$\begin{aligned} z^H &= \frac{|E|-|E'|}{3} + \frac{|E'|+s}{2} + \left\lceil \frac{|W|-s}{3} \right\rceil \\ &\leq \frac{|E|-|E'|}{2} + \frac{|E'|+s}{2} + \left\lceil \frac{|W|-s}{3} \right\rceil \\ &\leq \frac{|E|-|E'|}{2} + \frac{|E'|+s}{2} + \frac{|W|-s}{2} + \frac{1}{2} \\ &= \frac{|E|-|E'|}{2} + \frac{|E'|+|W|+t+u}{2} + \frac{1-t-u}{2} \\ &\leq \frac{3}{2}z^* + \frac{1-t-u}{2}. \end{aligned}$$

For $t+u \geq 1$ this settles the proof. For $t+u=0$ equality can hold throughout only when $|E| = |E'|$ and $|W|$ are multiples of 3, $z^* = (|E|+|W|)/3$ and $|W|-s=1$. Hence each component can be partitioned into triangles. But then a contradiction follows from the fact that $s \neq 0$, so there must be a component of odd size $6k+3$, for which the algorithm fails to find a triangle in phase 1. □

The bound is tight, as is shown by taking as input a graph, the components of which are the union of an even number of triangles. See Figure 2 for an example.

One might think that the following postprocessing of phase 2 of Algorithm (H_2) could improve the outcome: consider the 2-chains of phase 2, one by one. If there is an edge which forms a triangle with the two edges of the 2-chain under consideration, one covers this edge together with the 2-chain and deletes it from its own 2-chain. At the end of this post-processing, one eliminates the 2-chains which are now empty, i.e., the ones from which two edges have been deleted. The example in Figure 2 shows that the bound is not improved by applying this postprocessing procedure.

3.1.2. The 4-clique covering. Here we consider the problem of covering the edges and isolated vertices of a graph with 4-cliques. We present a linear time $\frac{7}{3}$-approximation algorithm, Algorithm (H_3).

An alternative approximation algorithm is the *modified greedy* applied to the problem presented as a set covering problem. The worst-case bound of the *modified greedy* in this case is $\frac{23}{12}$ (see [9]), which is better than $\frac{7}{3}$. The running time of the *modified greedy* is $O(n^4 \log n)$, where n is the number of vertices of the graph, which is worse than the running time of Algorithm (H_3), $O(m)$.

Algorithm (H_3) runs in two phases. In the first phase we cover a maximal number of edges with 4-cliques such that each 4-clique covers at least 3 edges. In the second phase, the remaining edges and the isolated vertices $W \subseteq V$ are covered optimally with a minimum number of 4-cliques.

An iteration of phase 1 consists of covering a connected subgraph induced on 4 vertices by a 4-clique and then of deleting the covered edges. Phase 1 proceeds until no connected component of G has more than 3 vertices. At the end of phase 1, all isolated vertices in $V \setminus W$ are deleted.

At the beginning of phase 2, the connected components of G, $G_1, G_2, \ldots, G_s$ are triangles, isolated vertices of W, single edges, or 2-chains (two adjacent edges). The single edges are covered two per 4-clique (if the number of isolated edges is odd, then the extra edge is covered alone) and each 2-chain or triangle is covered by its own 4-clique.

The algorithm is given below.

ALGORITHM (H_3).
Input: Graph $G = (V, E)$.
Set $C^H = \emptyset$.
Phase 1:
Do until no connected component of $G = (V, E)$ has more than three vertices:
Find a connected subgraph on 4 vertices, C.
Set $C^H \leftarrow C^H \cup C$. Let E_C be the edge set of C. Set $E \leftarrow E \setminus E_C$.
enddo
Phase 2: Let T be the set of triangles and 2-chains of G. Set $C^H \leftarrow C^H \cup T$. Let C_F be the set of 4-cliques required to cover the set F of isolated edges of G (two per 4-clique). Set $C^H \leftarrow C^H \cup C_F$.
Output C^H.
End of (H_3).

In the case when the input graph contains a set of isolated vertices to be covered, the following adjustment is applied to (H_3). At the termination of phase 2, we assign isolated vertices: one for each triangle or 2-chain in T, two for a single-edge clique. The remaining isolated vertices are covered 4 per clique with possibly 1, 2, or 3 vertices in the last clique used. We append these additional cliques to the set C^H.

The complexity and worst-case performance of Algorithm (H_3) are given by the next theorem.

THEOREM 3. *(H_3) is a linear time $\frac{7}{3}$-approximation algorithm.*

Proof. Phase 1 can be implemented using a linear time depth-first-search technique. Phase 2 scans through the remaining edges once. The assignment of isolated vertices will take linear time as well. Thus, the complexity of the algorithm is linear in the number of edges and isolated vertices, $O(m)$.

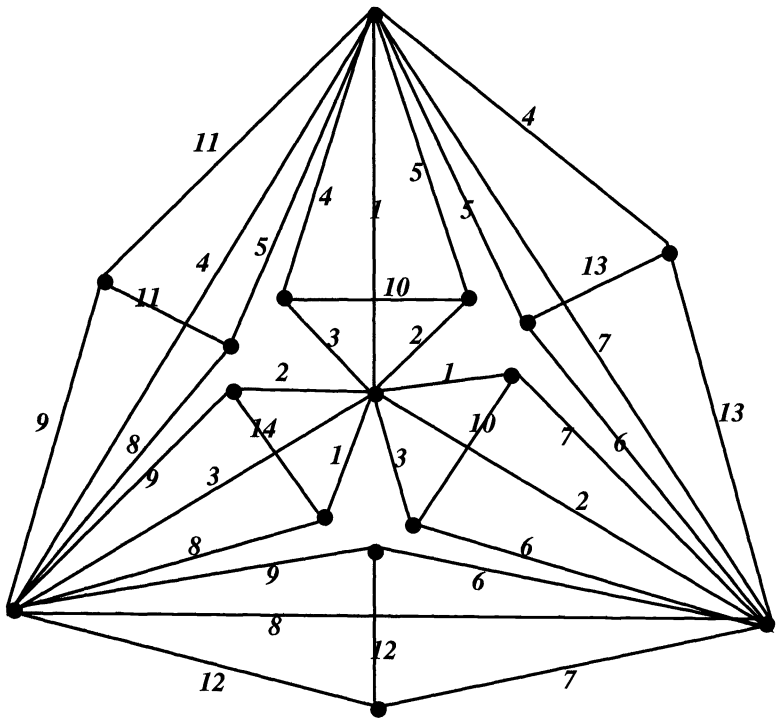


FIG. 3. *Illustration of the worst-case bound for the clique size* 4.

Let z^* be the optimal number of 4-cliques and αz^* be the total number of 4-cliques used during phase 1. Because at least 3 edges are covered by each 4-clique used in phase 1, the total number of edges covered during phase 1 is at least $3\alpha z^*$. Also, no more than 6 edges can be covered by a 4-clique. Therefore, the number of elements which remain to be covered during phase 2 is at most $(6 - 3\alpha)z^*$. The number of 4-cliques used during phase 2 is at most

$$\min\left\{\left\lceil\frac{(6-3\alpha)z^*}{2}\right\rceil, z^*\right\}.$$

The first term follows from the fact that a phase 2 4-clique covers at least two edges. The second term is because the remaining edges and isolated vertices are covered optimally and therefore must use at most z^* 4-cliques. Hence, the total number of 4-cliques used by the approximation algorithm is at most

$$\begin{aligned} z^H &\le \alpha z^* + \min\left\{\left\lceil\frac{(6-3\alpha)z^*}{2}\right\rceil, z^*\right\} \\ &\le \min\left\{\frac{(6-\alpha)z^*}{2} + \frac{1}{2}, (1+\alpha)z^*\right\} \\ &\le \frac{7}{3}z^* + \frac{1}{3}. \end{aligned}$$

The last inequality is derived by setting $\frac{(6-\alpha)z^*}{2} + \frac{1}{2} = (1+\alpha)z^*$. □

An example in Figure 3 illustrates the tightness of the bound. In this example, there are 24 edges forming 6 cliques of size 4. The optimal solution is $z^* = 6$. By

applying algorithm (H_3), edges are covered in the order in which they are labeled in the graph. Numbers associated with each edge indicate which clique the corresponding edge belongs to. Algorithm (H_3) uses 14 cliques, with the first 9 cliques containing 3 edges each, the next 4 cliques containing 2 edges each and the last clique containing only one edge. Thus the ratio $\frac{z^H}{z^*}$ is $\frac{14}{6} = \frac{7}{3}$.

3.1.3. k-clique covering in a graph. We extend our discussion to the general clique size, k, for the graph $G = (V, E)$. For this case the *greedy* for set covering gives a bound of $\mathcal{H}(\binom{k}{2}) \approx 2 \log k$ with running time $O(n^k)$, and the *modified greedy* [9] improves the bound to $\mathcal{H}(\binom{k}{2}) - \frac{1}{6}$. In this section, we present a linear time $(\frac{1}{2} + \frac{1}{k-1})k$-approximation algorithm (H_4) for the k-clique covering problem. Although the quality of the bound is worse than that derived from the set covering representation of the problem, the running time is significantly reduced.

Algorithm (H_4) finds a cover with star-shaped components. Phase 1 of (H_4) finds a sequence of vertices and edges, such that each edge and each vertex occurs once, and such that all edges in the sequence between consecutive vertices v_i and v_{i+1} are incident with v_i. The isolated vertices are appended at the end of this sequence. In phase 2 the sequence is broken into batches of size at most $k-1$. Such a batch of $k-1$ edges and vertices induces a graph on at most k vertices, and forms a clique. The algorithm is stated as follows.

ALGORITHM (H_4).
Input: Graph $G = (V, E)$.
Phase 1: Set $SEQ = \emptyset$, $\bar{V} = V$, $\bar{E} = E$;
while $\bar{V} \neq \emptyset$ **do begin**
Select a vertex $v \in \bar{V}$;
Set $SEQ = (SEQ, \{v\})$;
repeat
Find a vertex $u \in V \setminus \bar{V}$ such that $(u, v) \in \bar{E}$;
Set $SEQ = (SEQ, \{u, v\})$;
$\bar{E} \leftarrow \bar{E} \setminus \{(u, v)\}$;
until no such u can be found;
$\bar{V} \leftarrow \bar{V} \setminus \{v\}$;
end;
Let $SEQ = S_1, S_2, \ldots, S_N$, where $N = |E| + |V|$.
Phase 2: Set $C^H = \emptyset$, $\bar{k} = 0$;
while $\bar{k} < N$ **do begin**
Let C denote the component with vertex set $V(C) = S_{\bar{k}+1} \cup \cdots \cup S_{\bar{k}+k-1}$ and edge set $E(C) = \{S_{\bar{k}+1}, \ldots, S_{\bar{k}+k-1}\} \cap E$;
$C^H \leftarrow C^H \cup C$;
$\bar{k} = \bar{k} + k - 1$;
end;
Output $z_H = |C^H|$ as the number of cliques used by the algorithm;
End of (H_4)**.**

The complexity and a bound on the worst-case performance of Algorithm (H_4) are given in the following theorem.

THEOREM 4. *(H_4) is a linear time $(\frac{k}{2} + \frac{k}{k-1})$-approximation algorithm.*

Proof. Phase 1 can be realized in linear time by a simple breadth-first search. Phase 2 is obviously linear in the length of the sequence, $O(m)$.

Let z^H be the number of cliques used for covering edges by heuristic (H_4), and let z^* be the number used by an optimal clique covering. Let w denote the number of isolated vertices, V the set of nonisolated vertices, and E the set of edges. Then

$$z^H = \left\lceil \frac{|E| + |V| + w}{k-1} \right\rceil$$

and

$$kz^* \geq w + \sum_{v \in V} \left\lceil \frac{\delta(v)}{k-1} \right\rceil,$$

as each vertex v of degree $\delta(v)$ occurs in at least

$$\left\lceil \frac{\delta(v)}{k-1} \right\rceil$$

cliques. The latter number is bounded from below by $w + \frac{2|E|}{k-1}$, and also by $w + |V|$. It follows that

$$\begin{aligned} z^H &\leq \frac{|E| + |V| + w}{k-1} + 1 \\ &\leq \frac{1}{2}kz^* + \frac{1}{k-1}kz^* + 1. \end{aligned}$$

Note that for $k > 4$, this gives $z^H/z^* \leq \frac{3}{4}k$, and for increasing k, the bound goes to $\frac{1}{2}k$. □

The bound is tight, as can be seen by taking as input a graph consisting of N k-cliques. Each clique has $\frac{k(k-1)}{2}$ edges and k vertices. The algorithm partitions these edges and vertices in

$$\left\lceil N\left(\frac{k}{2} + \frac{k}{k-1}\right) \right\rceil$$

batches of size $k-1$, whereas the optimal cover uses N cliques.

3.2. k-clique covering of hyperedges of size $k-1$. In this section, we consider the k-clique covering problem for hypergraphs that have all hyperedges of size $k-1$. Following the notation introduced earlier, this is the case $(k, k-1)$. This problem is a set covering problem with all sets of size $\leq k$. This is because at most k hyperedges of size $k-1$ each fit in a clique on k vertices.

The complexity of applying the *greedy* or the *modified greedy* to this set covering problem is $\Omega(n^k)$ since the number of sets to be considered is $\binom{n}{k}$. The purpose of algorithm (H_5) is to select a small collection of $O(m^2)$ sets to be considered.

Let the hyperedges to be covered be $\{E_1, \ldots, E_m\}$. Two hyperedges (elements) E_i, E_j are called a *spanning pair* if $|E_i \cup E_j| = k$. Any k-clique containing more than one hyperedge must have a spanning pair.

ALGORITHM (H_5).
Input: $V; E = \{E_1, \ldots, E_m\}$.
Phase 1: {Generate a collection of spanning pairs}
Set $S = E$; {Include all singletons in the covering sets}
for all pairs $E_i, E_j \in E$ **do**

if $|E_i \cup E_j| = k$ **then**
$S \leftarrow S \cup \{E_i \cup E_j\}$; {Include all spanning pairs}
Phase 2:
Apply *modified greedy* to the set covering problem with a collection of sets S and a collection of elements $\{E_1, \ldots, E_m\}$;
End of (H_5).

THEOREM 5. (H_5) *is an* $(\mathcal{H}(k) - \frac{1}{6})$*-approximation algorithm with running time* $O(m^2k + m^{2.5})$.

Proof. Algorithm (H_5) reduces the computational complexity of the *greedy* set covering heuristic by limiting the number of covering sets. The set covering matrix developed in phase 1 has at most $\binom{m}{2} + m$ columns and m rows. To check if a hyperedge is covered by a set, we need to scan the hyperedge's vertex set, which takes $O(k)$ time. Since the *modified greedy* applied in phase 2 takes only $O(m^{2.5})$ [9], the overall running time of (H_5) is $O((\binom{m}{2} + m)k + m^{2.5}) = O(m^2k + m^{2.5})$.

As observed earlier, any clique containing more than one hyperedge must have a spanning pair. Algorithm (H_5) has included all spanning pairs together with singleton sets $E_i, i = 1, \ldots, m$ in the collection of sets S. Thus all possible covering sets are accounted for in solving the set covering problem. Therefore, using the *modified greedy* for this reduced problem is the same as for the original problem, and the worst-case bound is then $\mathcal{H}(k) - \frac{1}{6}$. □

Now we provide another heuristic, (H_6), based on graph matching. This algorithm has better complexity than (H_5) if $k > \sqrt{m}$, but its bound quality is not as good.

ALGORITHM (H_6).

Step 1: Construct the following undirected graph $G = (V, E)$: Each vertex represents a hyperedge; Two vertices i and j are connected by an edge if and only if the corresponding hyperedges i and j form a spanning pair.

Step 2: Find a maximum cardinality matching in G.

Step 3: Put each pair of hyperedges corresponding to the end vertices of a matching edge obtained in Step 2 in a single clique. Put the remaining unmatched vertices (hyperedges) in separate cliques. Output the collection of cliques C^H.

End of (H_6).

The following theorem gives the worst-case bound of Algorithm (H_6).

THEOREM 6. *Algorithm* (H_6) *is a* $\lceil \frac{k}{2} \rceil$*-approximation algorithm with running time* $O(m^{2.5})$.

Proof. For two hyperedges to be packed together, their corresponding vertices must be connected by an edge in G. It suffices to prove the bound for any connected component of G, so assume that G is connected.

Let t_i be the number of hyperedges packed in the ith clique of the optimal solution. These t_i hyperedges must form a t_i-clique in G. There exists a feasible matching with $\lfloor \frac{t_i}{2} \rfloor$ edges in each of these t_i-cliques. If t_i is odd, then the number of sets required to cover these nodes is less than or equal to $\lceil \frac{t_i}{2} \rceil$. Hence Step 3 results in a collection of $\sum_{i=1}^{z^*} \lceil \frac{t_i}{2} \rceil$ cliques. Let $z^H = |C^H|$ and z^* be the optimal number of cliques.

Since $t_i \leq k$, $z^H \leq \sum_{i=1}^{z^*} \lceil \frac{t_i}{2} \rceil \leq z^* \lceil \frac{k}{2} \rceil$.

The complexity of Algorithm (H_6) is dominated by Step 2 for finding a maximum matching. The complexity of Step 2 is $O(m^{2.5})$[20]. Therefore the overall running

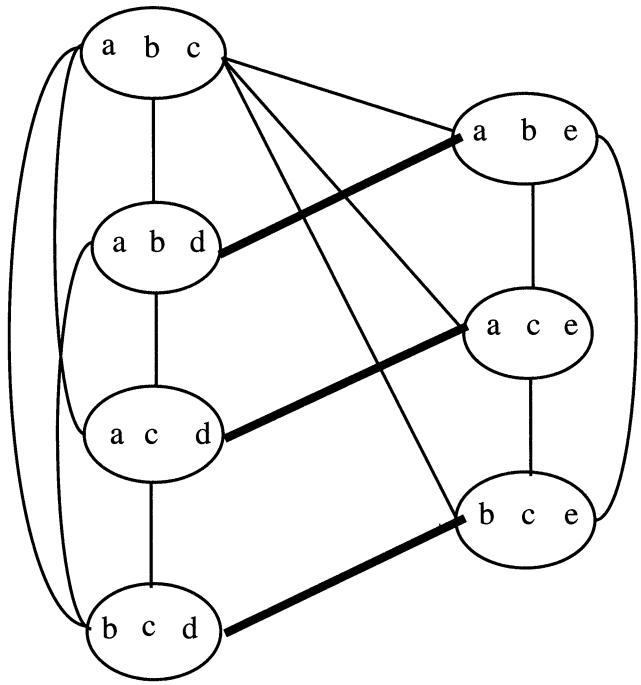


FIG. 4. *Illustration of the worst-case bound for* $(k-1)$*-hyperedges and cliques of size* k.

time is $O(m^{2.5})$. □

Figure 4 illustrates the tightness of the bound of Theorem 6. The four hyperedges on the left side can be packed in one clique of size 4. The same holds for the three hyperedges on the right side. Therefore the optimal number of cliques is 2. The maximum matching obtained by the heuristic are the three bold edges of the figure. The last hyperedge $(\{a, b, c\})$ has to be packed in a clique by itself. It follows that the number of cliques used by the heuristic is 4, which is $\frac{k}{2}$ times the optimal number of cliques.

4. Concluding remarks. In this paper, we introduced the problem of covering the edges of a hypergraph by k-cliques (CC_k). The problem is shown to be NP-hard and we describe a number of approximation algorithms. We identify the link between the (CC_k) problem and the set covering problem. The approximation algorithms described here are the first such algorithms for the k-clique covering problem. We describe a range of approximation algorithms applicable to various subclasses of the problem, with several algorithms for the same subclass offering a trade-off between the algorithm's running time and the quality of the approximate solution.

One natural generalization of this problem is the case where each clique has a different weight. If the occurrence of such a weighted case is practical, then there is a need for extending the results to the weighted case. Indeed, the set covering heuristic presented is immediately extendible, but such is not the case for every heuristic presented and analyzed.

Our study still leaves open the challenging task of coming up with optimal solutions to the (CC_k) problem. We believe that our approximation approach will prove instrumental and useful in algorithms that derive such optimal solutions.

REFERENCES

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and intractability of approximation problems*, in Proc. 33rd IEEE Symp. on Foundations of Computer Science, Pittsburgh, PA, IEEE Computer Society Press, Piscataway, NJ, 1992, pp. 14–23.

[2] M. L. Balinski, *On a selection problem*, Management Sci., 17 (1970), pp. 230–231.

[3] J. F. Bard, *A heuristic for minimizing the number of tool switches on a flexible machine*, IIE Trans., 20 (1988), pp. 382–391.

[4] V. Chvátal, *A greedy heuristic for the set-covering problem*, Math. Oper. Res., 4 (1979), pp. 233–235.

[5] D. A. Collier, *The measurement and operating benefits of component part commonality*, Decision Sci., 12 (1981), pp. 85–96.

[6] Y. Crama and A. Oerlemans, *A column generations approach to job grouping for flexible manufacturing systems*, European J. Oper. Res., to appear.

[7] M. Daskin, P. C. Jones, and T. J. Lowe, *Rationalizing tool selection in a flexible manufacturing system for sheet-metal products*, Oper. Res., 38 (1990), pp. 1104–1115.

[8] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, *A fast parametric maximum flow algorithm*, SIAM J. Comput., 18 (1989), pp. 30–55.

[9] O. Goldschmidt, D. S. Hochbaum, and G. Yu, *A modified greedy heuristic for the set covering problem with improved worst-case bound*, Inform. Process. Lett., 48 (1993), pp. 305–310.

[10] D. S. Hochbaum, *Approximation algorithms for the weighted set covering and node cover problems*, SIAM J. Comput., 11 (1982), pp. 555–556.

[11] R. Hirabayashi, H. Suzuki, and N. Tsuchiya, *Optimal tool module design problem for NC machine tools*, J. Oper. Res. Soc. Japan, 23 (1984), pp. 205–228.

[12] J. Holyer, *The NP-completeness of some edge-partition problems*, SIAM J. Comput., 10 (1981), pp. 713–717.

[13] S. Hwang, *A constraint-directed method to solve the part selection problem in flexible manufacturing systems planning stage*, in Proc. 2nd ORSA/TIMS Conference on Flexible Manufacturing Systems, Elsevier, New York, 1986, pp. 297–309.

[14] S. S. Hwang and A. W. Shogan, *Modelling and solution of an FMS part selection problem*, Internat. J. Prod. Res., 27 (1989), pp. 1349–1366.

[15] A. Itai and M. Rodeh, *Finding a minimum circuit in a graph*, SIAM J. Comput., 7 (1978), pp. 413–423.

[16] D. S. Johnson, *Approximation algorithms for combinatorial problems*, J. Comput. System Sci., 9 (1974), pp. 256–278.

[17] L. Lovász, *On the ratio of optimal integral and fractional covers*, Discrete Math., 13 (1975), pp. 383–390.

[18] C. Lund and M. Yanakakis, *On the hardness of aproximating minimization problems*, in Proc. 25th Annual ACM Symp. on Theory of Computing, 1993, ACM Press, New York, pp. 286–293.

[19] S. Masuyama and T. Ibaraki, *Chain packing in graphs*, Tech. Rep. 87014, Dept. of Applied Mathematics and Physics, Kyoto University, Japan, 1987.

[20] S. Micali and V. V. Vazirani, *An $O(|V|^{1/2}|E|)$ algorithm for finding maximum matching in general graphs*, in Proc. 21st Annual IEEE Symp. on Foundations of Computer Science, Long Beach, CA, 1980, pp. 17–27.

[21] J. M. W. Rhys, *A selection problem of shared fixed costs and network flows*, Management Sci., 17 (1970), pp. 200–207.

[22] C. S. Tang and E. V. Denardo, *Models arising from a flexible manufacturing machine, Part II: Minimizing the number of switching instants*, Oper. Res., 36 (1988), pp. 778–784.

[23] C. Whitney and T. Gaul, *Sequential decision problems for batching and balancing in FMSs*, Ann. Oper. Res., 3 (1985), pp. 301–316.

[24] G. Yu, D. Nehme, and N. Nayak, *Designing and Managing Multiple Product Setups for Electronic Circuit Board Assembly Process*, IBM Internal Report TR51.0711, Austin, TX, 1992.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 511–528, November 1996

GRAPH ALGORITHMS FOR CONFORMANCE TESTING USING THE RURAL CHINESE POSTMAN TOUR *

YINAN N. SHEN† AND FABRIZIO LOMBARDI‡

Abstract. This paper presents new results and graph algorithms for the automatic testing of protocols using "unique input/output" (*UIO*) sequences. *UIO* sequences can be efficiently employed in checking conformance of protocols to their specifications by using transition testing. The optimization of the test sequence is based on finding the rural Chinese postman tour of the state transition diagram of a finite state machine (FSM).

The process of conformance test generation using a touring algorithm is valid provided that certain connectivity properties of the graph are present. This implies that a weakly connected graph must be constructed. It is possible that this connectivity condition may not be met when multiple *UIO* sequences are used even if the reset capability and/or the self-loop properties are present. The "weakly connected graph problem" consists of finding an edge-induced subgraph of the FSM which is still weakly connected when multiple *UIO* sequences are used. The "multiple *UIO* tour minimization problem" addresses the assignment of edges to *UIO* sequences for minimizing the degree of the directed *UIO* graph. This process may not also minimize the length of the tour. The above two problems, left open in previous papers, are solved in this paper. It is proved that by appropriately changing the original assignment graph and using network flow techniques with a new *UIO* generation process referred to as chaining, efficient solutions can be provided. The theoretical approaches behind the solution to these problems are fully characterized.

Key words. graph algorithms, rural Chinese postman tour, finite state machine, protocol

AMS subject classifications. 68M15, 68R10

1. Introduction.

A protocol is a precise set of rules which defines the possible interactions among components of a communication system [1]. The complexity of today's communication systems makes the use of automated tools for the verification and validation of protocols imperative. This process is commonly referred to as *conformance testing.* Conformance testing has been widely advocated for ensuring that protocol implementations abide to their specifications when they are designed and/or installed [9].

Protocol implementations consist of a combination of both hardware and software components [1]. Users and manufacturers analyze and test protocols using a black box approach [19, 20]; i.e., the protocol must behave according to the specifications which outline its behavior by abiding to accepted standards (such as those dictated by professional groups [9, 15]) independent of its physical implementation. Such an approach is widely used because it yields to the certification process often required by international governing bodies such as the International Telegraph and Telephone Consultative Committee [15]. Conformance testing of protocols is therefore implementation independent and is based on a functional characterization; this characteristic precludes the use of traditional testing techniques which are viable for either hardware or software [19, 20].

Protocols are commonly tested using switching-based approaches [6, 10] in which

* Received by the editors October 19, 1992; accepted for publication (in revised form) November 10, 1995. This research was supported in part by grants from AT&T, North Atlantic Treaty Organization, and Texas Advanced Research Program.

† Actel Corporation, Sunnyvale, CA 94086.

‡ Texas A&M University, Department of Computer Science, College Station, TX 77843-3112 (lombardi@cs.tamu.edu).

the protocol is functionally modeled as a Mealy machine [2]. Examples of these approaches are the checking experiment, the W-set, and the touring method [2, 4, 5, 17]. In these approaches, state and edge verification of the finite state machine (FSM) is performed. Verification is accomplished using a so-called characterizing sequence (CS) [17] such as the distinguishing sequence (DS) which provides a distinct signature for each state of the FSM.

Recently, the use of unique input/output (UIO) sequences as characterizing sequences has been advocated for the purpose of testing a protocol [6]. A UIO sequence is an input/output sequence which is unique to a state of the FSM. UIO sequences are shorter than the DS of a checking experiment [6].

Testing of the protocol is accomplished by testing each edge in the graph representation of the FSM. Using the UIO method, each edge is verified by constructing a test subsequence. A test subsequence consists of the transition of the edge under test (thus observing the desired output) followed (or concatenated) by the appropriate UIO sequence with uniquely checks its end state. The test sequence of the protocol is generated by concatenating all test subsequences. An optimization technique for generating a test sequence using the UIO method has been proposed in [10]. This technique generates the test sequence of a protocol using the rural Chinese postman tour [12]. Since all edges of the FSM graph (i.e., the transitions of the FSM) must be tested, each vertex must be visited a number of times equal to or greater than the vertex in-degree. In [18], it was proved that there may exist multiple minimum-length UIO sequences for any given vertex (or state). Thus, one must choose which UIO sequence to use each time a state is visited. A proper assignment of UIO sequences to edges (i.e., the visits) can result in a shorter test sequence. An algorithm for producing such an assignment by means of network flow techniques has been given in [18]. The assignment is based on the minimization of the degree of the graph, usually achieved by augmentation [10]. Both real and randomly generated FSMs have been studied as test cases and have indicated that the savings in reducing the length of the tour are substantial.

Several issues are unsolved using the approach of [18]. First, the assignment of UIO sequences to edges may not minimize the length of the tour. This is due to the augmentation process which is employed in [18]. For example, a given UIO sequence assignment may lead to two edges of unit cost being replicated while an "optimal" assignment may lead to the replication of a single edge of high cost. The multiple UIO tour length minimization problem consists of finding the minimum length of the tour with multiple UIO sequences. In [10] it was also shown that a sufficiency condition for the validity of the rural Chinese postman tour algorithm is that the edge-induced subgraph derived from the FSM must be weakly connected. It has been proved [10] that if the FSM has reset and/or self-loops, the weakly connected condition can be satisfied; however, this condition is not met when multiple UIO sequences are used, therefore increasing the cost of the tour. The weakly connected graph problem consists of finding an edge-induced subgraph from the FSM which is still weakly connected when multiple UIO sequences are used.

The objective of this paper is to extend the basic results of [6, 10] for automatically testing communication protocols. The two problems discussed above will be addressed. Their characterization and solutions will be presented. This paper is organized as follows. Section 2 deals with a brief review of the approaches of [6, 8] as applicable to the proposed approach. Notation and definitions are also introduced. Section 3 introduces the UIO method of [18]. Section 4 presents the solution to the weakly connected graph problem. Section 5 deals with the analysis for the multiple UIO tour length minimization. In Appendix A, the definitions of the various graphs used in the

analysis are given for clarity.

2. Preliminaries. A protocol can be specified as a *deterministic finite state machine* (FSM) [10]; FSM $= \{S, I, O, NS, Z\}$ where $S = \{s_1, \ldots, s_n\}$ is a finite set of states, $I = \{i_1, \ldots, i_M\}$ is a finite set of inputs (or stimuli), and $O = \{o_1, \ldots, o_N\}$ is a finite set of outputs. The FSM operates according to two functions: the next state (NS) and the output functions (Z) are given by the two mappings NS: $S \times I \rightarrow S$ and Z: $S \times I \rightarrow O$, respectively. Machines specified in a form such that the output is a function of the current state and the applied input are commonly referred to as Mealy machines [3].

An FSM is represented by a directed graph $G = (V, E)$, where the set $V = \{v_1, \ldots, v_n\}$ represents the set of specified states S of the FSM and a directed edge (v_i, v_j) with label $L = a_k/o_l$ (denoted as $(v_i, v_j; L)$) represents a transition from state s_i to state s_j (with an operation given by an input a_k and an output o_l to the FSM). A *cost* is associated with each edge $(v_i, v_j; L)$. This usually is the time taken to realize the corresponding transition in the FSM. For clarity, in this paper it is initially assumed that the cost of every edge is 1. The conclusions of this paper can be generalized to the case with arbitrary cost. Hereafter, the terms FSM and graph G will be used interchangeably.

An FSM implementation can be tested using a method commonly referred to as the touring method [10]. In this method, a characterizing sequence (CS) is used: a CS is a sequence of inputs and outputs which exhibit some distinctive signature for each state of the FSM. The most common touring method is the *checking experiment* [2]. In a checking experiment, there is a characterizing sequence of input/output pairs $(i_1/o_1), (i_2/o_2) \ldots$ such that the response of the implementation to $i_1, i_2, \ldots$ is $o_1, o_2, \ldots$ if and only if the implementation behaves according to its specification. Checking experiments are based on the existence of an input sequence, called a *distinguishing sequence* (DS), as CS. A DS produces a distinct output sequence for each initial state of the FSM [2].

3. The *UIO* method. A new approach for protocol testing has been proposed in [6, 7]. This is based on the use of UIO sequences: a UIO sequence for a state s_i is an input/output sequence $UIO_i = (a_{k_1}/o_{l_1}) \cdots (a_{k_r}/o_{l_r})$ only to be observed when s_i is the initial state such that there is no $s_j \neq s_i$ for which UIO_i is a specified input/output sequence for initial state s_j [6]. Note that if the cost of every edge is not 1, "minimum length" should be substituted by "minimum cost." The minimum-cost test sequence for checking the correctness of a transition from state s_i to state s_j with input/output a_k/o_l is denoted as $TEST(v_i, v_j; a_k/o_l)$.

The first step of a touring method using UIO sequences [10, 17] is to construct the test subsequence set as follows: the *test subsequence set* (TSS) has as elements the *test subsequences* of all transitions of the FSM. Let $TAIL(E_i)$ ($HEAD(E_i)$) of a transition E_i denote the target (source) state of E_i, and, for a sequence of S transitions, let $TAIL(S)$ ($HEAD(S)$) denote the $TAIL$ ($HEAD$) of the last (first) transition in the sequence. The generation of a test subsequence of a transition E_i (TSS_i) consists of the following three steps (where the label of E_i is $INPUT_i/OUTPUT_i$ and $TAIL(E_i) = NEXTSTATE_i$ if the FSM is fault-free):

1. Put the FSM in state $HEAD(E_i)$.
2. Apply $INPUT_i$ and check if the output is $OUTPUT_i$.
3. Apply a CS to check if the tail state is $NEXTSTATE_i$.

Therefore, TSS_i can check whether both the output of E_i and the nextstate function of E_i are as expected. If UIO sequences are used as CS, then the test subsequence for E_i is given by $TSS_i = E_i \bullet UIO_k$, where UIO_k is the UIO sequence

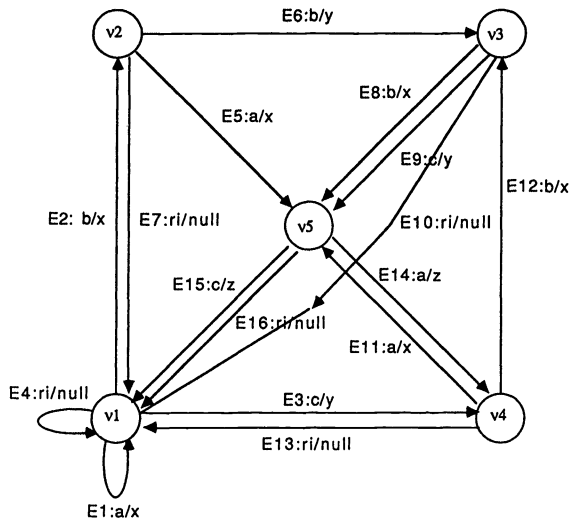


FIG. 1. *The graph representation of an FSM. All edges have an equal cost, and the label ri/null represents a reset edge.*

for state k, $\bullet$ stands for concatenation, E_i is the edge under test, and $k = TAIL(E_i)$.

A *test sequence* must contain the test subsequences of all transitions (edges) referred to as the *test subsequence set.* To concatenate two test subsequences, TSS_i and TSS_j, a *bridge sequence* BS_{ij} is required to transfer the tail state of TSS_i to the head state of TSS_j. If $TAIL(TSS_i) = HEAD(TSS_j)$ then $BS_{ij} = null$ sequence. The total length of the test sequence is therefore the sum of all test subsequences and all required bridge sequences.

The minimum-cost test sequence can be derived from the test subsequence set by constructing a new graph. This graph has the same vertex set of the FSM, but its edge set consists of two subsets: one is the transition set of the FSM and the other is the TSS set of the FSM. The test sequence in the new graph is a tour which traverses all the TSSs exactly once and the other edges as little as possible. This tour is called the rural Chinese postman tour [10, 13, 16]. The algorithm is as follows: first find the minimum-length UIO sequences of the FSM using a breadth-first search [6, 10], then find the edge set $E_C, E_C = \{(v_i, v_k; L_l \bullet UIO_j) | (v_i, v_j; L_l) \in E$ and $TAIL(UIO_j) = v_k\}$, where $TAIL(UIO_j)$ is defined as the final state of the sequence UIO_j. E_C directly corresponds to the TSS set. The cost of an edge $e_c \in E_C$ is defined as the sum of the costs of its components. A directed graph $G' = (V', E')$ is constructed from G such that $V' = V$ and $E' = E \cup E_C$. For example, the graph G' for the graph G of Figure 1 is shown in Figure 2. (This figure describes the same protocol as in [10].) Table 1 shows the UIO sequences for the FSM of Figure 1.

The solution to the rural Chinese postman tour problem on G' consists of the following two steps:

1. Construct a directed graph [10] $\hat{G}^* = (\hat{V}^*, \hat{E}^*)$ from G', where $\hat{V}^* \equiv V'$, each edge in E is included in $\hat{E}^*$ zero or more times, and each edge in E_C is included in $\hat{E}^*$ once and only once. The graph $\hat{G}^*$ is referred to as a *rural symmetric augmentation* of G'. The construction of $\hat{E}^*$ in [10] ensures that the total cost of edges in $\hat{E}^*$ is minimum subject to the constraint that in the in-degree of each vertex $v_i \in \hat{V}^*$ is equal to its out-degree. ($\hat{G}^*$ is therefore a *symmetric* graph.) For example, the graph $\hat{G}^*$ of the graph G' of Figure 2 is shown in Figure 3.

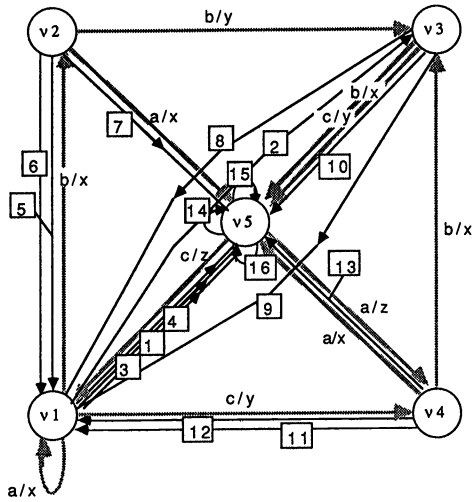


FIG. 2. *The graph G' for the graph shown in Figure* 1 (*dotted edges represent E, solid fine edges represent Ec, and reset edges are not shown*).

TABLE 1
UIO sequences for the FSM shown in Figure 1.

State	UIO sequence
v_1	$b/x \quad a/x$
v_2	b/y
v_3	$b/x \quad c/z$
v_4	$b/x \quad c/y$
v_5	c/z

2. Find an Euler tour [11, 13] of $\hat{G}^*$. As $\hat{G}^*$ is symmetric, such a tour is guaranteed to exist [13]. Note that only edges from E are used for augmentation. This is because each edge $e_c \in E_C$ is formed from a concatenation of a few edges in E; thus, the cost of E_C is greater than or equal to the cost of E. Hence, a test sequence of minimum cost can be generated only by using edges of E in the augmentation.

Define the index $\xi(v_i)$ of a vertex $v_i \in G'$ as the difference between the number of edges in E_C into v_i ($d_{in}^{E_C}(v_i)$) and the number of edges in E_C out of v_i ($d_{out}^{E_C}(v_i)$) and the degree $\Delta(G')$ of G' as $\sum_{i=1}^{n} |\xi(v_i)| = \sum_{i=1}^{n} |d_{in}^{E_C}(v_i) - d_{out}^{E_C}(v_i)|$. If $\Delta(G') = 0$, then no edge in E needs to be included in $\hat{E}^*$ in the first step of the above procedure; however, if $\Delta(G') \neq 0$, then the edges in E must be replicated for $\hat{G}^*$ to be symmetric. The optimal number of replications for each edge such that the resulting tour is of minimum cost is computed by means of a minimum-cost maximum-flow algorithm [14] on a graph $G_F = (V_F, E_F)$ constructed from G', where $V_F \equiv V' \cup \{s, t\}$ (s and t are the *source* and *target* of G_F), $E_F = E' \cup (s, v_i)$: $v_i \in C \cup (v_j, t)$: $v_j \in D$, and

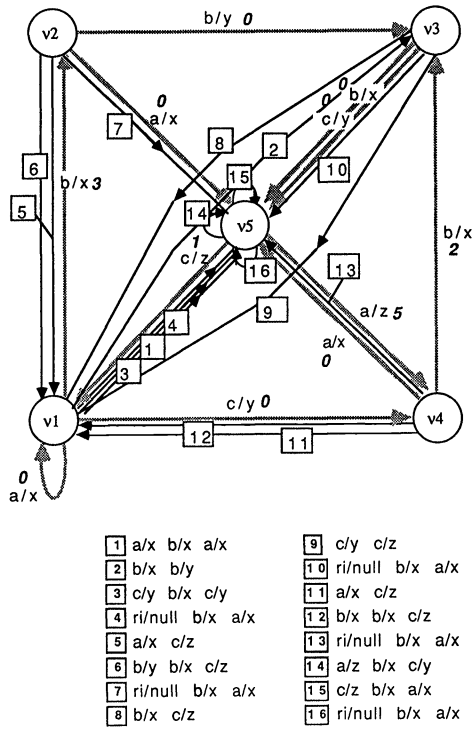


FIG. 3. *The rural symmetric augmentation* $\hat{G}^*$ *of the graph shown in Figure* 2. *The italic number associated with each edge corresponds to the number of times the edge appears in the rural symmetric augmentation (dotted edges represent* E, *solid fine edges represent* Ec, *and reset edges are not shown).*

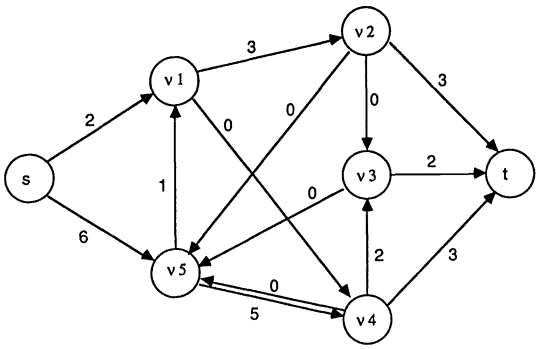


FIG. 4. *The graph* G_F *of the graph* G' *shown in Figure* 2 *and a minimum-cost maximum-flow.*

$C \subset V_F$ ($D \subset V_F$) is the set of vertices in G' with positive (negative) indices. The cost and capacity of each edge in G_F are as follows:

1. Each edge (s, v_i) has cost zero and capacity $\gamma(s, v_i) \equiv \xi(v_i)$.
2. Each edge (v_j, t) has cost zero and capacity $\gamma(v_j, t) \equiv -\xi(v_j)$.
3. The remaining edges in E_F have the same cost in G_F as in G' and have infinite capacity.

The graph G_F for the graph G' of Figure 2 is shown in Figure 4.

Given a minimum-cost maximum-flow F on G_F, a minimum-cost rural symmetric augmentation of $G', \hat{G}^*$ can be constructed from G_F [10]. A minimum-cost maximum-

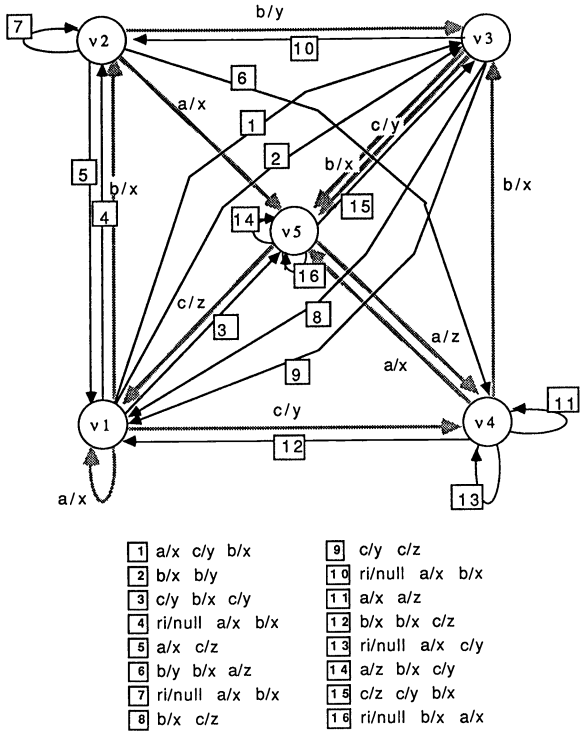


FIG. 5. *Graph G'_A with multiple UIO for the graph shown in Figure* 1 (*dotted edges represent E, solid fine edges represent Ec, and reset edges are not shown*).

flow of G_F for the graph G' of Figure 1 is given in Figure 4. The cost of the test sequence is 55 (including an initial reset input). The approach presented in [10] is such that only one G' can be generated from G, because exactly one UIO sequence per state is computed. Therefore, the value for the index of each vertex is fixed and is usually not equal to zero. This means that more edge replications occur, leading to an increased cost of the test sequence.

In [18], it has been proved that there may exist several minimum-length UIO sequences for a given state. An appropriate choice of UIO sequences for testing each edge can be used to construct an alternative graph G' with less $\Delta(G')$, therefore reducing the total length of the test sequence. Ideally, the alternative graph G' is symmetric, so no augmentation is needed; in this case, no edges from E are included in $\hat{G}^*$ and the cost of the test sequence is simply the cost of the edges in E_C. If multiple UIO sequences are used for each state, the graph G' with minimum-length UIO sequences (denoted to G'_A) is shown in Figure 5. Note that UIO_1^3 (see Table 2) is used to test transition $(v_1, v_1; a/x)$ (shown as edge 1) and UIO_1^2 is used to test the transition $(v_1, v_1; ri/null)$ (shown as edge 4). The rural symmetric augmentation $\hat{G}^*$ of the graph G' of Figure 5 is given in Figure 6.

The *multiple UIO sequence assignment problem* serves the above purpose and is defined as follows.

DEFINITION 1. *Given G and a set $MUIO_j = \{UIO_j^1, \ldots, UIO_j^{r_j}\}$ (where $r_j > 0$ is the number of minimum-length UIO sequences with distinct tail states for state s_j) for $v_j, j = 1, \ldots, n$, assign an element $UIO_j^\alpha \in MUIO_j$ for each edge $(v_i, v_j; L) \in E$ such that $\Delta(G')$ is minimized.*

To solve the multiple UIO sequence assignment problem, a directed weighted

TABLE 2
The multiple UIOs for the FSM shown in Figure 1.

State	UIO sequence	Tail	MUIO #
v_1	$a/x\ a/x$	v_1	UIO_1^1
	$a/x\ b/x$	v_2	UIO_1^2
	$c/y\ b/x$	v_3	UIO_1^3
	$a/x\ c/y$	v_4	UIO_1^4
	$b/x\ a/x$	v_5	UIO_1^5
v_2	b/y	v_3	UIO_2
v_3	$b/x\ c/z$	v_1	UIO_3^1
	$b/x\ a/z$	v_4	UIO_3^2
v_4	$b/x\ c/y$	v_5	UIO_4
v_5	c/z	v_1	UIO_5^1
	a/z	v_4	UIO_5^2

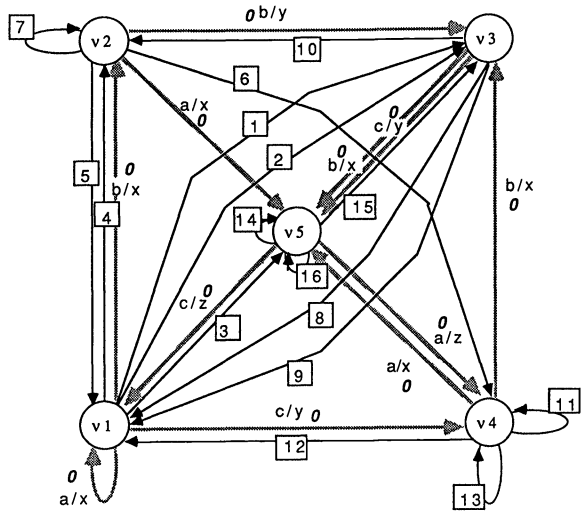


FIG. 6. *The rural symmetric augmentation G^* of the graph shown in Figure* 5. *The italic number associated with each edge corresponds to the number of times the edge appears in the rural symmetric augmentation (dotted edges represent E, solid fine edges represent Ec, and reset edges are not shown).*

graph $G_M = (V_M, E_M)$ is constructed such that $V_M \equiv \{s,t\} \cup V_X \cup V_Y$ ($V_X = \{x_1, \ldots, x_n\}$ and $V_Y = \{y_1, \ldots, y_n\}$) and $E_M \equiv E_S \cup E_T^- \cup E_T^+ \cup E^*$ ($E_S = \{(s, x_i)$: $x_i \in V_X\}, E_T^- = \{(y_i, t)$: $y_i \in V_Y\}, E_T^+ = \{(y_j, t)$: $y_j \in V_Y\}$, and $E^* = \{(x_i, y_j)$: there exists UIO_i^α such that $TAIL(UIO_i^\alpha) = v_j\}$). An edge $(y_j, t) \in E_T^-$ is denoted

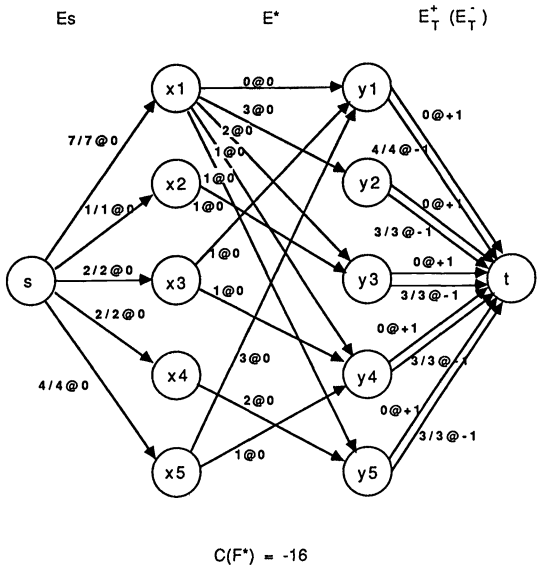


FIG. 7. *The graph G_M for the FSM of Figure* 1 *and the set of multiple UIO sequences of Table* 1. *The label on each edge represents "flow/capacity @cost" or "flow@cost" if capacity = infinity.*

as $(y_j, t)^-$ and an edge $(y_j, t) \in E_T^+$ is denoted as $(y_j, t)^+$. As an example, the graph G_M, constructed from the graph G of Figure 1 and the set of minimum-length UIO sequences of G in Table 1, is given in Figure 7.

The following conditions for cost and capacity of each edge in E_M apply:

1. each edge $(s, x_i) \in E_S$ has cost zero and capacity $\gamma(s, x_i) = d_{in}^E(v_i)$,
2. each edge $(y_j, t) \in E_T^-$ has cost -1 and capacity $\gamma(y_j, t) = d_{out}^E(v_j)$,
3. each edge $(y_j,\ t) \in E_T^+$ has cost $+1$ and infinite capacity,
4. each edge $(x_i, y_j) \in E^*$ has cost zero and infinite capacity,

where $d_{in}^E(v_i)$ and $d_{out}^E(v_i)$ are the in-degree and out-degree of a vertex v_i in G.

As the cost of each edge $e \in E_S \cup E^*$ is 0 and the cost of each edge $e \in E_T^+$ (E_T^-) is $+1$ (-1), the *cost* of the flow F on G_M is $C(F) = \sum_{(y_j,t)^+ \in E_T^+} F(y_j, t)^+ - \sum_{(y_j,t)^- \in E_T^-} F(y_j, t)^-$.

The minimum-cost maximum-flow F^* on G_M can be used to solve the *multiple UIO sequence assignment problem* by constructing a new graph G'_A such that for each $v_j \in V$, each edge $(v_k, v_j) \in E$ is assigned to exactly one UIO sequence $UIO_j^\alpha \in MUIO_j$ and each UIO_j^α in $MUIO_j$ is used exactly $F^*(x_j, y_i)$ times, where $(x_j, y_i) \in E^*$ is the edge in G_M which represents UIO_j^α. The minimum-cost maximum-flow F^* of the graph G_M is shown in Figure 7.

The following theorems were proved in [18] and are included for completeness.

THEOREM 1. *If F^* is a minimum-cost maximum-flow on G_M and G'_A is constructed from F^* using the multiple UIO sequence assignment procedure, then the cost $C(F^*) = \Delta(G'_A) - |E'|$.*

THEOREM 2. *If F^* is a minimum-cost maximum-flow on G_M, then the corresponding assignment of UIO sequences to the edges of G is such that the degree $\Delta(G'_A)$ of G'_A is minimized.*

Note that the above two theorems are valid provided that $\hat{G}^*$ is strongly connected. (This is one of the necessary conditions for the existence of an Euler tour.) Aho et al. [10] have also proved the following theorems.

THEOREM 3. *If the edge-induced subgraph $G[E_C]$ of G' is weakly connected, then*

$\hat{G}^*$ is strongly connected if single UIO sequences are used.

THEOREM 4. *If G has a reset capability, then $G[E_C]$ is weakly connected.*

THEOREM 5. *If G has at least a self-loop for every $v \in V$, then $G[E_C]$ is weakly connected.*

4. The weakly connected graph problem. The approach of [18] basically solves the UIO assignment problem, i.e., to assign to each vertex $v_j \in V$ $d_{in}^{E_C}(v_j)$ UIO sequences $UIO_j^\alpha \in MUIO$; in [18], the minimum-cost maximum-flow F^* on G_M of [10] gives an assignment of UIO sequences which has minimum $\Delta(G')$. However, the assignment problem as defined in [18] is confusing (as described in later paragraphs). This paper uses *the UIO concatenation problem* for the same purpose. The problem can be formulated as follows. The problem of UIO concatenation consists of two subproblems: the first subproblem determines the number of each UIO_j^α ($\alpha = 1, 2, \ldots, r_j$, where r_j is the number of $MUIO_j$). This is referred to as the *UIO assignment problem.* The second subproblem chooses the UIO sequence from UIO_j^α for each edge $(v_i, v_j) \in E$ and then chains the edge (v_i, v_j) with this UIO sequence. This is referred to as the *UIO chaining problem.*

In [18], the two above problems are not explicitly distinguished: the assignment problem is solved in [18] by finding a minimum-cost maximum-flow in G_M while the chaining problem is solved in [18] by choosing a UIO sequence UIO_j^α according to α in an ascending order. This implies that the weakly connected requirement in the graph may not necessarily be met (also in the presence of a reset capability in the FSM). Figure 8 shows an example. As there is more than one UIO sequence for v_3, E_{C6} can be chained with UIO_3^1 (instead of UIO_3^2), and E_{C12} can be chained with UIO_3^2 (instead of UIO_3^1) without changing the cost of the flow in Figure 7. This results in a new graph G'_A, as shown in Figure 8. However, the edge-induced subgraph $G[E_C]$ of G'_A is not weakly connected; this means that the graph $\hat{G}^*$ derived from G'_A of Figure 8 no longer will be strongly connected and the Euler tour on this $\hat{G}^*$ cannot be found except if more edges of E are added. Therefore, the total cost of the rural Chinese postman tour of Figure 8 is greater than that of Figure 5. In this example the cost will increase by 3 because at least three edges of E (i.e., E_3, E_{11}, and E_{15}) must be added to make the edge set $\{E_{C11}, E_{C12}, E_{C13}\}$ strongly connected with the other edges in E_C.

The UIO chaining problem can be described as follows: decide which UIO_j^α should be used for solving the assignment problem. If more than one UIO_j^α sequence is assigned to a vertex v_j (in this case, there must be more than one in-edge (v_i, v_j) to v_j in E), then there exist several alternatives for concatenating these edges (v_i, v_j) to UIO_j^α. For example, in Figure 9, two UIO sequences (UIO_3^1 and UIO_3^2) are assigned to v_3. This corresponds to the minimum-cost maximum-flow of Figure 7. Two edges ($E_6 = (v_2, v_3)$ and $E_{12} = (v_4, v_3)$) must be chained with two different UIO sequences. Two alternatives are possible:

1. $E_{C6} = E_6 \bullet UIO_3^1, E_{C12} = E_{12} \bullet UIO_3^2$ (this will result in the graph being not weakly connected), or

2. $E_{C6} = E_6 \bullet UIO_3^2, E_{C12} = E_{12} \bullet UIO_3^1$ (this will result in the graph being weakly connected).

Hence, different E_C can be constructed such that the edge-induced subgraph $G[E_C]$ may be weakly connected. (If the FSM has reset capabilities, this condition is valid.) The E_C with this property can be constructed as follows. The chaining subproblem can be solved by using a triple directed graph from $G = (V, E)$, which is defined as $G_T = (V_T, E_T)$, where $V_T \equiv V_W \cup V_X \cup V_Y$ (for $V_W = \{w_1, \ldots, w_n\}, V_X = \{x_1, \ldots, x_n\}, V_Y = \{y_1, \ldots, y_n\}$) and $E_T \equiv E \cup E^*$, and $E = \{(w_i, x_j)$: there exists an

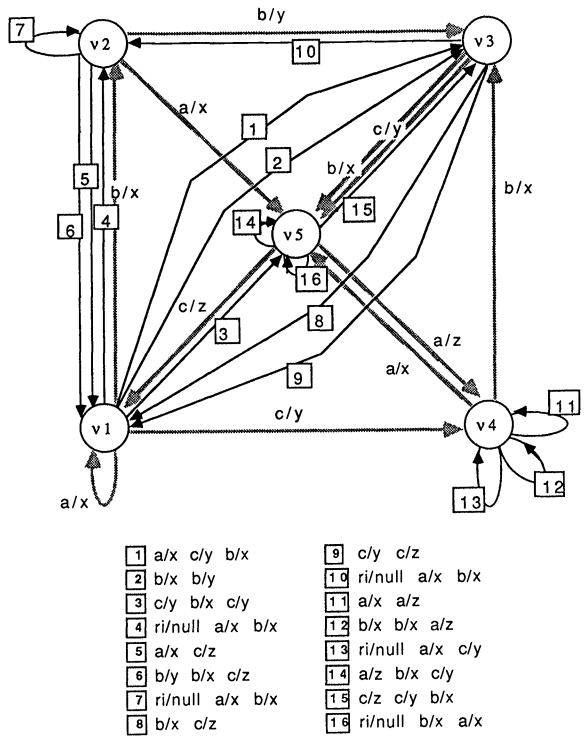


FIG. 8. *Another graph G'_A for chaining different UIOs to edge Ec6 and Ec12 of Figure* 5. *In this graph, the edge-induced subgraph $G[Ec]$ of G'_A is not weakly connected. Dotted edges represent E (reset edges are not shown). Solid fine edges represent Ec.*

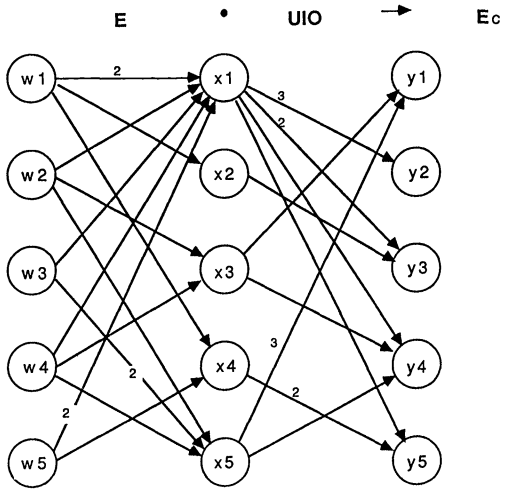


FIG. 9. *The graph G_T for the FSM of Figure* 1 *and the set of multiple UIO sequences of Table* 1. *The label on each edge represents the capacity; the capacity of an unlabeled edge is* 1.

edge (v_i, v_j) in $G\}, E^* = \{(x_i, y_j)$: there exists UIO_i^α such that $TAIL(UIO_i^\alpha) = v_j\}$.

Let each edge $(w_i, x_j) \in E$ have capacity $\gamma(w_i, x_j)$, which is equal to the number of edges (v_i, v_j), and each edge $(x_i, y_j) \in E^*$ has capacity $\gamma(x_i, y_j) = F^*(x_i, y_j)$, where F^* is the minimum-cost maximum-flow in G_M [18]. If $\gamma(x_i, y_j) = F^*(x_i, y_j) = 0$, the edge (x_i, y_j) can be omitted. An example of G_T is shown in Figure 9; this is the G_T of Figure 7.

Let the two adjacent edges, (w_i, x_k) and (x_k, y_j), be a *triple match edge* or, briefly, a trimatch edge. Any trimatch edge in G_T can construct an E_C in G'_A. It is obvious that different G'_A can be obtained if different trimatch edges are chosen. A trimatch in G_T in which each edge $e \in E_T$ appears $\gamma(e)$ times is called an *exact trimatch.*

The UIO chaining problem is, therefore, equivalent to finding an exact trimatch of the triple graph G_T such that the induced graph $G(E_C)$ is weakly connected. If all vertices $\{x_i\}$ are omitted, the exact trimatch is reduced to an exact binary match (bimatch). This bimatch can be viewed also as a group of two level trees $\{T_j\}$; the root of T_j is y_j which corresponds to vertex v_j by definition. Equivalently, all leaves are $\{v_i\}$ which correspond to the same vertices. If every bimatch edge (i.e., a tree edge) constructs an edge (v_i, v_j) in E_C, all vertices in tree T_j are weakly connected. Note that w_i and y_i in G_T correspond to the same vertex v_i in G'. The following lemma can be easily proved.

LEMMA 1. *If two trees have in common at least one vertex (including the root) in an exact bimatch, then all vertices of the two trees in G'_A are weakly connected.*

The following theorem directly relates chaining to weak connectivity.

THEOREM 6. *If an FSM has reset capability, then there always exists a chaining of every edge in E and the UIO used in F^* such that the edge-induced subgraph $G[E_C]$ of G'_A is weakly connected.*

Proof. If the FSM has reset capability, then there exists an edge $(v_i, v_1; L_k)$ from each vertex v_i $(i = 1, \ldots, n)$ to vertex v_1 (representing the reset state S_1 in G). In [10], it has been shown that if a single UIO of v_1 is used, $G[E_C]$ is a spanning subgraph of G' and therefore is weakly connected.

If exactly two UIO sequences of v_1 (denoted as UIO_1^1 and UIO_1^2) are used in F^* and $TAIL(UIO_1^1) = y_{\phi_1}$ and $TAIL(UIO_1^2) = y_{\phi_2}$, then it is possible to find a match (w_{ϕ_2}, y_{ϕ_1}). This match must exist for v_{ϕ_2} because it has a reset edge (v_{ϕ_2}, v_1) and $TAIL(UIO_1^1) = y_{\phi_1}$. UIO_1^1 or UIO_1^2 can be chained randomly with any other (w_i, x_1). The two trees (denoted as T_{ϕ_1} and T_{ϕ_2}) have the following properties:

1. Both trees include all vertices of G'_A.
2. The two trees have a common vertex v_{ϕ_2}.

Lemma 1 has shown that all vertices in both trees (as well as all vertices in G'_A) are weakly connected. Then, $G[E_C]$ is weakly connected also. If three or more UIO sequences (i.e., $UIO_1^1, UIO_1^2, UIO_1^3$) exist in F^*, their $TAIL$s are y_{ϕ_1}, y_{ϕ_2}, and y_{ϕ_3}, respectively. It is then possible to find matches (w_{ϕ_2}, y_{ϕ_1}) and (w_{ϕ_3}, y_{ϕ_2}), while other matches can be chained randomly. T_{ϕ_1} and T_{ϕ_2} have the vertex v_{ϕ_2} in common, while T_{ϕ_2} and T_{ϕ_3} have the vertex v_{ϕ_3} in common. Hence, all vertices in the trees (inclusive of G'_A) are weakly connected and $G[E_C]$ is also weakly connected. This process can be iteratively continued, thus proving the theorem. □

Consider as an example Figure 9. The reset state is $S_1; S_1$ has three UIO sequences in which $\gamma \neq 0, TAIL(UIO_1^1) = y_2, TAIL(UIO_1^2) = y_3$, and $TAIL(UIO_1^4) = y_5$. It is possible to find a match (w_3, y_2) (passing through x_1) and a match (w_5, y_3) (also passing through x_1). All other matches can be constructed randomly. The $G[E_C]$ thus constructed is weakly connected.

Theorem 6 not only provides a sufficient condition for $G[E_C]$ to be weakly connected (namely, that G has a reset capability) but also presents an algorithm for

constructing a weakly connected $G[E_C]$. This algorithm can be directly derived using the proof of Theorem 6. So, Theorem 6 gives a sufficient condition for the test sequence generated by the touring method.

5. Multiple *UIO* tour length minimization. In [18], the assignment of edges to multiple UIO sequences has been discussed. The approach of [18] minimizes $\Delta(G')$, but this does not necessarily minimize the length of the tour. In more general terms, it does not minimize the cost of the tour. In this section, an algorithm for finding the minimized tour cost is presented. The proposed approach is based on a multistage flow graph and on the assumption that the UIO sequences for any given state all have minimum cost. For multiple UIO sequences of a state, the following conditions apply for the minimum cost: if there is more than one UIO sequence with the same *TAIL* state, the chosen UIO sequence is the UIO of minimum cost; however, UIO sequences of a given state with different *TAIL* states may have different cost. Consider initially the following lemma.

LEMMA 2. *If $\Delta(G')$ of a minimum-cost maximum-flow F^* on G_M is equal to 0, then there is no other assignment of edges to the UIO sequence which has a tour of smaller cost.*

Proof. $\Delta(G') = 0$; hence, the rural symmetric augmentation of $\hat{G}^*$ can be constructed from G' without using any edge in E; i.e., $\hat{G}^* = (\hat{V}^*, \hat{E}^*)$, where $\hat{E}^* = E_C$. E_C is constructed by chaining an appropriate UIO to each E from the minimum-cost maximum-flow F^* on G_M. The cost of each multiple UIO sequence for any state is the minimum. Hence, the cost of the tour is minimum. □

Lemma 2 does not necessarily hold if $\Delta(G') \neq 0$; in some circumstances, the minimum-cost maximum-flow may lead to the replication of a costly edge of E for constructing the symmetric graph $\hat{G}^*$. In a later paragraph, this problem will be illustrated in detail.

To obtain the minimum-cost tour, a directed graph $G_Q = (V_Q, E_Q)$ is constructed from G as follows. $V_Q = \{s,t\} \cup V_W \cup V_X \cup V_Y \cup V_Z$, where s and t are the source and target of $G_F, V_W = \{w_1, \ldots, w_n\}, V_X = \{x_1, \ldots, x_n\}, V_Y = \{y_1, \ldots, y_n\}, V_Z = \{z_1, \ldots, z_n\}$, and

$$E_Q = E_S \cup E_T \cup E \cup E^* \cup E_B,$$

where $E_S = \{(s, x_i)\colon\ x_i \in V_X\}, E_T = \{(z_k, t)\colon\ z_k \in V_Z\}, E = \{(w_h, x_i)\colon$ there is an edge (h,i) in the graph $G\}$, $E^* = \{(x_i, y_j)\colon$ there exists UIO_i^α such that $TAIL(UIO_i^\alpha) = v_j\}$, and $E_B = \{(y_j, z_k)\colon$ a minimum-cost transition from v_j to v_k if it exists$\}$.

Note that unlike E^*, an edge (y_j, z_j) in E_B represents a *null* edge (not a self-loop of v_j). Every edge in E_Q has a cost and a capacity; let each edge $(s, w_h) \in E_S$ have zero cost and capacity $\gamma(s, w_h) = d^E_{out}(v_h)$; let each edge $(z_k, t) \in E_T$ have zero cost and capacity $\gamma(z_k, t) = d^E_{out}(v_k)$; let each edge $(w_h, x_i) \in E$ have the same cost as in G and capacity one; let each edge $(x_i, y_j) \in E^*$ have cost = the cost of UIO_i^α and infinite capacity; let each edge $(y_j, z_k) \in E_B$ have infinite capacity and a cost defined as $\text{cost}(y_j, z_k)$ = cost of the shortest transition (bridge sequence) from v_j to v_k if $j \neq k$, $\text{cost}(y_j, z_k) = 0$ if $j = k$.

A flow F on G_Q is a function which satisfies the following conditions:

1. for each vertex $w_h \in V_W$,

$$F(s, w_h) = \sum_{(w_h, x_i) \in E} F(w_h, x_i);$$

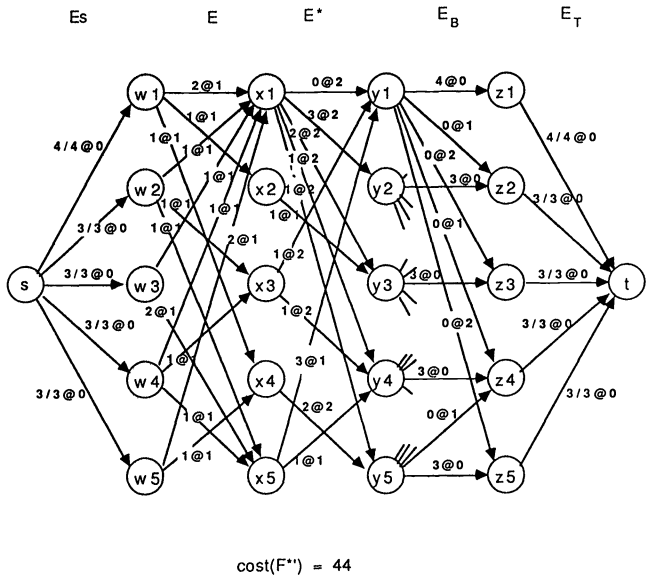


FIG. 10. *The graph G_Q for the FSM of Figure* 1 *and the set of multiple UIO sequences of Table* 1. *The label on each edge represents "flow/capacity@cost" or "flow@cost" if capacity = infinity.*

2. for each vertex $x_i \in V_X$,

$$\sum_{(w_h,x_i)\in E} F(w_h,x_i) = \sum_{(x_i,y_j)\in E^*} F(x_i,y_j);$$

3. for each vertex $y_j \in V_Y$,

$$\sum_{(x_i,y_j)\in E^*} F(x_i,y_j) = \sum_{(y_j,z_k)\in E_B} F(y_j,z_k);$$

4. for each vertex $z_k \in V_Z$,

$$F(z_k,t) = \sum_{(y_j,z_k)\in E_B} F(y_j,z_k);$$

5. for each edge $(s,x_i) \in E_S, F(s,x_i) \le \gamma(s,x_i)$; and,
6. for each edge $(z_k,t) \in E_T, F(z_k,t) \le \gamma(z_k,t)$.

The cost of the flow F is

$$C(F) = \sum_{(w_h,x_i)\in E} F(w_h,x_i)\text{cost}(w_h,x_i) + \sum_{(x_i,y_j)\in E^*} (x_i,y_j)\text{cost}(x_i,y_j) + \sum_{(y_j,z_k)\in E_B} F(y_j,z_k)\text{cost}(y_j,z_k).$$

The graph G_Q for the FSM of Figure 1 and the set of multiple UIO sequences of Table 2 is shown in Figure 10. For simplicity let the cost of each edge be one. In Figure 10, $\text{cost}(y_1,z_1) = 0$. Because a directed edge $(v_1,v_2;b/x)$ exists, $\text{cost}(y_1,z_2) = 1$. $\text{cost}(y_1,z_3) = 2$ because the shortest transition from v_1 to v_3 consists of two adjacent edges, $(v_1,v_2;b/x)$ and $(v_2,v_3;b/y)$. Note that Figure 10 differs from Figure 7 (taken from [18]) by the addition of E, E_B, V_W, and V_Z, while E_T^- is cancelled. This permits that the volume of flow into z_k is always equal to $\gamma(z_k,t)$ for any maximum flow on G_Q. The main difference between G_Q and G_M is that G_M does not explicitly take into account a bridge sequence and the cost of the edges, UIOs, and bridge sequences.

If the minimum-cost maximum-flow F^* on G_M has $\Delta(G') > 0$, Shen, Lombardi, and Dahbura [18] have shown that in this case, $\Delta(G')$ is minimum. However, the flow $F^{*\prime}$ on G_Q derived from the F^* on G_M cannot be guaranteed to be a minimum-cost flow. This occurs because $\Delta(G')$ guarantees only that the number of all streams of flow $F(y_j, z_k)\,(j \neq k)$ is minimum; in other words, minimum $\Delta(G')$ means that the number of nonnull bridge sequences is minimum, but the total length of the bridge sequences is not necessarily minimum.

Consider a minimum-cost maximum-flow F^* on G_M with $\Delta(G') = 1$ which, for example, results in one bridge sequence $B_{1,2}$ to be added; this is equivalent to $F(y_1, z_2) = 1$ in G_Q and another maximum-flow F_2 on G_M with $\Delta(G') = 2$ (which results in $F(y_1, z_3) = 1$ and $F(y_2, z_3) = 1$). Assume in this example $\text{cost}(y_1, z_2) = 5$, $\text{cost}(y_1, z_3) = 1$, and $\text{cost}(y_2, z_3) = 2$. In the latter case, although $\Delta(G')$ is not minimum, the tour length of F_2 is less than the one of F^* because it transforms the two flows into flows on G_Q. Hence, we have $\text{cost}(y_1, z_3) + \text{cost}(y_2, z_3) < \text{cost}(y_1, z_2)$.

In G_Q, each stream of flow from s to t consists of the edges $(s, w_h), (w_h, x_i)$, $(x_i, y_j), (y_j, z_k)$, and (z_k, t) in G_Q. The edge (w_h, x_i) maps an edge in G from vertex v_h to v_i. The edge (x_i, y_j) maps one of the UIO sequences in the $MUIO$ set (i.e., the multiple UIO set) for state v_i; the edge (y_j, z_k) maps a bridge sequence of G from v_j to v_k. Finally, the edge (s, x_h) and the edge (z_k, t) represent the presence of in-edges of v_h in G and the presence of out-edges of v_k in G, respectively. If a maximum-flow F in G_Q is found, a generalized graph G'' of G can be constructed on F.

Let this generalized graph be $G'' = (V'', E'')$ such that $V'' = V$ and $E'' = E_C \cup E_B$. For each stream of flow $f \in F$ from s to t, $f = (s, w_h, x_i, y_j, z_k, t)$. There are two edges e_C and e_B in G'', where $e_C = (v_h, v_i) \bullet (v_i, v_j), (v_h, v_i) \in E, (v_i, v_j) \in UIO_i^\alpha$, and $TAIL(UIO_i^\alpha) = v_j; e_B = (v_j, v_k)$, and e_B is the shortest bridge sequence from v_j to v_k. Note that the connectivity problem can be solved using the chaining problem described in §4.

LEMMA 3. *If F is a maximum flow on G_Q and G'' is constructed from F as described previously (where G is strongly connected and $G[E_C]$ is weakly connected), then G'' is a symmetric graph.*

Proof. By definition, $F(w_h, x_i)$ is the edge (v_h, v_i) in G, $F(x_i, y_j)$ is a UIO_i^α, and $F(y_j, z_k)$ is a series of edges $(v_j, v_{j1})(v_{j1}, v_{j2}) \cdots (v_{jr}, v_k)$ (where these edges may be *null* edges). As F is a maximum flow, every edge in E has been assigned a UIO in F.

According to the conditions given above, the flow is saturated and the in-degree for each vertex v_k in G'' is $\sum_{(x_i, y_k) \in E^*} F(x_i, y_k) + \sum_{(y_j, z_k) \in E_B} F(y_j, z_k)$ and the out-degree is $\sum_{(y_k, z_l) \in E_B} F(y_k, z_l) + F(z_k, t)$. As F is a maximum flow, then $\sum_{(y_j, z_k) \in E_B} F(y_j, z_k) = F(z_k, t)$ and $\sum_{(x_i, y_k) \in E^*} F(x_i, y_k) = \sum_{(y_k, z_l) \in E_B} F(y_k, z_l)$. This implies that the in-degree of v_k is E'' and is equal to the out-degree of v_k. Thus, G'' is symmetric. □

LEMMA 4. *If a directed graph is symmetric and weakly connected, then this directed graph is also strongly connected.*

Proof. Assume that the directed graph $G = (V, E)$ is not strongly connected but is weakly connected. Define the one-way from v_i to v_j as the directed edge (i, j) provided there is no path from v_j to v_i in G. By the weakly connected assumption, at least one one-way edge (i, j) exists. List all the one-way edges in G; there must exist a one-way edge (i, j) such that $v_j \in S_j$ where S_j is a proper subset of V, all vertices in S_j are strongly connected, and no one-way edge from $v \in S_j$ to any other vertex exists. In the final vertex subset S_j, there are two kinds of edges connected to $v \in S_j$: the edges between two vertices v_m and v_n where v_m and $v_n \in S_j$ and the edges from

v_k to v_l where $v_l \in S_j$ and v_k is not in S_j. Therefore, the index of S_j is as follows.

$$\xi(S_j) = \sum_{v_r \in S_j} d_{out}(v_r) - \sum_{v_r \in S_j} d_{in}(v_r) = \text{ number of edges } (v_m, v_n)$$
$$- (\text{number of edges } (v_m, v_n) + \text{number of edges } (v_k, v_l)) < 0.$$

However, by the condition of Lemma 4,

$$\xi(S_j) = \sum_{v_r \in S_j} \xi(v_r) = \sum 0 = 0.$$

This is a contradiction and Lemma 4 is proved. □

The applicability of Lemma 4 is general; i.e., it is not restricted to UIO sequences. If the above conditions are satisfied, then it is possible to chain each edge of E to a suitable UIO sequence and make the directed graph G'' weakly connected. By Lemma 3, the directed graph G'' from a maximum flow is symmetric. Therefore, there exists an Euler tour of G''.

LEMMA 5. *Any tour of a generalized graph G'' is a test sequence for the directed graph G.*

Proof. Consider $G'' = (V'', E'')$ and $E'' = E_C \cup E_B$. Every edge in E_C is obtained by chaining an edge in E and the corresponding UIO sequence. Thus, it is a test subsequence. The tour of G'' traverses all the edges in E_C; i.e., it includes all test subsequences. Hence, by definition it is a test sequence of G. □

LEMMA 6. *Let the cost of the maximum flow F on G_Q be* cost(F) *and the cost of the Euler tour of G'' be $C(G'')$; if G'' is strongly connected, then $C(G'') =$* cost(F).

Proof. By a previous definition, a flow from s to t on G_Q corresponds to two edges in G'', i.e., $e_C \in E_C$ and $e_B \in E_B$. Hence, $C(G'') = \sum_{f' \in F}$ cost(f'), where $f' = f'(s,t)$ is a single stream of flow from s to t and cost(s,t) = cost(s, w_h) + cost(w_h, x_i) + cost(x_i, y_j) + cost(y_j, z_k) + cost(z_k, t), but cost$(s, w_h) = 0$ and cost$(z_k, t) = 0$. Thus, $\sum$ cost$(s,t) = \sum$ (cost(w_h, x_i) + cost(x_i, y_j) + cost(y_j, z_k)) = cost(F). □

The following theorem therefore holds.

THEOREM 7. *Assume F^* is a minimum-cost maximum-flow on G_Q and G'' is constructed from F^*. If G'' is weakly connected, then there exists at least an Euler tour of G'' which has the minimum cost among all the rural Chinese postman tours of G'.*

Proof. Lemma 4 and the condition of a weakly connected graph guarantee the existence of an Euler tour in G''. Assume that there exists a rural Chinese postman tour of G' and let it be denoted by T_1. The total cost of this tour C_1 is less than the cost C of G''. A $\hat{G}_1^*$ can be constructed from T_1. Then, it is possible to construct a flow F_1 from $\hat{G}_1^*$ such that cost$(F_1) <$ cost(F^*). This yields a contradiction on the flow F^* being the minimum flow on G_Q. □

6. Conclusions. This paper has presented the solution to two problems left open in [18] for protocol verification and validation by multiple UIO sequences. The problems are the multiple UIO tour length minimization and the weakly connected graph problems and they arise in the characterization of conformance testing of protocols using the multiple UIO technique of [18]. It is proved that the solution of these problems can be achieved by modifying the assignment graph. The theory behind these conditions has been fully analyzed and proved.

It should be pointed out that the UIO method does not always guarantee 100% probability of detecting faults for conformance testing. In most cases, nearly 100%

fault detection is possible. The interested reader should refer to [21] for a solution to this problem.

The following problem is not addressed in this paper: by overlapping test subsequences, a shorter test sequence may be generated [22, 23]. This problem can be solved using a different flow method; due to lack of space this aspect is not discussed in this paper. However, such a test sequence with overlapped test subsequences does not necessarily have the same fault detection capabilities of the test sequence generated with no overlaps [23, 24]. (In most cases, the test sequence with overlaps has the lower fault coverage.)

Appendix A. The following graphs are used in the paper.

1. $G = (V, E)$: state transition graph of the FSM corresponding to the protocol specifications.

2. $G' = (V', E')$: directed graph constructed from G where $V' \equiv V$ and $E' \equiv E_E \cup E_C$. Each edge in E_C is the test subsequence for an edge in E. If a tour can traverse all edges in E_C, then this tour is a test sequence of the FSM.

3. $G_F = (V_F, E_F)$: flow graph derived from G'. The capacity of each edge in E_F depends on the index of the corresponding vertex v_i in G'. In G_F there are a source (s) and target (t). A minimum-cost maximum-flow F from s to t can be found which can be used to construct the symmetric directed graph $\hat{G}^*$.

4. $\hat{G}^* = (\hat{V}^*, \hat{E}^*)$: symmetric directed graph constructed from G' by replicating χ times each edge $(v_i, v_j) \in E$. χ may be 0, 1, or more and it can be obtained from the maximum flow F of G_F. If $\hat{G}^*$ is strongly connected, the Euler tour of $\hat{G}^*$ is a test sequence.

5. $G_M = (V_M, E_M)$: directed graph constructed from G and the $MUIO$s of all vertices of G. A source (s) and a target (t) are added such that a flow from s to t exists.

6. $G'_A = (V'_A, E'_A)$: alternative directed graph similar to G' but using $MUIO$s instead of UIOs. The assignment of $MUIO$ is based on the minimum-cost maximum-flow F on G_M. Similarly, the directed graph G_F and then the symmetric directed graph $\hat{G}^*$ can be also constructed from G'_A.

7. $G_T = (V_T, E_T)$: a directed graph obtained from G and the $MUIO$ sequences. The capacity of each edge depends on the minimum-cost maximum-flow F on G_M. G_T is used to solve the chaining problem.

8. $G_Q = (V_Q, E_Q)$: constructed from G and $MUIO$s as well as the bridge subsequences of G. In G_Q, a source (s) and a target (t) are added such that a flow from s to t can be found.

9. $G'' = (V'', E'')$: directed graph constructed from the flow F of G_Q. Each stream of flow in G_Q maps two edges (e_C and e_B) on G_Q. If F is a minimum-cost maximum-flow, G'' is always symmetric. The Euler tour of G'' is the optimized test sequence of the FSM.

REFERENCES

[1] G. V. Bochmann and C. A. Sunshine, *A survey of formal methods*, in Computer Networks and Protocols, P. E. Green, ed., Plenum Press, New York, 1983, pp. 561–578.

[2] F. C. Hennie, *Fault detection experiments for sequential circuits*, in Proc. 5th Ann. Symp. Switch. Theory and Logical Design, Princeton, NJ, 1964, pp. 95–110.

[3] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw–Hill, New York, 1978.

[4] G. Gonenc, *A method for the design of detection experiments*, IEEE Trans. Comput., 19 (1970), pp. 551–558.

[5] S. NAITO AND M. TSUNOYAMA, *Fault detection for sequential machines by transition tours,* in Proc. 11th IEEE Fault Tolerant Comput. Symp., Portland, ME, IEEE Computer Soc. Press, Washington, 1981, pp. 238–243.

[6] K. K. SABNANI AND A. T. DAHBURA, *A protocol test generation procedure,* Computer Networks, 15 (1988), pp. 285–297.

[7] M. U. UYAR AND A. T. DAHBURA, *Optimal test sequence generation for protocols: The Chinese postman algorithm applied to Q.931,* in Proc. IEEE Global Telecommunications Conference, Houston, TX, 1986, pp. 68–72.

[8] D. SIDHU AND T. LEUNG, *Fault coverage of protocol test methods,* in Proc. IEEE International Conference on Computer Communications, New Orleans, LA, 1988, pp. 80–85.

[9] B. WANG AND D. HUTCHINSON, *Protocol testing techniques,* Computer Communications, 10 (1987), pp. 79–87.

[10] A. V. AHO, A. T. DAHBURA, D. LEE, AND M. U. UYAR, *An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours,* in Protocol Specification, Testing, and Verification, VIII, S. Aggarwal and K. K. Sabnani, eds., Elsevier–North Holland, Amsterdam, 1988, pp. 75–86.

[11] J. A. BONDY AND U. S. R. MURTY, *Graph Theory with Applications*, Elsevier–North Holland, Amsterdam, 1976.

[12] M.-K. KUAN, *Graphic programming using odd or even points,* Chinese J. Math., 1 (1962), pp. 273–277.

[13] J. EDMONDS AND E. L. JOHNSON, *Matching, Euler tours and the Chinese postman,* Math. Programming, 5 (1973), pp. 88–124.

[14] R. E. TARJAN, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.

[15] *International Telegraph and Telephone Consultative Committee recommendation* X.25, in International Telegraph and Telephone Consultative Committee Orange Book, Vol. 8, Public data networks, 1977.

[16] A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms,* Addison–Wesley, Reading, MA, 1974.

[17] D. SIDHU AND T.-K. LEUNG, *Formal Methods for Protocol Testing: A Detailed Study,* Tech. report 86-23, Iowa State University, Ames, IA, 1986.

[18] Y.-N. SHEN, F. LOMBARDI, AND A. T. DAHBURA, *Protocol conformance testing by multiple UIO sequences,* in Protocol Specification, Testing and Verification, IX, E. Brinksma, G. Scollo, and C. A. Vissers, eds., Elsevier, New York, 1990, pp. 131–143.

[19] A. T. DAHBURA, private communication, AT&T Bell Labs, Murray Hill, NJ, 1990.

[20] M. CHEN, private communication, IBM, Yorktown Heights, NY, 1990.

[21] F. LOMBARDI AND Y.-N. SHEN, *Evaluation and improvement of fault coverage of conformance testing by UIO sequences,* IEEE Trans. Comm., 40 (1992), pp. 1288–1293.

[22] M.-S. CHEN, Y. CHOI, AND A. KERSHENBAUM, *Approaches utilizing segment overlap to minimize test sequence,* in International Federation of Information Processing 10th Int. Symp. on Prot. Spec., Test. and Verif., Orlando, FL, 1990, pp. 67–84.

[23] Y.-N. SHEN, X. SUN, F. LOMBARDI, AND D. SCIUTO, *Protocol conformance testing by discriminating UIO sequences,* in Protocol Spec., Testing and Verif. XI, Stockholm, Sweden, B. Jonsson, B. Pehrson, and J. Parrow, eds., North Holland, Amsterdam, 1992, pp. 349–364.

[24] X. SUN, Y.-N. SHEN, AND F. LOMBARDI, *On the verification and validation of protocols with high fault coverage using UIO sequences,* in Proc. 11th IEEE Symp. on Reliable Distributed Systems, Houston, TX, 1992, pp. 196–203.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 529–544, November 1996

002

MULTIPARTITION SERIES*

DAVID G. WAGNER†

Abstract. We investigate a class of generating series which enumerate multi-analogues of set partitions with very general weights and constraints imposed, and develop some of the relevant theory. The weights and constraints we consider are embodied in the definition of a "system," which includes weighted multiset systems as a simple special case. Three topics are discussed. First, we derive a composition formula valid for all systems, which specializes to composition formulas for familiar combinatorial structures in many cases. Second, we extend the Heilmann–Lieb theorem on matching polynomials to a similar statement valid for more general factors of multigraphs. Finally, we introduce a multi-analogue of the order polynomial of a labelled poset, and by applying our general composition theorem give a formula for the effect of composition of labelled posets on their E-polynomials.

Key words. matching polynomials, set systems, order polynomials

AMS subject classifications. 05A15, 30C15, 26C10

Introduction. Rook polynomials and matching polynomials have attracted much interest since their introduction in the 1940s and 1970s, respectively [12, 11]. In part this is due to the fact that their coefficients are analogues of Stirling numbers of the second kind and retain many properties of these numbers in a more general setting. Another source of interest in these and related polynomials is their connection with certain models in statistical mechanics (e.g., the Ising model and Potts models [1], models of adhesion of dimers [11], and models of π-electron bonding in aromatic molecules [8]). The location of zeros of these polynomials translates into information about phase transitions or energy spectra of the physical systems being modelled. Harper [10] showed that the "Stirling polynomials" $\sum_k S(n,k)t^k$ have only real zeros. Heilmann and Lieb [11] proved an analogous result for the matching polynomial of any finite graph. As a special case of one of our main results (Theorem 3.1) we obtain an appealing generalization of the Heilmann–Lieb theorem to more general factors of multigraphs (Theorem 3.3). As is well known (Theorem (51) of [9]), the fact that a polynomial $\sum_{i=0}^{d} a_i t^i$ has only real zeros implies that the sequence $\{a_i\}_0^d$ is logarithmically concave; that is, that $a_i^2 \geq a_{i-1}a_{i+1}$ for all $i = 1, \ldots, d-1$. When all the a_i are nonnnegative, it also implies that $\{a_i\}_0^d$ is unimodal; that is, there is an index k such that $a_0 \leq \cdots \leq a_k \geq \cdots \geq a_d$. Theorem 3.3 therefore implies many combinatorially interesting inequalities.

In fact, the research reported here began with Theorem 3.3 and grew out of an effort to extend it in the same way that §4 of [21] extends the Heilmann–Lieb theorem. In order to develop a definition of the composition of objects which was general enough to cover the case of weighted multigraphs, we found it convenient to introduce the concept of a "system." This abstraction is the focus of §1 and loosely can be thought of as a weighted collection of multisets with specified "interference" among its members. The related "multipartition series" is a generating series which encodes the information about a system which is relevant to the questions we have in

* Received by the editors October 3, 1994; accepted for publication (in revised form) October 10, 1995. This research was supported by National Science and Engineering Research Council of Canada operating grant OGP0105392.

† Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1 (dgwagner@math.uwaterloo.ca).

mind. A slight variation of this definition results in a multi-analogue of the chromatic polynomials of graph theory and the partition functions of statistical mechanics.

In §2 we define compositions of systems and prove a composition formula for the multipartition series of the systems involved (Theorem 2.6). This is an extensive generalization of Theorem 4.1 of [21]. In many special cases the composition of systems defined here reduces to the composition of combinatorially defined objects, and the resulting formula for the composite multipartition series yields an interesting combinatorial identity. One new instance of this is discussed in §4.

The content of §3 is a generalization of the Heilmann–Lieb theorem which gives a large class of systems for which the multipartition series satisfies a strong real-rootedness condition (Theorem 3.1). Several consequences of this are also discussed.

Finally, in §4, we show how order polynomials of labelled posets can be viewed in the context of multipartition series. We interpret the composition formula of §2 in terms of composite labelled posets, resulting in a composition formula for the "E-polynomials" of labelled posets. The formula involves an interesting multi-analogue of the E-polynomial; further investigation of these series, however, is deferred to another paper [23].

We introduce notation as we proceed, with the exception of the following more or less standard conventions. The symbol $:=$ is used for equality by definition. The cardinality of a set U is denoted by $\#U$. For $u \in U$ we write $U \smallsetminus u$ instead of $U \smallsetminus \{u\}$. For a function $f : U \to \mathbf{N}$ defined on a set U, let $f! := \prod_{u \in U} f(u)!$ and $|f| := \sum_{u \in U} f(u)$. We say that f is *finite* when $|f|$ is finite. We denote the function from U to $\mathbf{N}$ which is identically zero by 0 or by $\emptyset$, and the all-ones function is denoted by 1 or by U. Given a set $\mathbf{X} := \{X_u : u \in U\}$ of pairwise commuting indeterminates indexed by U, for a finite $f : U \to \mathbf{N}$ let $\mathbf{X}^f := \prod_{u \in U} X_u^{f(u)}$, and for a finite subset $S \subseteq U$ let $\mathbf{X}^S := \prod_{u \in S} X_u$. If $Q(\mathbf{X}) \in R[[\mathbf{X}]]$ for some ring R, then we denote by $[\mathbf{X}^f]Q(\mathbf{X})$ the coefficient of $\mathbf{X}^f$ in $Q(\mathbf{X})$. Given a set U, a (finite) *multiset on* U is any finite function $S : U \to \mathbf{N}$. The interpretation is that for each $u \in U$, $S(u)$ is the multiplicity of u as an element of S. Thus finite sets may be usefully confused with their indicator functions. We also write $\#S$ for $|S|$ and $u \in S$ for $S(u) > 0$. The set of all finite multisets on U is denoted by $\mathcal{M}(U)$. The set of all finite nonempty multisets on U is $\mathcal{M}^+(U) := \mathcal{M}(U) \smallsetminus 0$. We denote by $\mathcal{P}(U)$ the set of all finite subsets of U, and $\mathcal{P}^+(U) := \mathcal{P}(U) \smallsetminus 0$. For $S, T \in \mathcal{M}(U)$ the notation $S \subseteq T$ means that $S(u) \leq T(u)$ for all $u \in U$. All multisets we consider are finite.

1. Systems. By a *vertex-set* we mean a finite or countable set V, the elements of which are called *vertices*. A *multipartition* π of V is any multiset of nonempty multisets on V; that is, $\pi \in \mathcal{M}(\mathcal{M}^+(V))$. The elements of a multipartition are called *blocks*. Given a multipartition π of V we define the *elevation* of π to be the multiset $\mathrm{el}(\pi) : V \to \mathbf{N}$ given for $v \in V$ by

$$\mathrm{el}(\pi, v) \;:=\; \mathrm{el}(\pi)(v) := \sum_{S \in \mathcal{M}^+(V)} \pi(S) \cdot S(v). \tag{1}$$

An *ordered multipartition* is a finite sequence $(S_1, \ldots, S_m)$ of not necessarily distinct nonempty multisets on V. The set of all ordered multipartitions of V is $\mathcal{O}(V) := \bigsqcup\{\mathcal{M}^+(V)^m : m \geq 0\}$. The *base* of an ordered multipartition $(S_1, \ldots, S_m)$ is the multipartition π of V such that $\pi(U) := \#\{i : S_i = U\}$ for all $U \in \mathcal{M}^+(V)$; we denote the base of σ by $\mathrm{base}(\sigma)$. The number of ordered multipartitions with base equal to π is the multinomial coefficient $(\#\pi)!/\pi!$. Any concept involving a multipartition is

extended to one involving an ordered multipartition by using the base of the ordered multipartition. For example, if $\sigma = (S_1, \ldots, S_m)$ then we let $\sigma(U) := \#\{i : S_i = U\}$ for all $U \in \mathcal{M}^+(V)$. The concatenation of two ordered multipartitions σ and τ is an ordered multipartition, denoted by $\sigma \oplus \tau$.

Now fix a commutative $\mathbf{Q}$-algebra W, called the *weight ring*. A *structure* is a pair $(V, \mathcal{C})$ in which V is a vertex-set and $\mathcal{C}$ is a function $\mathcal{C} : \mathcal{O}(V) \to W$. This is clearly a very general definition but, as will be seen, the generality is useful and several special cases are of considerable interest. Let t and $\mathbf{X} := \{X_v : v \in V\}$ be pairwise commuting indeterminates algebraically independent over W. The *multipartition series* of $(V, \mathcal{C})$ is

$$P^{\mathcal{C}}(t, \mathbf{X}) := \sum_{\sigma \in \mathcal{O}(V)} \mathcal{C}(\sigma) \frac{t^{\#\sigma}}{(\#\sigma)!} \mathbf{X}^{\mathrm{el}(\sigma)}, \tag{2}$$

and the *polychromatic series* of $(V, \mathcal{C})$ is

$$Z^{\mathcal{C}}(t, \mathbf{X}) := \sum_{\sigma \in \mathcal{O}(V)} \mathcal{C}(\sigma) \binom{t}{\#\sigma} \mathbf{X}^{\mathrm{el}(\sigma)} \tag{3}$$

(this terminology is motivated by Example 1.3). Notice that for any finite $f : V \to \mathbf{N}$, both $[\mathbf{X}^f] P^{\mathcal{C}}(t, \mathbf{X})$ and $[\mathbf{X}^f] Z^{\mathcal{C}}(t, \mathbf{X})$ are polynomials in $W[t]$ of degree at most $|f|$.

One natural condition to impose on a structure $(V, \mathcal{C})$ is that whenever $\sigma, \sigma' \in \mathcal{O}(V)$ are such that $\mathrm{base}(\sigma) = \mathrm{base}(\sigma')$, then $\mathcal{C}(\sigma) = \mathcal{C}(\sigma')$. Such a structure will be called *basic*. Thus $\mathcal{B}(\pi)$ is well defined when $(V, \mathcal{B})$ is a basic structure and π is a multipartition of V, and it follows that

$$P^{\mathcal{B}}(t, \mathbf{X}) := \sum_{\pi \in \mathcal{M}(\mathcal{M}^+(V))} \mathcal{B}(\pi) \frac{t^{\#\pi}}{\pi!} \mathbf{X}^{\mathrm{el}(\pi)}. \tag{4}$$

Until §4, all of our examples of structures are basic.

Notice that if $\mathrm{el}(\pi) \in \mathcal{P}(V)$ then $\pi! = 1$, with the following consequence.

PROPOSITION 1.1. *Let $(V, \mathcal{B})$ be a basic structure, and let $U \in \mathcal{P}^+(V)$. Then $[\mathbf{X}^U] P^{\mathcal{B}}(t, \mathbf{X}) = \sum_{\pi} \mathcal{B}(\pi) t^{\#\pi}$ where the summation is over all set partitions π of U.*

By specializing to (the indicator structure of) a finite set system, the polynomial of Proposition 1.1 becomes the partition polynomial discussed in [21]. Further specializations result in matching, rook, and σ-polynomials [3, 4, 6, 7, 8, 11, 12].

A *multiset system* is a pair $(V, \mathcal{F})$ in which V is a vertex-set and $\mathcal{F}$ is a subset of $\mathcal{M}^+(V)$. When $\mathcal{F}$ is a subset of $\mathcal{P}^+(V)$ we say that $\mathcal{F}$ is a *set system* on V. A multipartition π of V *respects* $\mathcal{F}$ if and only if for all $S \in \mathcal{M}^+(V)$, if $\pi(S) > 0$ then $S \in \mathcal{F}$. A structure $(V, \mathcal{C})$ is *supported on* $(V, \mathcal{F})$ when it satisfies the condition that if $\mathcal{C}(\sigma) \neq 0$ then σ respects $\mathcal{F}$, for all $\sigma \in \mathcal{O}(V)$. It is often helpful to emphasize the presence of a multiset system $(V, \mathcal{F})$ which supports a given structure $(V, \mathcal{C})$, in which case we write $(V, \mathcal{F}, \mathcal{C})$. We call such a triple a *system*, and write $P^{\mathcal{C}}_{\mathcal{F}}(t, \mathbf{X})$ for its multipartition series. Given a multiset system $(V, \mathcal{F})$ we define the *indicator structure* $\mathcal{I}_{\mathcal{F}} : \mathcal{O}(V) \to W$ of $\mathcal{F}$ by $\mathcal{I}_{\mathcal{F}}(\sigma) := 1$ if σ respects $\mathcal{F}$, and $\mathcal{I}_{\mathcal{F}}(\sigma) := 0$ otherwise. We write $P_{\mathcal{F}}(t, \mathbf{X})$ for the multipartition series of $(V, \mathcal{F}, \mathcal{I}_{\mathcal{F}})$. Proof of Proposition 1.2 is left to the reader.

PROPOSITION 1.2. *Let $(V, \mathcal{F})$ be a multiset system. Then for any finite $f : V \to \mathbf{N}$,*

$$[\mathbf{X}^f] P_{\mathcal{F}}(t, \mathbf{X}) = \sum_{\sigma} \frac{t^{\#\sigma}}{(\#\sigma)!} \qquad \text{and} \qquad [\mathbf{X}^f] Z_{\mathcal{F}}(t, \mathbf{X}) = \sum_{\sigma} \binom{t}{\#\sigma}$$

where the summations are over all ordered multipartitions $\sigma \in \mathcal{O}(V)$ which respect $\mathcal{F}$ and have elevation f. Consequently,

$$P_{\mathcal{F}}(t, \mathbf{X}) = \exp\left(t \sum_{S \in \mathcal{F}} \mathbf{X}^S\right) \qquad \text{and} \qquad Z_{\mathcal{F}}(t, \mathbf{X}) = \left(1 + \sum_{S \in \mathcal{F}} \mathbf{X}^S\right)^t .$$

EXAMPLE 1.3. Let $G = (V, E)$ be a simple undirected graph in which V is either finite or countable. Denote by $\mathcal{E}_0(G)$ the set of all finite nonempty subsets S of V which induce an edge-free subgraph of G. Then for any $U \in \mathcal{P}(V)$,

$$[\mathbf{X}^U]\left(1 + \sum_{S \in \mathcal{E}_0(G)} \mathbf{X}^S\right)^t$$

is the usual chromatic polynomial [4, 16] of the subgraph $G|_U$ of G induced by U, and

$$[\mathbf{X}^U] \exp\left(t \sum_{S \in \mathcal{E}_0(G)} \mathbf{X}^S\right)$$

is the σ-polynomial [3, 4, 14] of $G|_U$.

EXAMPLE 1.4. Let $G = (V, E)$ be a simple undirected graph in which V is either finite or countable, and let $\mathcal{K}(G) := \{\{v\} : v \in V\} \cup E$. Then for any $U \in \mathcal{P}(V)$,

$$[\mathbf{X}^U] \exp\left(t \sum_{S \in \mathcal{K}(G)} \mathbf{X}^S\right)$$

is the modified matching polynomial [21] of the subgraph $G|_U$. Generally, for a finite $f : V \to \mathbf{N}$, we have $[\mathbf{X}^f] P_{\mathcal{K}(G)}(t, \mathbf{X}) = \sum_H t^{\#H}/r(H)$, where the summation is over all multisets $H \in \mathcal{M}(E)$ such that for all $v \in V$, $\mathrm{el}(H, v) \leq f(v)$, and where

$$r(H) := \left(\prod_{e \in E} H(e)!\right)\left(\prod_{v \in V} (f(v) - \mathrm{el}(H, v))!\right).$$

More generally than in Proposition 1.2, fix weights $c(S, m) \in W$ for each $S \in \mathcal{M}^+(V)$ and integer $m > 0$, and put $c(S, 0) := 1$ for all $S \in \mathcal{M}^+(V)$. This information may also be encoded by a collection of *structure series*

$$C_S(z) := \sum_{m \geq 0} c(S, m) \frac{z^m}{m!} \tag{5}$$

for each $S \in \mathcal{M}^+(V)$. We define a *separated structure* $(V, \mathcal{C})$ by setting

$$\mathcal{C}(\sigma) := \prod_{S \in \mathcal{M}^+(V)} c(S, \sigma(S)) \tag{6}$$

for each $\sigma \in \mathcal{O}(V)$. Notice that $(V, \mathcal{C})$ is basic. Any system $(V, \mathcal{F}, \mathcal{C})$ in which the structure $(V, \mathcal{C})$ is separated is called a *separated system.* The proof of Proposition 1.5 is analogous to that of Proposition 1.2, and is also omitted.

PROPOSITION 1.5. *Let $(V, \mathcal{F}, \mathcal{C})$ be a separated system defined by structure series $\{C_S(z) : S \in \mathcal{F}\}$. Then for any finite $f : V \to \mathbf{N}$,*

$$[\mathbf{X}^f]P_{\mathcal{F}}^{\mathcal{C}}(t, \mathbf{X}) = \sum_{\pi} \mathcal{C}(\pi)\frac{t^{\#\pi}}{\pi!}$$

where the summation is over all multipartitions π of V which respect $\mathcal{F}$ and have elevation f. Consequently,

$$P_{\mathcal{F}}^{\mathcal{C}}(t, \mathbf{X}) = \prod_{S \in \mathcal{F}} C_S(t\mathbf{X}^S).$$

EXAMPLE 1.6. Let $G = (V, E)$ be an undirected multigraph which has only finitely many loops or edges between each pair of vertices, and let $w : E \to W$ be a weight function on the edges of G. Suppose that for each $S \in \mathcal{M}^+(V)$ with $\#S = 2$, there are $m(S)$ edges in E incident with the multiset S of vertices, and that these edges have weights $w_1(S), \ldots, w_{m(S)}(S)$. We represent (G, w) by the separated system $(V, \mathcal{M}_{\leq 2}, \mathcal{G})$ in which $\mathcal{M}_{\leq 2} := \{S \in \mathcal{M}^+(V) : \#S \leq 2\}$, $\mathcal{G}$ is defined by the structure series

$$G_S(z) := \prod_{i=1}^{m(S)} (1 + w_i(S)z) \tag{7}$$

when $\#S = 2$, and $G_S(z) := 1 + z$ when $\#S = 1$. For $U \in \mathcal{P}(V)$ the polynomial $[\mathbf{X}^U]P_{\mathcal{M}_{\leq 2}}^{\mathcal{G}}(t, \mathbf{X})$ is a weighted modified matching polynomial, similar to that in Example 1.4. However, for general finite $f : V \to \mathbf{N}$ there is a difference: we have $[\mathbf{X}^f]P_{\mathcal{M}_{\leq 2}}^{\mathcal{G}}(t, \mathbf{X}) = \sum_H \omega(H)t^{\#H}$ where the summation is over all subsets $H \subseteq E$ such that for each $v \in V$, $f(v) - 1 \leq \mathrm{el}(H, v) \leq f(v)$, and where $\omega(H) := \prod_{e \in H} w(e)$.

We say that a separated system $(V, \mathcal{F}, \mathcal{C})$ is *completely separated* when for each $S \in \mathcal{F}$ there is a $c(S) \in W$ such that $c(S, m) = c(S)^m$ for all $m \geq 0$. In this case we have

$$P_{\mathcal{F}}^{\mathcal{C}}(t, \mathbf{X}) = \exp\left(t \sum_{S \in \mathcal{F}} c(S)\mathbf{X}^S\right) \text{ and } Z_{\mathcal{F}}^{\mathcal{C}}(t, \mathbf{X}) = \left(1 + \sum_{S \in \mathcal{F}} c(S)\mathbf{X}^S\right)^t. \tag{8}$$

Thus, a multiset system (with its indicator structure) is a special case of a completely separated system. From Proposition 1.2 and equation (8) it is evident that completely separated systems are the same as weighted multiset systems.

EXAMPLE 1.7. Let $G = (V, E)$ be a finite undirected multigraph. Define a completely separated system $\mathcal{G} : \mathcal{O}(V) \to \mathbf{Q}[q]$ as follows. For each $S \in \mathcal{M}^+(V)$ put

$$g(S) := \begin{cases} q^{\#E(G|_S)} & \text{if } S \in \mathcal{P}^+(V), \\ 0 & \text{if } S \notin \mathcal{P}^+(V), \end{cases}$$

and for $\sigma \in \mathcal{O}(V)$ let $\mathcal{G}(\sigma) := \prod_{S \in \mathcal{M}^+(V)} g(S)^{\sigma(S)}$. Consider the polynomial

$$Y_G(q, t) := [\mathbf{X}^V]Z^{\mathcal{G}}(t, \mathbf{X}) = [\mathbf{X}^V]\left(1 + \sum_{S \in \mathcal{P}^+(V)} q^{\#E(G|_S)}\mathbf{X}^S\right)^t.$$

Then $Q(G;x,y) := x^{-\#V(G)}Y_G(x+1,xy)$ is Tutte's dichromatic polynomial [20] of G; also, for $m \in \mathbf{N}$, $Y_G(q,m)$ is the partition function of the m-state Potts model [1] associated with G (where q is a quantity dependent on temperature). Comparison with Example 1.3 shows that $[q^0]Y_G(q,t)$ is the chromatic polynomial of G. Furthermore, for $k \in \mathbf{N}$ and $0 \leq q \leq 1$, $[t^k]Y_G(q,t)$ is the probability that after deleting each edge of G independently with probability q, the remaining spanning edge-subgraph of G has exactly k components. In particular, the case $k = 1$ gives the reliability polynomial [5] of G. These claims can easily be verified by induction using the contraction/deletion formula

$$Y_G(q,t) = (1-q)Y_{G/e}(q,t) + qY_{G\smallsetminus e}(q,t),$$

where $e \in E(G)$ and G/e and $G \smallsetminus e$ represent the contraction and the deletion of e, respectively.

2. Composite systems. Let V be a vertex-set, and let $\mathfrak{U} := \{U_v : v \in V\}$ be a set of pairwise disjoint vertex-sets indexed by V. The *composition of* $\mathfrak{U}$ *into* V is $V[\mathfrak{U}] := \bigsqcup\{U_v : v \in V\}$. We next consider the construction of multisets $S \in \mathcal{M}^+(V[\mathfrak{U}])$. Denote by $\mathcal{N}^+(V,\mathfrak{U})$ the set of those elements $(R,(\pi_v)_{v\in V})$ of $\mathcal{M}^+(V) \times \prod_{v\in V}\mathcal{M}(\mathcal{M}^+(U_v))$ such that for all $v \in V$, $\#\pi_v = R(v)$. Define a function $\Psi : \mathcal{N}^+(V,\mathfrak{U}) \to \mathcal{M}^+(V[\mathfrak{U}])$ by

$$\Psi(R,(\pi_v)) := \sum_{v\in V} \mathrm{el}(\pi_v). \tag{9}$$

We leave the simple verification of Proposition 2.1 to the reader.

PROPOSITION 2.1. *Let V, $\mathfrak{U}$, and Ψ be as above. Then for each $S \in \mathcal{M}^+(V[\mathfrak{U}])$ there is a unique $(R,(\pi_v)) \in \Psi^{-1}(S)$ such that R is a set.*

Now let V and $\mathfrak{U}$ be as above, let $(V,\mathcal{F})$ be a multiset system, and let $(\mathfrak{U},\mathfrak{G}) := \{(U_v,\mathcal{G}_v) : v \in V\}$ be a set of pairwise vertex-disjoint multiset systems indexed by V. We define the *composition of* $(\mathfrak{U},\mathfrak{G})$ *into* $(V,\mathcal{F})$, denoted by $(V[\mathfrak{U}],\mathcal{F}[\mathfrak{G}])$, as follows. The vertex-set $V[\mathfrak{U}]$ has been defined above. A multiset $S \in \mathcal{M}^+(V[\mathfrak{U}])$ is a member of $\mathcal{F}[\mathfrak{G}]$ if and only if there exists at least one $(R,(\pi_v)) \in \Psi^{-1}(S)$ such that $R \in \mathcal{F}$ and π_v respects $\mathcal{G}_v$ for each $v \in V$. The reader is also invited to check Proposition 2.2.

PROPOSITION 2.2. *If $(V,\mathcal{F})$ is a set system and each $(U_v,\mathcal{G}_v)$ is a set system, then $(V[\mathfrak{U}],\mathcal{F}[\mathfrak{G}])$ is also a set system.*

This proposition shows that in this case, the composition of multiset systems defined here reduces to the composition of set systems defined in [21]. Several structural properties of set systems which are preserved by composition are also discussed in [21]. For example, if $(V,\mathcal{F})$ is a simplicial complex and each $(U_v,\mathcal{G}_v)$ is a simplicial complex, then $(V[\mathfrak{U}],\mathcal{F}[\mathfrak{G}])$ is also a simplicial complex.

In order to define compositions of systems we next consider the construction of ordered multipartitions of $V[\mathfrak{U}]$. Let $\Theta(V,\mathfrak{U})$ consist of those elements $(\rho,(\tau_v)_{v\in V})$ of $\mathcal{O}(V) \times \prod_{v\in V}\mathcal{O}(U_v)$ such that for all $v \in V$, $\#\tau_v = \mathrm{el}(\rho,v)$. We define a function $\Upsilon : \Theta(V,\mathfrak{U}) \to \mathcal{O}(V[\mathfrak{U}])$ as follows. Given $(\rho,(\tau_v)) \in \Theta(V,\mathfrak{U})$, for each $v \in V$ we may write $\tau_v = \kappa_v^1 \oplus \cdots \oplus \kappa_v^m$ uniquely subject to the condition that $\#\kappa_v^i = R_i(v)$, where $\rho = (R_1,\ldots,R_m)$. Now for each $i = 1,\ldots,m$ let

$$S_i := \sum_{v\in V} \mathrm{el}(\kappa_v^i) \tag{10}$$

define a multiset in $\mathcal{M}^+(V[\mathfrak{U}])$. Finally, we let $\Upsilon(\rho,(\tau_v)) := (S_1,\dots,S_m)$.

Proposition 2.3 records some useful facts about Υ; their proofs are routine and we omit them.

PROPOSITION 2.3. *Let $(V,\mathcal{F})$ be a multiset system and let $(\mathfrak{U},\mathfrak{G})$ be a collection of pairwise vertex-disjoint multiset systems indexed by V.*

(a) *For each $\sigma \in \mathcal{O}(V[\mathfrak{U}])$, there is a unique $(\rho,(\tau_v)) \in \Upsilon^{-1}(\sigma)$ for which each block of ρ is a set.*

(b) *If $(\rho,(\tau_v)) \in \Theta(V,\mathfrak{U})$ is such that ρ respects $\mathcal{F}$ and each τ_v respects $\mathcal{G}_v$ then $\Upsilon(\rho,(\tau_v))$ respects $\mathcal{F}[\mathfrak{G}]$.*

(c) *If $\sigma \in \mathcal{O}(V[\mathfrak{U}])$ respects $\mathcal{F}[\mathfrak{G}]$ then there is at least one $(\rho,(\tau_v)) \in \Upsilon^{-1}(\sigma)$ for which ρ respects $\mathcal{F}$ and each τ_v respects $\mathcal{G}_v$.*

Now to define the composition of systems, let $(V,\mathcal{F})$ and $(\mathfrak{U},\mathfrak{G})$ be as in Proposition 2.3, let $\mathcal{C}$ be any structure supported on $\mathcal{F}$, and let $\mathfrak{B} := \{\mathcal{B}_v : v \in V\}$ be a set of structures supported on the corresponding members of $\mathfrak{G}$. For an ordered multipartition $\sigma \in \mathcal{O}(V[\mathfrak{U}])$ we define the composite structure $\mathcal{C}[\mathfrak{B}]$ evaluated at σ to be

$$\mathcal{C}[\mathfrak{B}](\sigma) := \sum_{(\rho,(\tau_v))\in\Upsilon^{-1}(\sigma)} \mathcal{C}(\rho) \prod_{v\in V} \mathcal{B}_v(\tau_v). \tag{11}$$

The triple $(V[\mathfrak{U}],\mathcal{F}[\mathfrak{G}],\mathcal{C}[\mathfrak{B}])$ is a *composite system.*

For any multipartition π of V, let $\mathcal{O}_\pi(V)$ denote the set of all multipartitions $\sigma \in \mathcal{O}(V)$ such that $\mathrm{base}(\sigma) = \pi$, and recall that $\#\mathcal{O}_\pi(V) = (\#\pi)!/\pi!$.

LEMMA 2.4. *Let V and $\mathfrak{U}$ be as above, and let $\sigma = (S_1,\dots,S_m) \in \mathcal{O}(V[\mathfrak{U}])$. Then there is a bijective correspondence*

$$\Upsilon^{-1}(\sigma) \rightleftharpoons \prod_{i=1}^{m} \bigsqcup_{(R_i,(\pi_v^i))\in\Psi^{-1}(S_i)} \prod_{v\in V} \mathcal{O}_{\pi_v^i}(V).$$

Proof. We construct a bijection from the right side to the left side as follows. A typical element on the right side corresponds to an m-tuple $((R_1,(\kappa_v^1)),\dots,(R_m,(\kappa_v^m)))$ in which each $\kappa_v^i \in \mathcal{O}(U_v)$ and each $(R_i,(\mathrm{base}(\kappa_v^i))) \in \Psi^{-1}(S_i)$. From this element we construct $\rho := (R_1,\dots,R_m)$ and $\tau_v := \kappa_v^1 \oplus \cdots \oplus \kappa_v^m$ for each $v \in V$; thus $\Upsilon(\rho,(\tau_v)) = \sigma$. One easily checks that this construction in fact gives a bijection.

PROPOSITION 2.5. *Let $(V,\mathcal{F},\mathcal{C})$ and $(\mathfrak{U},\mathfrak{G},\mathfrak{B})$ be as above.*

(a) *The structure $\mathcal{C}[\mathfrak{B}]$ is supported on $\mathcal{F}[\mathfrak{G}]$.*

(b) *If $\mathcal{C}$ is basic and each $\mathcal{B}_v$ is basic then $\mathcal{C}[\mathfrak{B}]$ is basic.*

(c) *If $\mathcal{C}$ is completely separated and each $\mathcal{B}_v$ is completely separated then $\mathcal{C}[\mathfrak{B}]$ is completely separated.*

(d) *If $\mathcal{F}$ is a set system and $\mathcal{C}$ and all $\mathcal{B}_v$ are indicator structures, then $\mathcal{C}[\mathfrak{B}]$ is an indicator structure.*

Proof. Parts (a) and (b) are routine. For part (c), from (11) and Lemma 2.4 we calculate that

$$\begin{aligned}\mathcal{C}[\mathfrak{B}](\sigma) &= \sum_{(\rho,(\tau_v))\in\Upsilon^{-1}(\sigma)} \left(\prod_{R\in\mathcal{M}^+(V)} c(R)^{\rho(R)}\right) \prod_{v\in V} \mathcal{B}_v(\tau_v) \\ &= \prod_{i=1}^{m} \sum_{(R_i,(\pi_v^i))\in\Psi^{-1}(S_i)} c(R_i) \prod_{v\in V} \mathcal{B}_v(\pi_v^i) \frac{(\#\pi_v^i)!}{\pi_v^i!},\end{aligned}$$

where the second equality is justified because if $\tau_v = \kappa_v^1 \oplus \cdots \oplus \kappa_v^m$ and $\mathcal{B}_v$ is completely separated, then it is basic and $\mathcal{B}_v(\tau_v) = \mathcal{B}_v(\kappa_v^1) \ldots \mathcal{B}_v(\kappa_v^m)$. Therefore, if we put

$$h(S) := \sum_{(R,(\pi_v))\in\Psi^{-1}(S)} c(R) \prod_{v\in V} \mathcal{B}_v(\pi_v) \frac{(\#\pi_v)!}{\pi_v!} \tag{12}$$

for each $S \in \mathcal{M}^+(V[\mathfrak{U}])$, it follows that

$$\mathcal{C}[\mathfrak{B}](\sigma) = \prod_{S\in\mathcal{M}^+(V[\mathfrak{U}])} h(S)^{\sigma(S)}$$

for each $\sigma \in \mathcal{O}(V[\mathfrak{U}])$. Comparing this with equation (6), we see that $\mathcal{C}[\mathfrak{B}]$ is completely separated.

Part (d) is a direct consequence of Proposition 2.1 and equation (12).

The following composition theorem was proved in [21] in the special case of finite set systems, but only for the $\mathbf{X}^1$th coefficient, and by a much more clumsy argument. Let $\mathbf{y} := \{y_v : v \in V\}$ be pairwise commuting independent indeterminates, let $\mathbf{d} := \{\partial_v : v \in V\}$ where $\partial_v := \partial/\partial y_v$ for each $v \in V$, and let $\mathbf{X}|_v := \{X_u : u \in U_v\}$ for each $v \in V$.

THEOREM 2.6. *Let $(V, \mathcal{F}, \mathcal{C})$ and $(\mathfrak{U}, \mathfrak{G}, \mathfrak{B})$ be as above. Then*

$$P^{\mathcal{C}[\mathfrak{B}]}_{\mathcal{F}[\mathfrak{G}]}(t, \mathbf{X}) = [\mathbf{y}^0] P^{\mathcal{C}}_{\mathcal{F}}(t, \mathbf{d}) \prod_{v\in V} P^{\mathcal{B}_v}_{\mathcal{G}_v}(y_v, \mathbf{X}|_v).$$

Proof. We collect terms on the right side using the relevant definitions, viz:

$$\begin{aligned}
&[\mathbf{y}^0] P^{\mathcal{C}}_{\mathcal{F}}(t, \mathbf{d}) \prod_{v\in V} P^{\mathcal{B}_v}_{\mathcal{G}_v}(y_v, \mathbf{X}|_v) \\
&= [\mathbf{y}^0] \left(\sum_{\rho\in\mathcal{O}(V)} \mathcal{C}(\rho) \frac{t^{\#\rho}}{(\#\rho)!} \mathbf{d}^{\mathrm{el}(\rho)} \right) \prod_{v\in V} \sum_{\tau_v\in\mathcal{O}(U_v)} \mathcal{B}_v(\tau_v) \frac{y_v^{\#\tau_v}}{(\#\tau_v)!} (\mathbf{X}|_v)^{\mathrm{el}(\tau_v)} \\
&= \sum_{(\rho,(\tau_v))\in\Theta(V,\mathfrak{U})} \mathcal{C}(\rho) \frac{t^{\#\rho}}{(\#\rho)!} \prod_{v\in V} \mathcal{B}_v(\tau_v) (\mathbf{X}|_v)^{\mathrm{el}(\tau_v)} \\
&= \sum_{\sigma\in\mathcal{O}(V[\mathfrak{U}])} \frac{t^{\#\sigma}}{(\#\sigma)!} \mathbf{X}^{\mathrm{el}(\sigma)} \sum_{(\rho,(\tau_v))\in\Upsilon^{-1}(\sigma)} \mathcal{C}(\rho) \prod_{v\in V} \mathcal{B}_v(\tau_v),
\end{aligned}$$

as was to be shown.

Theorem 2.6 implies that

$$P^{\mathcal{C}}_{\mathcal{F}}(t, \mathbf{X}) = [\mathbf{y}^0] P^{\mathcal{C}}_{\mathcal{F}}(t, \mathbf{d}) \prod_{v\in V} \exp(y_v X_v) \tag{13}$$

for any system $(V, \mathcal{F}, \mathcal{C})$. Thus, we lose no information in shifting our attention from the multipartition series $P^{\mathcal{C}}_{\mathcal{F}}(t, \mathbf{X})$ to the *compositional operator* $\Phi^{\mathcal{C}}_{\mathcal{F}} : W[t, \mathbf{y}] \to W[t]$ of the system $(V, \mathcal{F}, \mathcal{C})$, defined by $\Phi^{\mathcal{C}}_{\mathcal{F}} := [\mathbf{y}^0] P^{\mathcal{C}}_{\mathcal{F}}(t, \mathbf{d})$. The utility of this point of view is seen in the next section. It is important to notice that for any system $(V, \mathcal{F}, \mathcal{C})$, $\Phi^{\mathcal{C}}_{\mathcal{F}}$ is $W[t]$-linear. Another useful property is given in Proposition 2.7, the proof of

which is omitted as it follows straightforwardly from the chain rule for differentiation. For a multiset $S \in \mathcal{M}^+(V)$ and a vertex $v \in S$, let $S \setminus v$ denote the multiset given by

$$(S \setminus v)(w) := \begin{cases} S(w) & \text{if } w \neq v, \\ S(v) - 1 & \text{if } w = v. \end{cases}$$

Given a separated system $(V, \mathcal{F}, \mathcal{C})$ and $S \in \mathcal{F}$, let $(V, \mathcal{F} \setminus S, \mathcal{C} \setminus C_S)$ denote the separated system supported on $\mathcal{F} \setminus S$ and defined by the structure series $\{C_T(z) : T \in \mathcal{F} \setminus S\}$.

PROPOSITION 2.7. *Let $(V, \mathcal{F}, \mathcal{C})$ be a separated system defined by structure series $\{C_S(z) : S \in \mathcal{F}\}$. Then for any $v \in V$ and $Q(\mathbf{y}) \in W[\mathbf{y}]$,*

$$\Phi^{\mathcal{C}}_{\mathcal{F}} y_v Q(\mathbf{y}) = t \sum_{v \in S \in \mathcal{F}} \Phi^{\mathcal{C} \setminus C_S}_{\mathcal{F} \setminus S} C'_S(t\mathbf{d}^S)\mathbf{d}^{S \setminus v} Q(\mathbf{y})$$

where $C'_S(z) = dC_S(z)/dz$.

3. Graphic systems. Throughout this section we take $\mathbf{R}$ for the weight ring W. We use the method of interlacing zeros to show that certain conditions on a system $(V, \mathcal{F}, \mathcal{C})$ imply that each coefficient $[\mathbf{X}^f]P^{\mathcal{C}}_{\mathcal{F}}(t, \mathbf{X})$ of the multipartition series is a polynomial in $\mathbf{R}[t]$ with only real nonpositive zeros. This method is outlined in the Appendix for the readers' convenience.

For lack of a better term we shall say that $Q(\mathbf{y}) \in \mathbf{R}[\mathbf{y}]$ is *realistic* when $Q(\mathbf{y}) = \prod_{v \in V} Q_v(y_v)$ and each $Q_v(y_v)$ has only real nonpositive zeros (and, of course, only finitely many $Q_v(y_v) \neq 1$). Consider the following three conditions on the compositional operator $\Phi^{\mathcal{C}}_{\mathcal{F}}$ of a system $(V, \mathcal{F}, \mathcal{C})$:

(α) For any realistic $Q(\mathbf{y})$, $\Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y})$ has only real nonpositive zeros.
(β) For any realistic $Q(\mathbf{y})$, and any $v \in V$, $\Phi^{\mathcal{C}}_{\mathcal{F}} \partial_v Q(\mathbf{y}) \prec \Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y})$.
(γ) For any realistic $Q(\mathbf{y})$, any $v \in V$, and any $v \in S \in \mathcal{F}$,

$$\Phi^{\mathcal{C} \setminus C_S}_{\mathcal{F} \setminus S} C'_S(t\mathbf{d}^S)\mathbf{d}^{S \setminus v} Q(\mathbf{y}) \prec \Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y}).$$

We prove that these three conditions hold for a certain class of separated systems.

A *Pólya–Laguerre class one series* is a series $C(z) \in \mathbf{R}[[z]]$ of the form

$$C(z) := \exp(a_0 z) \prod_{i \geq 1} (1 + a_i z) \tag{14}$$

in which $a_i \geq 0$ for all $i \geq 0$, and $\sum_{i \geq 1} a_i$ converges (cf. [13, Chap. 7, §2]). A *Pólya–Laguerre class one structure* is a separated structure defined by a Pólya–Laguerre class one structure series. A multiset system $(V, \mathcal{F})$ is *graphic* when $\#S \leq 2$ for all $S \in \mathcal{F}$, and a *Pólya–Laguerre class one graphic system* is a Pólya–Laguerre class one structure supported on a graphic multiset system. (Examples 1.4 and 1.6 are both of this form.)

THEOREM 3.1. *If $(V, \mathcal{F}, \mathcal{C})$ is a Pólya–Laguerre class one graphic system then $\Phi^{\mathcal{C}}_{\mathcal{F}}$ satisfies conditions (α), (β), and (γ).*

Proof. Let $Q(\mathbf{y})$ be realistic. We prove the conditions simultaneously for all such systems by induction on $\deg Q := \sum_{v \in V} \deg Q_v$. The basis of induction $\deg Q \leq 1$ is easily checked. Suppose that the theorem has been proved for $\deg Q \leq n - 1$ and assume that $\deg Q = n > 0$. Without loss of generality we may assume that each Q_v is monic.

Let $v \in V$ be any vertex such that $\deg Q_v > 0$, let $\theta \geq 0$ be such that $Q_v(-\theta) = 0$, and put $\widehat{Q}(\mathbf{y}) := Q(\mathbf{y})/(y_v + \theta)$. By linearity of $\Phi^{\mathcal{C}}_{\mathcal{F}}$ and Proposition 2.7 we have $\Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y}) = tR(t) + \theta \Phi^{\mathcal{C}}_{\mathcal{F}} \widehat{Q}(\mathbf{y})$, where

$$R(t) := \sum_{v \in S \in \mathcal{F}} \Phi^{\mathcal{C} \smallsetminus C_S}_{\mathcal{F} \smallsetminus S} C'_S(t\mathbf{d}^S) \mathbf{d}^{S \smallsetminus v} \widehat{Q}(\mathbf{y}).$$

Now $\widehat{Q}(\mathbf{y})$ satisfies the induction hypothesis, so that by (α) and (γ) applied to $\widehat{Q}(\mathbf{y})$ and Lemmas A.3 and A.1(c) we see that

$$\Phi^{\mathcal{C}}_{\mathcal{F}} \widehat{Q}(\mathbf{y}) \prec tR(t)$$

and hence by Lemma A.2(c) that

$$\Phi^{\mathcal{C}}_{\mathcal{F}} \widehat{Q}(\mathbf{y}) \prec \Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y}) \lll tR(t).$$

This establishes part (α) of the induction step.

To establish part (β) of the induction step, let $v \in V$ be arbitrary. If $\deg Q_v = 0$ then $\partial_v Q(\mathbf{y}) = 0$ and there is nothing to prove. Otherwise, let the zeros of Q_v be $-\theta_{v1}, \ldots, -\theta_{vk}$, and for $i = 1, \ldots, k$ put $\widehat{Q}_i(\mathbf{y}) := Q(\mathbf{y})/(y_v + \theta_{vi})$. By part (α) of the induction step above, we have $\Phi^{\mathcal{C}}_{\mathcal{F}} \widehat{Q}_i(\mathbf{y}) \prec \Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y})$ for all $i = 1, \ldots, k$. But $\partial_v Q(\mathbf{y}) = \sum_{i=1}^{k} \widehat{Q}_i(\mathbf{y})$, so that by linearity of $\Phi^{\mathcal{C}}_{\mathcal{F}}$ and Lemma A.3 we find that $\Phi^{\mathcal{C}}_{\mathcal{F}} \partial_v Q(\mathbf{y}) \prec \Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y})$, as desired.

It remains to verify part (γ) of the induction step. Let $v \in V$ and $v \in S \in \mathcal{F}$ be arbitrary. Let the Pólya–Laguerre class one structure series $C_S(z)$ be given in (14). Thus $C'_S(z) = \sum_{j \geq 0} a_j \widehat{C}^j_S(z)$ where

$$\widehat{C}^j_S(z) := \begin{cases} C_S(z) & \text{if } j = 0, \\ C_S(z)/(1 + a_j z) & \text{if } j \geq 1. \end{cases}$$

For each $j \geq 0$ let $\Gamma_j := \Phi^{\mathcal{C} \smallsetminus C_S}_{\mathcal{F} \smallsetminus S} \widehat{C}^j_S(t\mathbf{d}^S)$. By linearity of $\Phi^{\mathcal{C} \smallsetminus C_S}_{\mathcal{F} \smallsetminus S}$ and Lemma A.3, in order to prove that (γ) holds it suffices to show that for all $j \geq 0$,

$$\Gamma_j \mathbf{d}^{S \smallsetminus v} Q(\mathbf{y}) \prec \Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y}). \tag{15}$$

We do this by checking two cases: $j = 0$ or $j \geq 1$. When $\#S = 2$ let $S = \{v, w\}$, where $w = v$ is possible. When $j = 0$ we have

$$\Gamma_j \mathbf{d}^{S \smallsetminus v} Q(\mathbf{y}) = \begin{cases} \Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y}) & \text{if } \#S = 1, \\ \Phi^{\mathcal{C}}_{\mathcal{F}} \partial_w Q(\mathbf{y}) & \text{if } \#S = 2. \end{cases}$$

In either subcase, parts (α) and (β) of the induction step establish the validity of (15). For $j \geq 1$ we have

$$\Gamma_j (1 + a_j t\mathbf{d}^S) Q(\mathbf{y}) = \Phi^{\mathcal{C}}_{\mathcal{F}} Q(\mathbf{y}). \tag{16}$$

In either subcase $\#S = 1$ or $\#S = 2$, parts (α) and (β) of the induction step imply that

$$\Gamma_j \mathbf{d}^{S \smallsetminus v} Q(\mathbf{y}) \prec \Gamma_j Q(\mathbf{y}) \tag{17}$$

and that

$$\Gamma_j \mathbf{d}^S Q(\mathbf{y}) \prec \Gamma_j \mathbf{d}^{S \setminus v} Q(\mathbf{y}). \tag{18}$$

Hence, from (18) and Lemma A.1(c) we deduce that

$$\Gamma_j \mathbf{d}^{S \setminus v} Q(\mathbf{y}) \prec t\Gamma_j \mathbf{d}^S Q(\mathbf{y}). \tag{19}$$

Now from (16), (17), (19), and Lemma A.3, we conclude that (15) holds. This completes the induction step and the proof.

Theorem 3.1 has as an application the following theorem.

THEOREM 3.2. *Let $G = (V, E)$ be a finite multigraph. For each $v \in V$, let $Q_v(y) := \sum_m N_v(m) y^m/m!$ be a polynomial with only real nonpositive zeros. For each $m \geq 0$ let $N(m) := \sum_H \prod_{v \in V} N_v(\deg_H(v))$, where the sum is over all edge-subgraphs H of G with m edges. Then the polynomial $\sum_m N(m) t^m$ has only real nonpositive zeros.*

Proof. Let $Q(\mathbf{y}) := \prod_{v \in V} Q_v(y_v)$ and let $(V, \mathcal{M}_2, \mathcal{G})$ be the separated system supported on $\mathcal{M}_2 := \{S \in \mathcal{M}(V) : \#S = 2\}$ and defined by structure series $G_S(z) := (1+z)^{m(S)}$ where $m(S)$ is the number of edges of G incident with the multiset S of vertices, for each $S \in \mathcal{M}_2(V)$. The polynomial in the conclusion is then $\Phi^{\mathcal{G}}_{\mathcal{M}_2} Q(\mathbf{y})$. But $Q(\mathbf{y})$ is realistic, and $(V, \mathcal{M}_2, \mathcal{G})$ is a Pólya–Laguerre class one graphic system, by hypothesis. The result follows from Theorem 3.1, by property (α).

As a further special case, we have the following direct generalization of the Heilmann–Lieb theorem (which is the special case $f_0 \equiv 0$ and $f_1 \equiv 1$).

THEOREM 3.3. *Let $G = (V, E)$ be a finite multigraph. Fix two functions f_0 and f_1 from V to $\mathbf{N}$ such that for all $v \in V$, $f_0(v) \leq f_1(v) \leq f_0(v) + 1$. For each $m \in \mathbf{N}$ let $N(m)$ be the number of edge-subgraphs H of G which have m edges and are such that for all $v \in V$, $f_0(v) \leq \deg_H(v) \leq f_1(v)$. Then the polynomial $\sum_m N(m) t^m$ has only real zeros.*

Proof. This follows from Theorem 3.2 by taking

$$N_v(m) := \begin{cases} 1 & \text{if } \ m = f_0(v) \ \text{ or } \ m = f_1(v), \\ 0 & \text{otherwise.} \end{cases}$$

As the following example shows, this theorem is in a sense the best possible. Consider the graph $K_{1,3}$ and let w be the vertex of degree three. Take $f_0 \equiv 0$, $f_1(w) = 2$, and $f_1(v) = 1$ for $v \neq w$. Then the polynomial in the conclusion of Theorem 3.3 is in this case $1 + 3t + 3t^2$, which has nonreal zeros.

By a simple construction we can obtain a consequence of Theorem 3.3 which is superficially more general. Given a finite multigraph $G = (V, E)$, the multiset F of *flags* of G consists of those pairs (v, e) with $v \in V$, $e \in E$, and $v \in e$, counted with multiplicities. Thus, a loop e at v contributes two copies of (v, e) to F. The *vertex-partition* of F is the multipartition ν of F, with elevation F, such that (v, e) and (v', e') are in the same block of ν if and only if $v = v'$.

THEOREM 3.4. *Let $G = (V, E)$ be a finite multigraph, and let π be any refinement of the vertex-partition of F. Fix two functions $f_0, f_1 : \pi \to \mathbf{N}$ such that for each $B \in \pi$, $f_0(B) \leq f_1(B) \leq f_0(B) + 1$. For each $m \geq 0$, let $N(m)$ denote the number of edge-subgraphs H of G with m edges, which are such that for each $B \in \pi$, the number of flags of H which are in B is between $f_0(B)$ and $f_1(B)$. Then the polynomial $\sum_m N(m) t^m$ has only real zeros.*

Proof. Construct a new multigraph $G' = (V', E')$ as follows. The vertex-set is $V' := \pi$. Given an edge $e = \{u, v\}$ of E, let B_u^e and B_v^e be the blocks of π which contain the flags (u, e) and (v, e), respectively. Then put $e' := \{B_u^e, B_v^e\}$ and $E' := \{e' : e \in E\}$, counted with multiplicities. Now, by applying Theorem 3.3 to the graph G' we obtain the result.

4. Order series. For this section we assume some familiarity with the theory of labelled posets and P-partitions, as developed in [17, 18]. Let V be a finite set with $\#V = n$, say, and let $A := (V, <, \omega)$ be a labelled poset with underlying set V; that is, $(V, <)$ is a poset and $\omega : V \to [n]$ is any bijection, where $[n] := \{1, 2, \ldots, n\}$. When ω is order-preserving, A is *naturally labelled*; when ω is order-reversing, A is *strictly labelled*. The case of naturally labelled posets subsumes unlabelled posets within this framework. An *inversion* in A is a pair $(a, b) \in V^2$ with $a < b$ and $\omega(a) > \omega(b)$. Two labellings $\omega : V \to [n]$ and $\omega' : V \to [n]$ are *isotonic* when the corresponding sets of inversions are equal. Isotony is thus an equivalence relation, and everything we have to say about a labelled poset depends only upon its isotony class. For any subset S of V, we use $\downarrow S$ to denote the lower order ideal ("downset") generated by S, and $\max S$ to denote the set of maximal elements of S.

Given a labelled poset A, define a structure $\mathcal{A} : \mathcal{O}(V) \to \mathbf{Q}$ as follows. For an ordered multipartition $\sigma = (S_1, \ldots, S_m) \in \mathcal{O}(V)$ we put $\mathcal{A}(\sigma) := 1$ if for each $i = 1, \ldots, m$,

(i) $S_i \in \mathcal{P}^+(V)$,

(ii) S_i^2 contains no inversions of A, and

(iii) $S_i \cap \downarrow(S_1 \cup \cdots \cup S_{i-1}) \subseteq \max(S_1 \cup \cdots \cup S_{i-1})$;

otherwise we put $\mathcal{A}(\sigma) := 0$. The structure $(V, \mathcal{A})$ is called the *order structure* of A. Let $\mathcal{O}(A) := \mathcal{O}(V, <, \omega)$ denote the set of all $\sigma \in \mathcal{O}(V)$ such that $\mathcal{A}(\sigma) = 1$. Notice that $(V, \mathcal{A})$ is supported on $\mathcal{P}^+(V)$, and that $\mathcal{A}$ is not separated, and in fact is not even basic unless $(V, <)$ is an antichain.

Consider a labelled poset $A = (V, <, \omega)$ with $\#V = n$, and for each $0 \le j \le n$, let $e_j(A)$ be the number of order-preserving surjections $f : A \to [j]$ such that if (a, b) is an inversion in A then $f(a) < f(b)$. The *order polynomial* [17, 18] of A is

$$\Omega(A; t) := \sum_{j=0}^{n} e_j(A) \binom{t}{j}.$$

PROPOSITION 4.1. *Let $A = (V, <, \omega)$ be a finite labelled poset. Then $[\mathbf{X}^V] Z^{\mathcal{A}}(t, \mathbf{X})$ is the order polynomial of A.*

Proof. Notice that $[\mathbf{X}^V] Z^{\mathcal{A}}(t, \mathbf{X}) = \sum_\sigma \binom{t}{\#\sigma}$ where the sum is over all $\sigma \in \mathcal{O}(A)$ such that $\mathrm{el}(\sigma) = V$. To each such $\sigma = (S_1, \ldots, S_j)$ we associate a surjection $f : A \to [j]$ by $f^{-1}(i) := S_i$ for each $1 \le i \le j$. This gives a bijection between the set of $\sigma \in \mathcal{O}(A)$ with $\mathrm{el}(\sigma) = V$ and the set of surjections in the definition of $\Omega(A; t)$. The result follows.

Actually, the multipartition series $P^{\mathcal{A}}(t, \mathbf{X})$ of order structures do not possess many nice properties, and it is more convenient to work with a close relative. By analogy with the terminology of [3, 4], we define the "augmented multipartition series" of a system $(V, \mathcal{F}, \mathcal{C})$ to be

$$\widetilde{P}_{\mathcal{F}}^{\mathcal{C}}(t, \mathbf{X}) := \sum_{\sigma \in \mathcal{O}(V)} \mathcal{C}(\sigma) t^{\#\sigma} \mathbf{X}^{\mathrm{el}(\sigma)}. \tag{20}$$

In the case of the order structure of a finite labelled poset A, the polynomial $[\mathbf{X}^V]\widetilde{P}^{\mathcal{A}}(t,\mathbf{X})$ is the *E-polynomial* [2, 15, 22] of A: $E(A;t) := \sum_{j=0}^{n} e_j(A)t^j$, with the notation as above. The following conjecture is due to Neggers [15] in the naturally labelled case, and to Stanley in general [19].

CONJECTURE 4.2. *Let $A = (V, <, \omega)$ be a labelled poset. Then, for any $U \in \mathcal{M}^+(V)$, all zeros of $[\mathbf{X}^U]\widetilde{P}^{\mathcal{A}}(t,\mathbf{X})$ are in the interval $[-1,0]$.*

Actually, the conjecture has previously been made only for $U \in \mathcal{P}^+(V)$, but as we see in Corollary 4.5, this is not an essential difference.

For example, consider a strictly labelled totally ordered set A. Any set of at least two elements of V then contains an inversion in A, from which it follows that

$$\widetilde{P}^{\mathcal{A}}(t,\mathbf{X}) = \prod_{v\in V} \frac{1}{1-tX_v}. \tag{21}$$

One also sees that if A is an antichain then

$$\widetilde{P}^{\mathcal{A}}(t,\mathbf{X}) = \left(1 - t\sum_{S\in\mathcal{P}^+(V)} \mathbf{X}^S\right)^{-1}. \tag{22}$$

The following composition formula for augmented multipartition series can be proved by mimicking the proof of Theorem 2.6. We use the notation $\mathbf{y}^{-1}$ for $\{y_v^{-1} : v \in V\}$.

PROPOSITION 4.3. *Let $(V,\mathcal{F},\mathcal{C})$ and $(\mathfrak{U},\mathfrak{G},\mathfrak{B})$ be as in Theorem 2.6. Then*

$$\widetilde{P}^{\mathcal{C}[\mathfrak{B}]}_{\mathcal{F}[\mathfrak{G}]}(t,\mathbf{X}) = [\mathbf{y}^0]\widetilde{P}^{\mathcal{C}}_{\mathcal{F}}(t,\mathbf{y}^{-1}) \prod_{v\in V} \widetilde{P}^{\mathcal{B}_v}_{\mathcal{G}_v}(y_v, \mathbf{X}|_v).$$

It is clear that for any system $(V,\mathcal{F},\mathcal{C})$,

$$\widetilde{P}^{\mathcal{C}}_{\mathcal{F}}(t,\mathbf{X}) = [\mathbf{y}^0]\widetilde{P}^{\mathcal{C}}_{\mathcal{F}}(t,\mathbf{y}^{-1}) \prod_{v\in V} \frac{1}{1-y_vX_v}. \tag{23}$$

Given a labelled poset $A = (V, <, \omega)$ and pairwise disjoint labelled posets $\mathbf{B} = \{B_v : v \in V\}$ indexed by V, we define the *composition of* $\mathbf{B}$ *into* A as follows [22]. For each $v \in V$ let $B_v := (U_v, <, \nu_v)$ and let $m_v := \#U_v$. Then $A[\mathbf{B}] := (V[\mathfrak{U}], <, \varphi)$ where $V[\mathfrak{U}] := \bigcup_{v\in V} U_v$, and $x < y$ in $A[\mathbf{B}]$ if and only if either $x \in U_a$ and $y \in U_b$ with $a < b$ in A, or $x < y$ in B_v for some $v \in V$. We put $r := \sum_{v\in V} m_v$ and note that there is a unique set of order-preserving injections $\psi_v : [m_v] \to [r]$ for $v \in V$ such that their images are pairwise disjoint and if $\omega(a) < \omega(b)$ then $\psi_a(i) < \psi_b(j)$ for all $i \in [m_a]$ and $j \in [m_b]$. Finally, we define φ on $V[\mathfrak{U}]$ by putting $\varphi|_{U_v} := \psi_v \circ \nu_v$ for each $v \in V$.

The following theorem shows that the composition of systems defined in §2 specializes to a relationship among order structures of composite labelled posets.

THEOREM 4.4. *Let $A = (V, <, \omega)$ be a finite labelled poset and let $\mathbf{B} = \{B_v : v \in V\}$ be a collection of pairwise disjoint labelled posets indexed by V. Let $(U_v, \mathcal{B}_v)$ be the order structure of B_v, for each $v \in V$, and put $(\mathfrak{U},\mathfrak{B}) := \{(U_v,\mathcal{B}_v) : v \in V\}$. Then the composite structure $(V[\mathfrak{U}], \mathcal{A}[\mathfrak{B}])$ is the order structure of the composite labelled poset $A[\mathbf{B}]$.*

Proof. By definition, for any $\sigma \in \mathcal{O}(V[\mathfrak{U}])$,

$$\mathcal{A}[\mathfrak{B}](\sigma) = \sum_{(\rho,(\tau_v))\in\Upsilon^{-1}(\sigma)} \mathcal{A}(\rho) \prod_{v\in V} \mathcal{B}_v(\tau_v).$$

Since $\mathcal{A}$ is supported on $\mathcal{P}^+(V)$, Proposition 2.3(a) implies that the only term such that $\mathcal{A}(\rho) \neq 0$ is the unique $(\rho,(\tau_v)) \in \Upsilon^{-1}(\sigma)$ for which each block of ρ is a set. Hence $\mathcal{A}[\mathfrak{B}](\sigma) = 1$ if and only if $\mathcal{A}(\rho) = 1$ and $\mathcal{B}_v(\tau_v) = 1$ for all $v \in V$. We now show that these conditions are equivalent to having $\sigma \in \mathcal{O}(A[\mathbf{B}])$, which suffices to prove the theorem. As the argument consists of straightforwardly checking definitions we merely outline the main steps.

Consider any $\sigma = (S_1, \ldots, S_m) \in \mathcal{O}(V[\mathfrak{U}])$ and let $(\rho,(\tau_v))$ be the unique element of $\Upsilon^{-1}(\sigma)$ for which each block of $\rho = (R_1, \ldots, R_m)$ is a set. Then for each $v \in V$, $\tau_v = \kappa_v^1 \oplus \cdots \oplus \kappa_v^m$, where the ordered multipartitions κ_v^i are $\kappa_v^i = ()$ if $v \notin R_i$, and $\kappa_v^i = (S_i \cap U_v)$ otherwise. Let $T_v^i := S_i \cap U_v$. We make three claims, the proofs of which are left to the reader.

Claim 1. For each $1 \leq i \leq m$, S_i is a set if and only if for each $v \in V$, T_v^i is a set.

Claim 2. For each $1 \leq i \leq m$, S_i^2 contains no inversions of $A[\mathbf{B}]$ if and only if R_i^2 contains no inversions of A, and for each $v \in V$, $(T_v^i)^2$ contains no inversions of B_v.

Claim 3. For each $1 \leq i \leq m$:

$$S_i \cap {\downarrow}(S_1 \cup \cdots \cup S_{i-1}) \subseteq \max(S_1 \cup \cdots \cup S_{i-1})$$

if and only if

$$R_i \cap {\downarrow}(R_1 \cup \cdots \cup R_{i-1}) \subseteq \max(R_1 \cup \cdots \cup R_{i-1})$$

and, for each $v \in V$,

$$T_v^i \cap {\downarrow}(T_v^1 \cup \cdots \cup T_v^{i-1}) \subseteq \max(T_v^1 \cup \cdots \cup T_v^{i-1}).$$

Given these claims, we may argue as follows. For any $\sigma \in \mathcal{O}(V[\mathfrak{U}])$, the first paragraph shows that $\mathcal{A}[\mathfrak{B}](\sigma) = 1$ if and only if $\mathcal{A}(\rho) = 1$ and, for each $v \in V$, $\mathcal{B}_v(\tau_v) = 1$. This holds if and only if ρ satisfies conditions (i), (ii), and (iii) for A, and for each $v \in V$, τ_v satisfies conditions (i), (ii), and (iii) for B_v, by definition. The three claims above show that this occurs if and only if σ satisfies conditions (i), (ii), and (iii) for $A[\mathbf{B}]$, which completes the proof.

COROLLARY 4.5. *Let $A = (V, <, \omega)$ be a finite labelled poset, and let $U \in \mathcal{M}^+(V)$. Then $[\mathbf{X}^U]Z^{\mathcal{A}}(t, \mathbf{X})$ is the order polynomial of $A[\mathbf{B}]$, where, for each $v \in V$, B_v is a strictly labelled totally ordered set of size $U(v)$.*

Proof. Equivalently, we show that $[\mathbf{X}^U]\widetilde{P}^{\mathcal{A}}(t, \mathbf{X})$ is the E-polynomial of $A[\mathbf{B}]$. Notice that

$$[\mathbf{X}^U]\widetilde{P}^{\mathcal{A}}(t, \mathbf{X}) = [\mathbf{y}^0]\widetilde{P}^{\mathcal{A}}(t, \mathbf{y}^{-1})\mathbf{y}^U,$$

and from equation (21), the E-polynomial (in the indeterminate y) of a strictly labelled totally ordered set of size n is y^n. Proposition 4.3 and Theorem 4.4 imply the result.

In principle, Proposition 4.3 and Theorem 4.4 give a composition formula for E-polynomials of labelled posets, but in order to use it effectively we must be able to describe the series $\widetilde{P}^{\mathcal{A}}(t, \mathbf{X})$ explicitly. In fact, one can be quite specific about the form of $\widetilde{P}^{\mathcal{A}}(t, \mathbf{X})$, but the arguments are too long to be included here and appear in another paper [23].

A. Appendix: The method of interlacing zeros. We summarize here the concepts and lemmas required for our proof of Theorem 3.1.

Suppose that $p, q \in \mathbf{R}[t]$ both have only real zeros, that those of p are $\xi_1 \leq \cdots \leq \xi_n$, and that those of q are $\theta_1 \leq \cdots \leq \theta_m$. We say that q *interlaces* p if $\deg p = 1 + \deg q$ and the zeros of p and q satisfy

$$\xi_1 \leq \theta_1 \leq \xi_2 \leq \cdots \leq \theta_m \leq \xi_{m+1}.$$

We also say that q *alternates left of* p if $\deg p = \deg q$ and the zeros of p and q satisfy

$$\theta_1 \leq \xi_1 \leq \theta_2 \leq \cdots \leq \theta_m \leq \xi_m.$$

We use the notations $p \dagger q$ for "p interlaces q," $p \ll q$ for "p alternates left of q," and $p \prec q$ for "either $p \dagger q$ or $p \ll q$." Any polynomial which stands in one of these relations must have only real zeros. By convention we say that for any real-rooted polynomial p, all of $p \dagger 0$, $0 \dagger p$, $p \ll 0$, and $0 \ll p$ hold.

Lemma A.1 is immediate from the definitions.

LEMMA A.1. *Let $p, q \in \mathbf{R}[t]$ have only real nonpositive zeros.*

(a) *One has $q \dagger p$ if and only if $p \ll tq$.*

(b) *One has $q \ll p$ if and only if $p \dagger tq$.*

(c) *One has $q \prec p$ if and only if $p \prec tq$.*

The next two lemmas are easy consequences of the intermediate value theorem.

LEMMA A.2. *Suppose that $p, q \in \mathbf{R}[t]$ have only real zeros and have leading coefficients of the same sign.*

(a) *If $q \ll p$ then $q \ll q + p$ and $q + p \ll p$.*

(b) *If $q \dagger p$ then $q \dagger q + p$ and $q + p \ll p$.*

(c) *If $q \prec p$ then $q \prec q + p$ and $q + p \ll p$.*

LEMMA A.3. *Suppose that $p, q_1, \ldots, q_n \in \mathbf{R}[t]$ have only real zeros and have leading coefficients of the same sign, and put $f = q_1 + \cdots + q_n$. If for each $i = 1, \ldots, n$ either $q_i \dagger p$ or $q_i \ll p$ then either $f \dagger p$ or $f \ll p$. The first case occurs if and only if $q_i \dagger p$ for all $i = 1, \ldots, n$.*

REFERENCES

[1] R. J. BAXTER, *Exactly Solved Models in Statistical Mechanics*, Academic Press, London, New York, 1982.

[2] F. BRENTI, *Unimodal, Log-concave, and Pólya Frequency Sequences in Combinatorics*, Mem. Amer. Math. Soc., 413, 1989.

[3] F. BRENTI, *Expansions of chromatic polynomials and log-concavity*, Trans. Amer. Math. Soc., 332 (1992), pp. 729–756.

[4] F. BRENTI, G. F. ROYLE, AND D. G. WAGNER, *Location of zeros of chromatic and related polynomials of graphs*, Canad. J. Math., 46 (1994), pp. 55–80.

[5] C. J. COLBOURN, *The Combinatorics of Network Reliability*, Oxford University Press, New York, 1987.

[6] C. D. GODSIL AND I. GUTMAN, *On the theory of the matching polynomial*, J. Graph Theory, 5 (1981), pp. 137–144.

[7] J. GOLDMAN, J. JOICHI, AND D. WHITE, *Rook theory*, III. *Rook polynomials and the chromatic structure of graphs*, J. Combin. Theory Ser. B, 25 (1978), pp. 135–142.

[8] I. GUTMAN, *Acyclic systems with extremal Hückel π-electron energy*, Theoret. Chim. Acta, 45 (1977), pp. 79–87.

[9] G. H. HARDY, J. E. LITTLEWOOD, AND G. PÓLYA, *Inequalities*, Cambridge University Press, Cambridge, UK, 1959.

[10] L. H. HARPER, *Stirling behaviour is asymptotically normal*, Ann. Math. Statistics, 38 (1967), pp. 410–414.

[11] O. J. HEILMANN AND E. H. LIEB, *Theory of monomer-dimer systems*, Comm. Math. Phys., 25 (1972), pp. 190–232.
[12] I. KAPLANSKY AND J. RIORDAN, *The problem of the rook and its applications*, Duke Math. J., 13 (1946), pp. 259–268.
[13] S. KARLIN, *Total Positivity, Vol.* I, Stanford University Press, Stanford, CA, 1968.
[14] R. R. KORFHAGE, *σ-polynomials and graph colouring*, J. Combin. Theory Ser. B, 24 (1978), pp. 137–153.
[15] J. NEGGERS, *Representations of finite partially ordered sets*, J. Combin. Info. System Sci., 3 (1978), pp. 113–133.
[16] R. C. READ AND W. T. TUTTE, *Chromatic polynomials*, in Selected Topics in Graph Theory 3, L. W. Beineke and R. J. Wilson, eds., Academic Press, New York, 1988.
[17] R. P. STANLEY, *Ordered structures and partitions*, Mem. Amer. Math. Soc. 119, 1972.
[18] ———, *Enumerative combinatorics, Vol.* I, Wadwsworth & Brooks/Cole, Monterey, CA, 1986.
[19] ———, Personal communication, 1986.
[20] W. T. TUTTE, *On dichromatic polynomials*, J. Combin. Theory, 2 (1967), pp. 301–320.
[21] D. G. WAGNER, *The partition polynomial of a finite set system*, J. Combin. Theory Ser. A, 56 (1991), pp. 138–159.
[22] ———, *Enumeration of functions from posets to chains*, Europ. J. Combin., 13 (1992), pp. 313–324.
[23] ———, *Order series of labelled posets*, Order, 10 (1993), pp. 161–181.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 545–561, November 1996

003

VALUATED MATROID INTERSECTION I: OPTIMALITY CRITERIA*

KAZUO MUROTA†

Abstract. The independent assignment problem (or the weighted matroid intersection problem) is extended using Dress and Wenzel's matroid valuations, which are attached to the vertex set of the underlying bipartite graph as an additional weighting. Specifically, the problem considered is as follows: given a bipartite graph $G = (V^+, V^-; A)$ with arc weight $w : A \to \mathbf{R}$ and matroid valuations ω^+ and ω^- on V^+ and V^-, respectively, find a matching $M(\subseteq A)$ that maximizes $\sum\{w(a) \mid a \in M\} + \omega^+(\partial^+ M) + \omega^-(\partial^- M)$, where $\partial^+ M$ and $\partial^- M$ denote the sets of vertices in V^+ and V^- incident to M. As natural extensions of the previous results for the independent assignment problem, two optimality criteria are established: one in terms of potentials and the other in terms of negative cycles in an auxiliary graph.

Key words. weighted matroid intersection problem, independent assignment problem, valuated matroid, combinatorial optimization

AMS subject classifications. 90C35, 90C27, 90B80

1. Introduction.

The weighted matroid intersection problem and its extensions has played a major role in the theory of combinatorial optimization. (See, for instance, Edmonds [7], [8], Faigle [9], Fujishige [14], and Lawler [20].) One of its equivalent variants introduced by Iri and Tomizawa [17] is the independent assignment problem defined as follows. We are given a bipartite graph $G = (V^+, V^-; A)$, matroids $\mathbf{M}^+ = (V^+, \mathcal{B}^+)$ and $\mathbf{M}^- = (V^-, \mathcal{B}^-)$, and a weight function $w : A \to R$, where (V^+, V^-) is the bipartition of the vertex set of G, A is the arc set of G, $\mathbf{M}^+$ (resp., $\mathbf{M}^-$) is defined on V^+ (resp., V^-) in terms of the family of bases $\mathcal{B}^+$ (resp., $\mathcal{B}^-$), and R is a totally ordered additive group. (Typically $R = \mathbf{R}$ (reals), $\mathbf{Q}$ (rationals), or $\mathbf{Z}$ (integers).) The independent assignment problem is to find a matching $M(\subseteq A)$ that maximizes

$$w(M) \equiv \sum\{w(a) \mid a \in M\}, \tag{1.1}$$

subject to the constraint

$$\partial^+ M \in \mathcal{B}^+, \quad \partial^- M \in \mathcal{B}^-, \tag{1.2}$$

where $\partial^+ M$ (respectively, $\partial^- M$) denotes the set of vertices in V^+ (respectively, V^-) incident to M. The independent assignment problem has been shown to be a useful framework in which to formulate engineering problems in systems analysis. (See, e.g., Iri [16], Murota [22], and Recski [30].)

Recently, on the other hand, Dress and Wenzel [5], [6] introduced the notion of valuation on a matroid. A valuation on a matroid $\mathbf{M} = (V, \mathcal{B})$ is a function $\omega : \mathcal{B} \to R$ which enjoys the exchange property: for $B, B' \in \mathcal{B}$ and $u \in B - B'$, there exists $v \in B' - B$ such that $B - u + v \in \mathcal{B}$, $B' + u - v \in \mathcal{B}$, and

$$\omega(B) + \omega(B') \leq \omega(B - u + v) + \omega(B' + u - v). \tag{1.3}$$

A matroid equipped with a valuation is called valuated matroid.

* Received by the editors January 18, 1995; accepted for publication (in revised form) October 30, 1995.

† Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-01, Japan (murota@kurims.kyoto-u.ac.jp). This research was completed while the author was at Forschungsinstitut für Diskrete Mathematik, Universität Bonn.

A valuation ω can be induced from a weight function $\eta : V \to R$ and $\alpha \in R$ by

$$\omega(B) = \alpha + \sum\{\eta(u) \mid u \in B\} \quad \text{for } B \in \mathcal{B}. \tag{1.4}$$

Such a valuation will be called separable (called "essentially trivial" in [6]).

A (nonseparable) valuated matroid naturally arises from a polynomial matrix with coefficients from a field. Let $A(x)$ be an $m \times n$ matrix of rank m with each entry being a polynomial in a variable x, and let $\mathbf{M} = (V, \mathcal{B})$ denote the (linear) matroid defined on the column set V of $A(x)$ by the linear independence of the column vectors. Then a valuated matroid is obtained if $\omega(B)$ for $B \in \mathcal{B}$ is defined to be the degree in x of the determinant of the $m \times m$ submatrix with columns in B; i.e.,

$$\omega(B) = \deg_x \det A[B]. \tag{1.5}$$

In fact, the Grassmann–Plücker identity implies the exchange property of ω, as pointed out in Dress and Wenzel [5], [6] in a more algebraic term of "field valuation." Some examples of valuated matroids of a more combinatorial-geometrical flavor are reported in Terhalle [33], whereas Example 3.3 in §3 below shows another example arising from graphs. Details on valuated matroids are given in §3.

In this paper we consider an extension of the independent assignment problem to its valuated version. Namely we assume that $\mathbf{M}^+ = (V^+, \mathcal{B}^+)$ and $\mathbf{M}^- = (V^-, \mathcal{B}^-)$ are equipped with valuations $\omega^+ : \mathcal{B}^+ \to R$ and $\omega^- : \mathcal{B}^- \to R$ and consider the problem of finding a matching $M(\subseteq A)$ that maximizes

$$\Omega(M) \equiv w(M) + \omega^+(\partial^+ M) + \omega^-(\partial^- M),$$

subject to the constraint (1.2). We shall call this problem the *valuated independent assignment problem* (VIAP). This is a proper extension of the independent assignment problem, whereas it obviously reduces to the ordinary independent assignment problem in the case that ω^+ and ω^- are trivial valuations that vanish identically on $\mathcal{B}^+$ and $\mathcal{B}^-$, respectively, and also in a more general case of separable valuations.

In the present paper we establish two forms of optimality criteria for the valuated independent assignment problem by extending in a natural way the two well-known optimality criteria for the ordinary independent assignment problem. The first (Theorem 4.1) is in terms of potentials, as in Frank [10], and the second (Theorem 4.3) is in terms of negative cycles in an auxiliary graph, as in Fujishige [12]. (See also Fujishige [14] and Zimmermann [35], which give a similar condition for submodular flows.) The negative-cycle criterion yields a primal-type cycle-canceling algorithm for solving the valuated independent assignment problem, to be reported in part II [25], which is an extension of Fujishige's [12] and Zimmermann's [36] for the ordinary independent assignment problem.

The driving wheels for these extensions are the proper generalizations of the fundamental lemmas on the exchangeability in a matroid to those in a valuated matroid. Among others it should be mentioned that the so-called no-shortcut lemma (see Lemma 3.3 in §3 for a precise statement) is generalized to what we shall call unique-max lemma (Lemma 3.8 in §3). When specialized to a valuated matroid associated with a polynomial matrix as in (1.5), the no-shortcut lemma states that a triangular matrix having nonzero diagonal elements is nonsingular, whereas the unique-max lemma reveals a stronger property and that a square matrix having a unique maximum-degree transversal is nonsingular. (See Remark 3.2 for more about this.)

The objective of this paper is twofold. The first is purely theoretical within the field of combinatorial optimization. As compared with the richness of matroid optimizations (greedy algorithm, intersection/union, lexico-optimality, etc.), not much is known about valuated-matroid optimizations. All the known results center around greedy procedures for a single valuated matroid (cf. Dress and Wenzel [5], Dress and Terhalle [2]–[4], Murota [24]). The present results, along with the algorithms in [25], will contribute to the development of the theory of valuated-matroid optimization. This line of research is pursued further in the subsequent papers [26]–[28]; the optimality criteria are extended to the submodular flow problem in [27], duality theorems are established in [26], [28] in relation to convex analysis, and the matroid union operation is extended to valuations in [28].

The second objective is more application oriented. As explained above, the valuated matroid is a combinatorial abstraction of polynomial matrices. In view of the principal role of polynomial matrices in system engineering (see, e.g., Rosenbrock [31], Vidyasagar [34]) as well as the previous success in application of matroids to it, it is natural to hope for successful applications of valuated matroid to engineering problems. In this connection it should be emphasized that most of the significant applications of matroid theory have been related, more or less, to the matroid intersection problem. This paper will lay the foundation for future engineering applications. Some applications of the valuated matroid intersection to mixed matrices [22], which in fact have been the motivation of this paper, are discussed in [29].

2. Problem formulations. In this section we describe the problem and its variants. Suppose we are given a bipartite graph $G = (V^+, V^-; A)$, valuated matroids $\mathbf{M}^+ = (V^+, \mathcal{B}^+, \omega^+)$ and $\mathbf{M}^- = (V^-, \mathcal{B}^-, \omega^-)$, and a weight function $w : A \to R$. The valuated independent assignment problem is the following.

VALUATED INDEPENDENT ASSIGNMENT PROBLEM. Find a matching $M(\subseteq A)$ that maximizes

$$\Omega(M) \equiv w(M) + \omega^+(\partial^+ M) + \omega^-(\partial^- M) \tag{2.1}$$

subject to the constraint

$$\partial^+ M \in \mathcal{B}^+, \quad \partial^- M \in \mathcal{B}^-. \tag{2.2}$$

Clearly the two matroids must have the same rank for the feasibility of this problem. It is sometimes convenient to extend the domain of the definition of ω^+ to 2^{V^+} by simply setting $\omega^+(B) = -\infty$ for $B \subseteq V^+$ with $B \notin \mathcal{B}^+$ and similarly for ω^-. Then the constraint (2.2) will be implicit in the objective function $\Omega(M)$.

The above problem reduces to the independent assignment problem if the valuations are trivial with $\omega^\pm(B) = 0$ for $B \in \mathcal{B}^\pm$ and reduces further to the conventional assignment problem if the matroids are trivial or free with $\mathcal{B}^\pm = 2^{V^\pm}$.

Just as the weighted matroid intersection and partition problems may be regarded as special cases of the independent assignment problem, the following three problems fall into the category of our problem. Suppose now we are given a pair of valuated matroids $\mathbf{M}^1 = (V, \mathcal{B}^1, \omega^1)$ and $\mathbf{M}^2 = (V, \mathcal{B}^2, \omega^2)$ defined on a common ground set V and a weight function $w : V \to R$.

INTERSECTION PROBLEM. Find a common base $B \in \mathcal{B}^1 \cap \mathcal{B}^2$ that maximizes $w(B) + \omega^1(B) + \omega^2(B)$. (In case the valuations are separable as (1.4) this problem reduces to the usual optimal common base problem.)

DISJOINT BASES PROBLEM. Find disjoint bases B^1 and B^2 (i.e., $B^1 \cap B^2 = \emptyset$, $B^1 \in \mathcal{B}^1$, and $B^2 \in \mathcal{B}^2$) that maximize $\omega^1(B^1) + \omega^2(B^2)$.

PARTITION PROBLEM. Find a partition $(B, V-B)$ of V that maximizes $\omega^1(B)+\omega^2(V-B)$.

As a matter of course, the disjoint bases problem for more than two valuated matroids can also be formulated as a valuated independent assignment problem. The partition problem is an intersection problem in disguise, since it is the intersection problem for $\mathbf{M}^1$ and $(\mathbf{M}^2)^*$, the dual of $\mathbf{M}^2$, whose valuation is defined by $(\omega^2)^*(B) = \omega^2(V-B)$.

Remark 2.1. The valuated independent assignment problem can easily be generalized to an independent linkage-type problem (cf. Fujishige [13], Iri [15]). The underlying bipartite graph is replaced with an arbitrary (directed or undirected) graph having specified entrance and exit vertex sets, on which valuated matroids are defined, and matchings are replaced by linkings from the entrance to the exit. The optimality criteria of the present paper, mutatis mutandis, are easily shown to remain valid for this linkage-type problem.

In the ordinary independent assignment problem the constraint imposed on a matching M is more often that $\partial^{\pm}M$ be independent in $\mathbf{M}^{\pm}$ than that $\partial^{\pm}M$ be a base in $\mathbf{M}^{\pm}$. This motivates us to consider the following problem parametrized by an integer k:

VIAP(k). Maximize

$$\Omega(M, B^+, B^-) \equiv w(M) + \omega^+(B^+) + \omega^-(B^-),$$

subject to the constraint that M is a matching of size k, and

$$\partial^+ M \subseteq B^+ \in \mathcal{B}^+, \quad \partial^- M \subseteq B^- \in \mathcal{B}^-.$$

In fact, the primal-dual type augmenting algorithm of [25] consists of solving this problem successively for $k = 0, 1, 2, \ldots$. The optimality criteria for VIAP(k) are derived in §5.

3. Properties of a valuated matroid.

3.1. Examples. The first two examples are already mentioned in the Introduction.

Example 3.1. Let $\mathbf{M} = (V, \mathcal{B})$ be a matroid. For $\eta : V \to R$ and $\alpha \in R$,

$$\omega(B) = \alpha + \sum\{\eta(u) \mid u \in B\} \qquad (B \in \mathcal{B})$$

is a matroid valuation. Such ω is called a separable valuation.

Example 3.2. Let $A(x)$ be an $m \times n$ matrix of rank m with each entry being a polynomial (or rational function) in a variable x, and let $\mathbf{M} = (V, \mathcal{B})$ denote the (linear) matroid defined on the column set V of $A(x)$ by the linear independence of the column vectors. Then $\omega : \mathcal{B} \to \mathbf{Z}$ defined by $\omega(B) = \deg_x \det A[B]$ $(B \in \mathcal{B})$ is a matroid valuation (see Dress and Wenzel [6] for the proof), where $\deg_x(f/g) = \deg_x f - \deg_x g$ for two polynomials f and g in x. An example of nonseparable valuation of this kind is provided by

$$A(x) = \begin{pmatrix} x+1 & x & 1 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix}.$$

Example 3.3. In addition to the above two constructions that can be found in the literature [6] we point out here another instance of a (nonseparable) valuated matroid

that arises from the minimum cost of a linking in a graph. Let $G = (\overline{V}, \overline{A})$ be a directed graph having no self-loops and S and T be disjoint subsets of the vertex set $\overline{V}$. By L we denote (the arc set of) a Menger-type vertex-disjoint linking from S to T and by $\partial^+ L$ the set of its initial vertices (in S); put $U = \overline{V} - (S \cup T)$. As is well known, $\mathcal{B} = \{\partial^+ L \mid L \in \mathcal{L}\}$, where $\mathcal{L}$ denotes the family of maximum linkings, defines a matroid $\mathbf{M} = (S, \mathcal{B})$. Given a cost function $c : \overline{A} \to \mathbf{Z}$ such that every cycle has a nonnegative cost,

$$\omega(B) = -\min\left\{\sum_{a \in L} c(a) \mid \partial^+ L = B, L \in \mathcal{L}\right\} \qquad (B \in \mathcal{B})$$

is a matroid valuation.

To see this, first note that, by the max-flow min-cut theorem, we may assume $\partial^- L = T$, where $\partial^- L$ designates the set of the terminal vertices of L (in T). Consider a rational function matrix, say $A(x) = (A_{ij}(x))$, in variable x with the row set indexed by $T \cup U$ and the column set by $S \cup U$ defined by

$$A_{ij}(x) = \begin{cases} 1 & (i = j \in U), \\ \alpha_{ij} x^{-c(j,i)} & ((j,i) \in \overline{A}), \end{cases}$$

where $\{\alpha_{ij} \mid (j,i) \in \overline{A}\}$ is an algebraically independent set of real numbers. Then we have $\omega(B) = \deg_x \det A[B \cup U]$, which is a version of Example 3.2. An example of a nonseparable valuation of this kind is provided by $G = (\overline{V}, \overline{A})$ with $\overline{V} = S \cup T$, $S = \{s_1, s_2, s_3, s_4\}$, $T = \{t_1, t_2\}$, $\overline{A} = \{(s_1, t_1), (s_2, t_2), (s_3, t_1), (s_3, t_2), (s_4, t_1), (s_4, t_2)\}$, $c(a) = 0$ except for $c(s_3, t_2) = -1$, and $c(s_4, t_2) = -2$. See also [27, Ex. 2.3], in which a flow-type generalization is given.

3.2. Basic properties. This subsection describes some relevant results of Dress and Wenzel [5], [6] on the maximization of a matroid valuation.

Let $\mathbf{M} = (V, \mathcal{B}, \omega)$ be a valuated matroid of rank r. For $B \in \mathcal{B}$ and $v \in V - B$, we denote by $C(B, v)$ the unique circuit contained in $B + v$ (= the fundamental circuit of v relative to B). For $B \in \mathcal{B}$, $v \in V - B$, and $u \in C(B, v)$, we define

$$\omega(B, u, v) = \omega(B - u + v) - \omega(B). \tag{3.1}$$

For convenience we set

$$\omega(B, u, v) = -\infty \quad \text{for } u \notin C(B, v).$$

This is also a consequence of our former convention to put $\omega(B') = -\infty$ for $B' \notin \mathcal{B}$.

The following lemma is most fundamental, showing that the local optimality implies the global optimality.

LEMMA 3.1 (see [5], [6]). *Let $B \in \mathcal{B}$. Then $\omega(B) \geq \omega(B')$ for any $B' \in \mathcal{B}$ if and only if*

$$\omega(B, u, v) \leq 0 \quad \text{for any } (u, v) \text{ with } u \in C(B, v). \tag{3.2}$$

Proof. The original proof is by induction on $|B - B'|$. An alternative proof is given later in Remark 3.1. □

For the maximization of ω, the greedy algorithm of [5] starts with an arbitrary base $B_0 = \{u_1, u_2, \ldots, u_r\} \in \mathcal{B}$ and repeats the following for $k = 1, 2, \ldots, r$: find $v_k \in V - B_{k-1}$ such that

$$\omega(B_{k-1} - u_k + v_k) \geq \omega(B_{k-1} - u_k + v) \quad (\forall v \in V - B_{k-1}),$$

and put $B_k = B_{k-1} - u_k + v_k$. Then B_r can be shown to be optimal. In this way an optimal base (maximizing ω) can be found with $r(|V| - r) + 1$ function evaluations of ω.

For $\eta : V \to R$ we define $\omega[\eta] : \mathcal{B} \to R$ (or $2^V \to R \cup \{-\infty\}$) by

$$\omega[\eta](B) = \omega(B) + \sum\{\eta(u) \mid u \in B\}. \tag{3.3}$$

This is again a valuation on $\mathbf{M}$. This operation is called a similarity transformation. A valuation ω is separable (or "essentially trivial" in the terminology of [6]) if and only if $\omega[\eta](B) =$ constant $(\forall B \in \mathcal{B})$ for some $\eta : V \to R$.

$\mathbf{M}^* = (V, \mathcal{B}^*, \omega^*)$, defined by

$$\mathcal{B}^* = \{B \mid V - B \in \mathcal{B}\}, \qquad \omega^*(B) = \omega(V - B),$$

is again a valuated matroid, called the dual of $\mathbf{M} = (V, \mathcal{B}, \omega)$.

3.3. Further exchange properties. We shall establish a number of lemmas concerning basis exchanges in a single valuated matroid. They will play the key roles throughout this paper.

For $B \in \mathcal{B}$ and $B' \subseteq V$ we consider the exchangeability graph, denoted $G(B, B')$, in the usual sense in matroid theory. Namely, $G(B, B')$ is a bipartite graph having $(B - B', B' - B)$ as the vertex bipartition and $\{(u, v) \mid u \in B - B', v \in B' - B, u \in C(B, v)\}$ as the arc set. The following fact (Brualdi [1]) is well known in matroid theory.

LEMMA 3.2. *Let $B \in \mathcal{B}$. If B' is also a base, then $G(B, B')$ has a perfect matching.*

The converse of the above statement is not always true. A partial converse is the key property underlying the (weighted or unweighted) matroid intersection algorithm and is known as the no-shortcut lemma. (For this name we refer to Kung [19]; see Iri and Tomizawa [17, Lem. 2], Krogdahl [18], Lawler [20, Lem. 3.1 of Chap. 8] and Schrijver [32, Thm. 4.3].) It can be stated as follows, in a form suitable for its extension to a valuated matroid.

LEMMA 3.3 (no-shortcut lemma). *Let $B \in \mathcal{B}$ and $B' \subseteq V$ with $|B'| = |B|$. If there exists exactly one perfect matching in $G(B, B')$, then $B' \in \mathcal{B}$.*

To capture the exchangeability with valuations, we need quantitative extensions of the above statements. To this end we attach "arc weight" $\omega(B, u, v)$ of (3.1) to each arc (u, v) of $G(B, B')$ and denote by $\widehat{\omega}(B, B')$ the maximum weight of a perfect matching in $G(B, B')$ with respect to the arc weight $\omega(B, u, v)$. Lemma 3.2 is extended as follows.

LEMMA 3.4 (upper-bound lemma). *For $B, B' \in \mathcal{B}$,*

$$\omega(B') \leq \omega(B) + \widehat{\omega}(B, B'). \tag{3.4}$$

Proof. For any $u_1 \in B - B'$ there exists $v_1 \in B' - B$ with

$$\omega(B) + \omega(B') \leq \omega(B - u_1 + v_1) + \omega(B' + u_1 - v_1),$$

which can be rewritten as

$$\omega(B') \leq \omega(B, u_1, v_1) + \omega(B'_2)$$

with $B'_2 = B' + u_1 - v_1$. By the same argument applied to (B, B'_2) we obtain

$$\omega(B'_2) \leq \omega(B, u_2, v_2) + \omega(B'_3)$$

for some $u_2 \in (B - B') - u_1$ and $v_2 \in (B' - B) - v_1$, where $B'_3 = B'_2 + u_2 - v_2 = B' - \{u_1, u_2\} + \{v_1, v_2\}$. Hence

$$\omega(B') \leq \omega(B'_3) + \sum_{i=1}^{2} \omega(B, u_i, v_i).$$

Repeating this process we arrive at

$$\omega(B') \leq \omega(B) + \sum_{i=1}^{m} \omega(B, u_i, v_i) \leq \omega(B) + \widehat{\omega}(B, B'),$$

where $m = |B - B'| = |B' - B|$, $B - B' = \{u_1, \ldots, u_m\}$, and $B' - B = \{v_1, \ldots, v_m\}$. □

Remark 3.1. Lemma 3.4 (upper-bound lemma) gives an alternative proof for the optimality condition given in Lemma 3.1. The necessity of (3.2) is obvious. For sufficiency take any $B' \in \mathcal{B}$ and consider $G(B, B')$. The condition (3.2) is equivalent to all the arcs having nonpositive weights. Hence $\widehat{\omega}(B, B') \leq 0$, which implies $\omega(B') \leq \omega(B)$ by Lemma 3.4.

In Lemma 3.4 it is natural to ask for a (sufficient) condition under which the bound (3.4) is tight. Comparison of Lemmas 3.3 and 3.4 will suggest the following.

UNIQUE-MAX CONDITION. There exists exactly one maximum-weight perfect matching in $G(B, B')$.

In what follows we shall show that this is indeed a sufficient condition for the tightness. (See Lemma 3.8.)

First we note the following fact, rephrasing the unique-max condition in terms of "potential" or "dual variable."

LEMMA 3.5. *Let $B \in \mathcal{B}$ and $B' \subseteq V$ with $|B' - B| = |B - B'| = m$.*

(1) *$G(B, B')$ has a perfect matching if and only if there exist $\widehat{p} : (B - B') \cup (B' - B) \to R$ and indexings of the elements of $B - B'$ and $B' - B$, say $B - B' = \{u_1, \ldots, u_m\}$ and $B' - B = \{v_1, \ldots, v_m\}$, such that*

$$\omega(B, u_i, v_j) - \widehat{p}(u_i) + \widehat{p}(v_j) \begin{cases} = 0 & (1 \leq i = j \leq m), \\ \leq 0 & (1 \leq i, j \leq m). \end{cases} \tag{3.5}$$

(2) *The pair (B, B') satisfies the unique-max condition if and only if there exist $\widehat{p} : (B - B') \cup (B' - B) \to R$ and indexings of the elements of $B - B'$ and $B' - B$, say $B - B' = \{u_1, \ldots, u_m\}$ and $B' - B = \{v_1, \ldots, v_m\}$, such that*

$$\omega(B, u_i, v_j) - \widehat{p}(u_i) + \widehat{p}(v_j) \begin{cases} = 0 & (1 \leq i = j \leq m), \\ \leq 0 & (1 \leq j < i \leq m), \\ < 0 & (1 \leq i < j \leq m). \end{cases} \tag{3.6}$$

Proof. This is an immediate corollary of the complementary slackness well known in matching theory. (See, e.g., Lawler [20], and Lovász and Plummer [21].) Let $M = \{(u_i, v_i) \mid i = 1, \ldots, m\}$ be a maximum-weight perfect matching and $\widehat{p}$ be an optimal potential (or dual variable). Then $\omega(B, u, v) - \widehat{p}(u) + \widehat{p}(v) \leq 0$ for all arcs (u, v). Call an arc tight if this inequality holds true with equality and define G^* to be the subgraph of G consisting of tight arcs. The complementary slackness says that maximum-weight perfect matchings in $G(B, B')$ are in one-to-one correspondence with perfect matchings in G^*. □

LEMMA 3.6. *Let $B \in \mathcal{B}$ and u, u°, v, v° be four distinct elements with $\{u, u^\circ\} \subseteq B$, $\{v, v^\circ\} \subseteq V - B$, and let $B' = B - \{u, u^\circ\} + \{v, v^\circ\}$. Assume that $M = \{(u, v), (u^\circ, v^\circ)\}$ is the unique maximum-weight perfect matching in $G(B, B')$.*

(1) *$B' \in \mathcal{B}$ and $\omega(B') = \omega(B) + \widehat{\omega}(B, B')$.*

(2) *For $B^\circ = B - u^\circ + v^\circ$ we have*

$$\begin{aligned}
\omega(B^\circ, u, v) &= \omega(B, u, v), \\
\omega(B^\circ, u, u^\circ) &= \omega(B, u, v^\circ) - \omega(B, u^\circ, v^\circ), \\
\omega(B^\circ, v^\circ, v) &= \omega(B, u^\circ, v) - \omega(B, u^\circ, v^\circ).
\end{aligned}$$

Proof. (1) Putting $B^* = B - u + v$ we see

$$\omega(B^*) + \omega(B^\circ) = \omega(B, u, v) + \omega(B, u^\circ, v^\circ) + 2\omega(B) = \widehat{\omega}(B, B') + 2\omega(B). \tag{3.7}$$

By applying the exchange axiom (1.3) to (B°, B^*) with $u \in B^\circ - B^*$, we have

$$\omega(B^*) + \omega(B^\circ) \leq \omega(B^* - v' + u) + \omega(B^\circ + v' - u)$$

for some $v' \in B^* - B^\circ = \{u^\circ, v\}$. Combining this with (3.7) we obtain

$$\widehat{\omega}(B, B') + 2\omega(B) \leq \omega(B^* - v' + u) + \omega(B^\circ + v' - u). \tag{3.8}$$

Suppose that $v' = u^\circ$. Then

$$\begin{aligned}
\text{RHS of (3.8)} &= \omega(B^* - u^\circ + u) + \omega(B^\circ + u^\circ - u) \\
&= \omega(B - u^\circ + v) + \omega(B - u + v^\circ) \\
&= \omega(B, u^\circ, v) + \omega(B, u, v^\circ) + 2\omega(B).
\end{aligned}$$

This means that $M' = \{(u^\circ, v), (u, v^\circ)\}$ is also a maximum-weight perfect matching in $G(B, B')$, a contradiction to the uniqueness of M.

Therefore we have $v' = v$ in (3.8), and then

$$\text{RHS of (3.8)} = \omega(B^* - v + u) + \omega(B^\circ + v - u) = \omega(B) + \omega(B').$$

Hence follows $\omega(B) + \widehat{\omega}(B, B') \leq \omega(B')$. The reverse inequality has already been shown in the upper-bound lemma (Lemma 3.4). Note that $B' \in \mathcal{B}$ follows from $\omega(B') \neq -\infty$.

(2) By straightforward calculations as follows,

$$\begin{aligned}
\omega(B^\circ, u, v) &= \omega(B - u^\circ + v^\circ - u + v) - \omega(B - u^\circ + v^\circ) \\
&= \omega(B') - \omega(B) - \omega(B, u^\circ, v^\circ) \\
&= \widehat{\omega}(B, B') - \omega(B, u^\circ, v^\circ) \\
&= \omega(B, u, v), \\
\omega(B^\circ, u, u^\circ) &= \omega(B - u + v^\circ) - \omega(B - u^\circ + v^\circ) \\
&= \omega(B, u, v^\circ) - \omega(B, u^\circ, v^\circ), \\
\omega(B^\circ, v^\circ, v) &= \omega(B - u^\circ + v) - \omega(B - u^\circ + v^\circ) \\
&= \omega(B, u^\circ, v) - \omega(B, u^\circ, v^\circ). \qquad \square
\end{aligned}$$

LEMMA 3.7. *Let $B \in \mathcal{B}$ and $B' \subseteq V$ with $|B'| = |B|$. If there exists exactly one maximum-weight perfect matching M in $G(B, B')$, then for any $(u^\circ, v^\circ) \in M$ the following hold true.*

(1) $B^\circ \equiv B - u^\circ + v^\circ \in \mathcal{B}$.

(2) *There exists exactly one maximum-weight perfect matching in* $G(B^\circ, B')$.

(3) $\widehat{\omega}(B^\circ, B') = \widehat{\omega}(B, B') - \omega(B, u^\circ, v^\circ)$.

Proof. (1) This is obvious.

(2) Using the notation in Lemma 3.5 we have $M = \{(u_i, v_i) \mid i = 1, \ldots, m\}$ and $(u^\circ, v^\circ) = (u_k, v_k)$ for some k. Put

$$B_{ij} = B^\circ - u_i + v_j = B - \{u_i, u^\circ\} + \{v_j, v^\circ\}$$

for $i \neq k$, $j \neq k$. It then follows from (3.4) and (3.6) that

$$\begin{aligned}
&\omega(B^\circ, u_i, v_j) \\
&= \omega(B_{ij}) - \omega(B^\circ) \\
&\leq \widehat{\omega}(B, B_{ij}) - \omega(B, u_k, v_k) \\
&= \max\left(\omega(B, u_k, v_k) + \omega(B, u_i, v_j), \omega(B, u_i, v_k) + \omega(B, u_k, v_j)\right) - \omega(B, u_k, v_k) \\
&\leq [\widehat{p}(u_i) + \widehat{p}(u_k) - \widehat{p}(v_j) - \widehat{p}(v_k)] - [\widehat{p}(u_k) - \widehat{p}(v_k)] \\
&= \widehat{p}(u_i) - \widehat{p}(v_j),
\end{aligned}$$

where the second inequality is strict for $i < j$. For $i = j$, on the other hand, both inequalities are satisfied with equalities, since $G(B, B_{ii})$ has a unique maximum-weight perfect matching $\{(u_i, v_i), (u^\circ, v^\circ)\}$ and Lemma 3.6 implies $\omega(B^\circ, u_i, v_i) = \omega(B, u_i, v_i) = \widehat{p}(u_i) - \widehat{p}(v_i)$. Thus, the potential $\widehat{p}$ for (B, B') serves as a certificate of the unique-max condition also for (B°, B').

(3) $\widehat{\omega}(B^\circ, B') = \sum_{i \neq k} (\widehat{p}(u_i) - \widehat{p}(v_i)) = \widehat{\omega}(B, B') - \omega(B, u^\circ, v^\circ)$. □

We are now in a position to state the main result of this section, the unique-max lemma, which is a quantitative extension of the no-shortcut lemma.

LEMMA 3.8 (unique-max lemma). *Let* $B \in \mathcal{B}$ *and* $B' \subseteq V$ *with* $|B'| = |B|$. *If there exists exactly one maximum-weight perfect matching in* $G(B, B')$, *then* $B' \in \mathcal{B}$ *and*

$$\omega(B') = \omega(B) + \widehat{\omega}(B, B'). \tag{3.9}$$

Proof. This is proved by induction on $m = |B - B'|$. The case of $m = 1$ is obvious. So assume $m \geq 2$. Take any (u°, v°) contained in the unique maximum-weight perfect matching, and put $B^\circ = B - u^\circ + v^\circ$. (B°, B') satisfies the unique-max condition by Lemma 3.7(2), and we have

$$\omega(B') = \omega(B^\circ) + \widehat{\omega}(B^\circ, B')$$

by the induction hypothesis. By Lemma 3.7(3) we see

$$\widehat{\omega}(B^\circ, B') = \widehat{\omega}(B, B') - \omega(B, u^\circ, v^\circ)$$

while $\omega(B^\circ) = \omega(B) + \omega(B, u^\circ, v^\circ)$ by definition. Hence follows (3.9). □

Remark 3.2. Some remark is in order regarding the relationship between the no-shortcut condition (= uniqueness of a perfect matching in $G(B, B')$) and the unique-max condition (= uniqueness of the *maximum-weight* perfect matching in $G(B, B')$). Obviously the former implies the latter, and not conversely in general. For a separable valuation (cf. (1.4) and §3.2), however, these two conditions are equivalent, and consequently the unique-max lemma reduces to the no-shortcut lemma. See also Frank [10, Lem. 2] in this connection.

Remark 3.3. An alternative proof of the unique-max lemma was suggested by Sebő after the submission of the first draft. This proof makes use of the no-shortcut lemma in contrast to the above proof. Let $\widehat{p} : (B - B') \cup (B' - B) \to R$ be as in Lemma 3.5 and extend it to $\widehat{p} : V \to R$ by defining $\widehat{p}(u) = +M$ for $u \in B \cap B'$ and $\widehat{p}(v) = -M$ for $v \in V - (B \cup B')$ with a sufficiently large $M > 0$. It follows from the exchange property (1.3) that the family of the maximizers of $\omega[\widehat{p}]$,

$$\mathcal{B}^\circ = \{B^\circ \in \mathcal{B} \mid \omega[\widehat{p}](B^\circ) \geq \omega[\widehat{p}](B'') \ (B'' \in \mathcal{B})\},$$

forms the basis family of a matroid, say $\mathbf{M}^\circ = (V, \mathcal{B}^\circ)$. We claim that $B \in \mathcal{B}^\circ$. To see this, first note that $\omega[\widehat{p}](B'') - \omega[\widehat{p}](B) \leq \omega(B'') - \omega(B) - M \leq 0$ unless $B \cap B' \subseteq B'' \subseteq B \cup B'$. If $B \cap B' \subseteq B'' \subseteq B \cup B'$, on the other hand, we have $\omega[\widehat{p}](B'') - \omega[\widehat{p}](B) \leq 0$ by the upper-bound lemma and the inequality

$$\omega[\widehat{p}](B, u, v) = \omega(B, u, v) - \widehat{p}(u) + \widehat{p}(v) \leq 0 \qquad (u \in B - B'', v \in B'' - B).$$

We also claim that the exchangeability graph $G^\circ(B, B')$ in $\mathbf{M}^\circ$ has a unique perfect matching, since $B - u_i + v_i \in \mathcal{B}^\circ$ $(1 \leq i \leq m)$ and $B - u_i + v_j \notin \mathcal{B}^\circ$ $(1 \leq i < j \leq m)$ by (3.6). By applying the no-shortcut lemma to the given pair (B, B') in the matroid $\mathbf{M}^\circ = (V, \mathcal{B}^\circ)$, we obtain $B' \in \mathcal{B}^\circ$, which means $\omega[\widehat{p}](B') = \omega[\widehat{p}](B)$, i.e., $\omega(B') = \omega(B) + \sum_{i=1}^m \widehat{p}(u_i) - \sum_{i=1}^m \widehat{p}(v_i) = \omega(B) + \widehat{\omega}(B, B')$.

The following lemma is used in part II [25] in justifying a variant of the cycle-canceling algorithm.

LEMMA 3.9. *Under the same assumption as in Lemma* 3.8, *let* $\widehat{p}$, u_i, *and* v_j *be as in Lemma* 3.5. *Then*

$$\omega(B', v_j, u_i) \leq \widehat{p}(v_j) - \widehat{p}(u_i) \qquad (1 \leq i, j \leq m).$$

Proof. Putting $B'_{ij} = B' - v_j + u_i$ and using Lemma 3.4, Lemma 3.8, and (3.6) we see that

$$\begin{aligned}
\omega(B', v_j, u_i) &= \omega(B'_{ij}) - \omega(B') \leq \widehat{\omega}(B, B'_{ij}) - \widehat{\omega}(B, B') \\
&\leq \left[\sum_{k \neq i} \widehat{p}(u_k) - \sum_{k \neq j} \widehat{p}(v_k)\right] - \left[\sum_{k=1}^m \widehat{p}(u_k) - \sum_{k=1}^m \widehat{p}(v_k)\right] \\
&= \widehat{p}(v_j) - \widehat{p}(u_i). \qquad \square
\end{aligned}$$

4. Optimality criteria.

4.1. Theorems. Two optimality criteria are given for the valuated independent assignment problem (2.1)–(2.2) on $G = (V^+, V^-; A)$ with valuated matroids $\mathbf{M}^+ = (V^+, \mathcal{B}^+, \omega^+)$, $\mathbf{M}^- = (V^-, \mathcal{B}^-, \omega^-)$, and weight function $w : A \to R$, where R is a totally ordered additive group (e.g., $R = \mathbf{R}$, $\mathbf{Q}$, or $\mathbf{Z}$). Both of these criteria are natural extensions of the corresponding results for the ordinary independent assignment problem, which have been extended also for the submodular flow problem. (See Frank [10], [11], Fujishige [12], [14], and Zimmermann [35].) The proofs are postponed to §4.2.

The first theorem refers to a "potential" function. It may be emphasized that in the case of $R = \mathbf{Z}$ the integrality of p is a part of the assertion.

THEOREM 4.1. (1) *An independent assignment* M *in* G *is optimal for the valuated independent assignment problem* (2.1)–(2.2) *if and only if there exists a "potential" function* $p : V^+ \cup V^- \to R$ *such that*

(i) $w(a) - p(\partial^+ a) + p(\partial^- a) \begin{cases} \leq 0 & (a \in A), \\ = 0 & (a \in M), \end{cases}$

(ii) *$\partial^+ M$ is a maximum-weight base of $\mathbf{M}^+$ with respect to $\omega^+[p^+]$,*

(iii) *$\partial^- M$ is a maximum-weight base of $\mathbf{M}^-$ with respect to $\omega^-[-p^-]$,*

where $p^\pm$ is the restriction of p to $V^\pm$ and $\omega^+[p^+]$ (resp., $\omega^-[-p^-]$) is the similarity transformation defined in (3.3); *namely,*

$$\omega^+[p^+](B^+) = \omega^+(B^+) + \sum\{p(u) \mid u \in B^+\} \qquad (B^+ \subseteq V^+),$$

$$\omega^-[-p^-](B^-) = \omega^-(B^-) - \sum\{p(u) \mid u \in B^-\} \qquad (B^- \subseteq V^-).$$

(2) *Let p be a potential that satisfies* (i)–(iii) *above for some (optimal) independent assignment $M = M_0$. An independent assignment M' is optimal if and only if it satisfies* (i)–(iii) *(with M replaced by M').*

The optimality condition for the intersection problem deserves a separate statement, in a form of Frank's weight splitting [10], though it is an immediate corollary of the above theorem. Recall that the intersection problem is to maximize $w(B) + \omega^1(B) + \omega^2(B)$ for a pair of valuated matroids $\mathbf{M}^1 = (V, \mathcal{B}^1, \omega^1)$ and $\mathbf{M}^2 = (V, \mathcal{B}^2, \omega^2)$ and a weight function $w : V \to R$.

THEOREM 4.2. *A common base B of $\mathbf{M}^1 = (V, \mathcal{B}^1, \omega^1)$ and $\mathbf{M}^2 = (V, \mathcal{B}^2, \omega^2)$ maximizes $w(B) + \omega^1(B) + \omega^2(B)$ if and only if there exist $w^1, w^2 : V \to R$ such that*

(i) *["weight splitting"] $w(v) = w^1(v) + w^2(v)$ $(v \in V)$,*

(ii) *B is a maximum-weight base of $\mathbf{M}^1$ with respect to $\omega^1[w^1]$,*

(iii) *B is a maximum-weight base of $\mathbf{M}^2$ with respect to $\omega^2[w^2]$,*

where $\omega^1[w^1]$ (resp., $\omega^2[w^2]$) is the similarity transformation defined in (3.3).

To describe the second criterion we need to introduce an auxiliary graph $\widetilde{G}_M = (\widetilde{V}, \widetilde{A})$ associated with an independent assignment M. We put $B^+ = \partial^+ M$, $B^- = \partial^- M$, and denote by $C^\pm(\cdot, \cdot)$ the fundamental circuit in $\mathbf{M}^\pm$. The vertex set $\widetilde{V}$ of $\widetilde{G}_M$ is given by $\widetilde{V} = V^+ \cup V^-$ and the arc set $\widetilde{A}$ consists of four disjoint parts:

$$\widetilde{A} = A^\circ \cup M^\circ \cup A^+ \cup A^-,$$

where

$$\begin{aligned} A^\circ &= \{a \mid a \in A\} \qquad (\text{copy of } A), \\ M^\circ &= \{\overline{a} \mid a \in M\} \qquad (\overline{a}\text{: reorientation of } a), \\ A^+ &= \{(u, v) \mid u \in B^+, v \in V^+ - B^+, u \in C^+(B^+, v)\}, \\ A^- &= \{(v, u) \mid u \in B^-, v \in V^- - B^-, u \in C^-(B^-, v)\}. \end{aligned}$$

In addition, arc length $\gamma_M(a)$ $(a \in \widetilde{A})$ is defined by

$$\gamma_M(a) = \begin{cases} -w(a) & (a \in A^\circ), \\ w(\overline{a}) & (a = (u, v) \in M^\circ, \overline{a} = (v, u) \in M), \\ -\omega^+(B^+, u, v) & (a = (u, v) \in A^+), \\ -\omega^-(B^-, u, v) & (a = (v, u) \in A^-), \end{cases} \tag{4.1}$$

where $\omega^+(B^+, u, v)$ and $\omega^-(B^-, u, v)$ are defined as in (3.1). We call a directed cycle of negative length a negative cycle.

THEOREM 4.3. *An independent assignment M in G is optimal for the valuated independent assignment problem* (2.1)–(2.2) *if and only if there exists in $\widetilde{G}_M$ no negative cycle with respect to the arc length γ_M.*

Remark 4.1. The exchangeability graphs $G(B^+, V^+ - B^+)$ for $\mathbf{M}^+$ and $G(B^-, V^- - B^-)$ for $\mathbf{M}^-$ introduced in §3.3 are embedded in $\widetilde{G}_M$. Namely, they can be identified with the subgraphs (V^+, A^+) and (V^-, A^-), respectively. Note, however, that the arc weight is the negative of the arc length. □

Remark 4.2. The optimality criterion in Theorem 4.2 can be reformulated as a Fenchel-type duality between the matroid valuations and their conjugate functions, as reported in [26]. It is also mentioned that Theorem 4.2 is extended for the submodular flow problem in [27].

4.2. Proofs. We are to prove the equivalence of the following three conditions for an independent assignment M:

(OPT): M is optimal.

(NNC): There is no negative cycle in $\widetilde{G}_M$.

(POT): There exists a potential p with (i)–(iii) in Theorem 4.1.

We prove (OPT) $\Rightarrow$ (NNC) $\Rightarrow$ (POT) $\Rightarrow$ (OPT) and finally the second part of Theorem 4.1. We abbreviate γ_M to γ whenever convenient.

(OPT) $\Rightarrow$ (NNC): Suppose $\widetilde{G}_M$ has a negative cycle. Let Q ($\subseteq \widetilde{A}$) be the arc set of a negative cycle having the smallest number of arcs, and put

$$\overline{B}^+ = B^+ - \{\partial^+ a \mid a \in Q \cap A^+\} + \{\partial^- a \mid a \in Q \cap A^+\}, \tag{4.2}$$

$$\overline{B}^- = B^- - \{\partial^- a \mid a \in Q \cap A^-\} + \{\partial^+ a \mid a \in Q \cap A^-\}, \tag{4.3}$$

where $B^+ = \partial^+ M$ and $B^- = \partial^- M$ as before.

LEMMA 4.4. *$(B^+, \overline{B}^+)$ and $(B^-, \overline{B}^-)$ satisfy the unique-max condition in $\mathbf{M}^+$ and $\mathbf{M}^-$, respectively.*

Proof. We prove the claim for $(B^+, \overline{B}^+)$ by adapting Fujishige's proof technique developed in [12] (which can be found also in [14, Lem. 5.4]).

Take a maximum-weight perfect matching $M' = \{(u_i, v_i) \mid i = 1, \ldots, m\}$ (where $m = |B^+ - \overline{B}^+|$) in the exchangeability graph $G(B^+, \overline{B}^+)$ for $\mathbf{M}^+$ as well as the potential function $\widehat{p}$ in Lemma 3.5. Then M' is a subset of

$$A^* = \{(u, v) \mid u \in B^+ - \overline{B}^+, v \in \overline{B}^+ - B^+, \omega^+(B^+, u, v) - \widehat{p}(u) + \widehat{p}(v) = 0\}.$$

Put $Q' = (Q - A^+) \cup M'$, where M' is now regarded as a subset of A^+ as in Remark 4.1. Q' is a disjoint union of cycles in $\widetilde{G}_M$ with its length

$$\gamma(Q') = \gamma(Q) + [\gamma(M') - \gamma(Q \cap A^+)] \tag{4.4}$$

being negative, since $-\gamma(M')$ is equal to the maximum weight of a perfect matching in $G(B^+, \overline{B}^+)$ and $Q \cap A^+$ can be identified with a perfect matching in $G(B^+, \overline{B}^+)$. The minimality of Q (with respect to the number of arcs) implies that Q' itself is a negative cycle having the smallest number of arcs.

Suppose, to the contrary, that $(B^+, \overline{B}^+)$ does not satisfy the unique-max condition. Since $(u_i, v_i) \in A^*$ for $i = 1, \ldots, m$, it follows from Lemma 3.5 that there are distinct indices i_k ($k = 1, \ldots, q; q \geq 2$) such that $(u_{i_k}, v_{i_{k+1}}) \in A^*$ for $k = 1, \ldots, q$, where $i_{q+1} = i_1$. That is,

$$\omega^+(B^+, u_{i_k}, v_{i_{k+1}}) = \widehat{p}(u_{i_k}) - \widehat{p}(v_{i_{k+1}}) \qquad (k = 1, \ldots, q). \tag{4.5}$$

On the other hand we have

$$\omega^+(B^+, u_{i_k}, v_{i_k}) = \widehat{p}(u_{i_k}) - \widehat{p}(v_{i_k}) \qquad (k = 1, \ldots, q). \tag{4.6}$$

It then follows that

$$\sum_{k=1}^{q} \omega^+(B^+, u_{i_k}, v_{i_{k+1}}) = \sum_{k=1}^{q} \omega^+(B^+, u_{i_k}, v_{i_k}) \quad \left(= \sum_{k=1}^{q} [\widehat{p}(u_{i_k}) - \widehat{p}(v_{i_k})] \right),$$

i.e.,

$$\sum_{k=1}^{q} \gamma(u_{i_k}, v_{i_{k+1}}) = \sum_{k=1}^{q} \gamma(u_{i_k}, v_{i_k}). \tag{4.7}$$

For $k = 1, \ldots, q$, let $P'(v_{i_{k+1}}, u_{i_k})$ denote the path on Q' from $v_{i_{k+1}}$ to u_{i_k}, and let Q'_k be the directed cycle formed by arc $(u_{i_k}, v_{i_{k+1}})$ and path $P'(v_{i_{k+1}}, u_{i_k})$. Obviously,

$$\gamma(Q'_k) = \gamma(u_{i_k}, v_{i_{k+1}}) + \gamma(P'(v_{i_{k+1}}, u_{i_k})) \qquad (k = 1, \ldots, q). \tag{4.8}$$

A simple but crucial observation here is that

$$\left(\bigcup_{k=1}^{q} P'(v_{i_{k+1}}, u_{i_k}) \right) \cup \{(u_{i_k}, v_{i_k}) \mid k = 1, \ldots, q\} = q' \cdot Q'$$

for some q' with $1 \leq q' < q$, where the union denotes the multiset union, and this expression means that each element of Q' appears q' times on the left-hand side. Hence by adding (4.8) over $k = 1, \ldots, q$ we obtain

$$\begin{aligned} \sum_{k=1}^{q} \gamma(Q'_k) &= \sum_{k=1}^{q} \gamma(u_{i_k}, v_{i_{k+1}}) + \sum_{k=1}^{q} \gamma(P'(v_{i_{k+1}}, u_{i_k})) \\ &= \left[\sum_{k=1}^{q} \gamma(u_{i_k}, v_{i_{k+1}}) - \sum_{k=1}^{q} \gamma(u_{i_k}, v_{i_k}) \right] + q' \cdot \gamma(Q') \\ &= q' \cdot \gamma(Q') \ < 0, \end{aligned}$$

where the last equality is due to (4.7). This implies that $\gamma(Q'_k) < 0$ for some k, while Q'_k has a smaller number of arcs than Q'. This contradicts the minimality of Q'. Therefore, $(B^+, \overline{B}^+)$ satisfies the unique-max condition, similarly for $(B^-, \overline{B}^-)$. □

LEMMA 4.5. *For a negative cycle Q in $\widetilde{G}_M$ having the smallest number of arcs,*

$$\overline{M} = (M - \{a \in M \mid \overline{a} \in Q \cap M^\circ\}) \cup (Q \cap A^\circ)$$

is an independent assignment with $\Omega(\overline{M}) \geq \Omega(M) - \gamma_M(Q)$ $(> \Omega(M))$.

Proof. Note that $\overline{B}^+ = \partial^+ \overline{M}$, $\overline{B}^- = \partial^- \overline{M}$ for $\overline{B}^+$, $\overline{B}^-$ as defined in (4.2), (4.3), and recall the notation $B^+ = \partial^+ M$, $B^- = \partial^- M$. By Lemma 4.4 and Lemma 3.8 (unique-max lemma), we have

$$\begin{aligned} \omega^+(\overline{B}^+) &= \omega^+(B^+) + \widehat{\omega}^+(B^+, \overline{B}^+) \geq \omega^+(B^+) - \gamma(Q \cap A^+), \\ \omega^-(\overline{B}^-) &= \omega^-(B^-) + \widehat{\omega}^-(B^-, \overline{B}^-) \geq \omega^-(B^-) - \gamma(Q \cap A^-). \end{aligned}$$

Also we have

$$w(\overline{M}) = w(M) - \gamma(Q \cap (A^\circ \cup M^\circ)).$$

Addition of these inequalities yields $\Omega(\overline{M}) \geq \Omega(M) - \gamma(Q)$. □

The above lemma shows "(OPT) ⇒ (NNC)".

(NNC) ⇒ (POT): by the well-known fact in graph theory, (NNC) implies the existence of a function $p : V^+ \cup V^- \to R$ such that

$$\gamma(a) + p(\partial^+ a) - p(\partial^- a) \geq 0 \qquad (a \in \widetilde{A}).$$

This condition for $a \in A^\circ \cup M^\circ$ is equivalent to condition (i) in Theorem 4.1. For $a = (u, v) \in A^+$, where $u \in C^+(B^+, v)$, it means

$$\omega^+(B^+, u, v) - p(u) + p(v) \leq 0.$$

Namely,

$$\omega^+[p^+](B^+, u, v) \leq 0 \qquad (u \in C^+(B^+, v)),$$

which in turn implies condition (ii) in Theorem 4.1 by Lemma 3.1. Similarly, the above condition for $a \in A^-$ implies condition (iii) in Theorem 4.1. Thus "(NNC) ⇒ (POT)" has been shown.

(POT) ⇒ (OPT): for any independent assignment M and any function $p : V^+ \cup V^- \to R$, we see

$$\begin{aligned}
\Omega(M) &= \omega^+(\partial^+ M) + \omega^-(\partial^- M) + w(M) \\
&= \left[\omega^+(\partial^+ M) + \sum_{a \in M} p(\partial^+ a)\right] + \left[\omega^-(\partial^- M) - \sum_{a \in M} p(\partial^- a)\right] \\
&\quad + \sum_{a \in M} [w(a) - p(\partial^+ a) + p(\partial^- a)] \\
&= \omega^+[p^+](\partial^+ M) + \omega^-[-p^-](\partial^- M) + \sum_{a \in M} w_p(a),
\end{aligned}$$

where $w_p(a) = w(a) - p(\partial^+ a) + p(\partial^- a)$.

Suppose M and p satisfy (i)–(iii) of Theorem 4.1 and take an arbitrary independent assignment M'. Then we have

$$\begin{aligned}
\Omega(M') &= \omega^+[p^+](\partial^+ M') + \omega^-[-p^-](\partial^- M') + \sum_{a \in M'} w_p(a) \\
&\leq \omega^+[p^+](\partial^+ M) + \omega^-[-p^-](\partial^- M) \\
&= \Omega(M).
\end{aligned}$$

This shows that M is optimal, establishing "(POT) ⇒ (OPT)".

Finally for the second half of Theorem 4.1 we note in the above inequality that $\Omega(M') = \Omega(M)$ if and only if $\omega^+[p^+](\partial^+ M') = \omega^+[p^+](\partial^+ M)$, $\omega^-[-p^-](\partial^- M') = \omega^-[-p^-](\partial^- M)$, $w_p(a) = 0$ for $a \in M'$.

We have completed the proofs of Theorems 4.1 and 4.3.

5. Extension to VIAP(k).

5.1. Theorems. In this section the optimality criteria for VIAP(k) introduced at the end of §2 are derived from Theorems 4.1 and 4.3. The proofs are given in §5.2.

THEOREM 5.1. (1) *A feasible solution (M, B^+, B^-) for VIAP(k) is optimal if and only if there exists a "potential" function $p : V^+ \cup V^- \to R$ such that*

(i) $w(a) - p(\partial^+ a) + p(\partial^- a) \begin{cases} \leq 0 & (a \in A), \\ = 0 & (a \in M), \end{cases}$

(ii) B^+ *is a maximum-weight base of* $\mathbf{M}^+$ *with respect to* $\omega^+[p^+]$,

(iii) B^- *is a maximum-weight base of* $\mathbf{M}^-$ *with respect to* $\omega^-[-p^-]$,

(iv) $p(u) \geq p(v) \qquad (u \in V^+,\ v \in B^+ - \partial^+ M)$,

(v) $p(u) \leq p(v) \qquad (u \in V^-,\ v \in B^- - \partial^- M)$.

(2) *Let* p *be a potential that satisfies* (i)–(v) *above for some (optimal)* (M_0, B_0^+, B_0^-). *Then* (M, B^+, B^-) *is optimal if and only if it satisfies* (i)–(v).

To express the optimality in terms of negative cycles we need to introduce an auxiliary graph $\widetilde{G}_{(M,B^+,B^-)} = (\widetilde{V}, \widetilde{A})$ associated with (M, B^+, B^-), which is a slight modification of the one used in §4. The vertex set $\widetilde{V}$ of $\widetilde{G}_{(M,B^+,B^-)}$ is given by

$$\widetilde{V} = V^+ \cup V^- \cup \{s^+, s^-\},$$

where s^+ and s^- are new vertices referred to as the source vertex and the sink vertex, respectively. The arc set $\widetilde{A}$ consists of eight disjoint parts:

$$\widetilde{A} = (A^\circ \cup M^\circ) \cup (A^+ \cup F^+ \cup S^+) \cup (A^- \cup F^- \cup S^-),$$

where

$$\begin{aligned}
A^\circ &= \{a \mid a \in A\} \qquad \text{(copy of } A\text{)}, \\
M^\circ &= \{\overline{a} \mid a \in M\} \qquad (\overline{a}\text{: reorientation of } a), \\
A^+ &= \{(u, v) \mid u \in B^+, v \in V^+ - B^+, u \in C^+(B^+, v)\}, \\
F^+ &= \{(u, s^+) \mid u \in V^+\}, \\
S^+ &= \{(s^+, v) \mid v \in B^+ - \partial^+ M\}, \\
A^- &= \{(v, u) \mid u \in B^-, v \in V^- - B^-, u \in C^-(B^-, v)\}, \\
F^- &= \{(s^-, u) \mid u \in V^-\}, \\
S^- &= \{(v, s^-) \mid v \in B^- - \partial^- M\}.
\end{aligned}$$

The arc length $\gamma(a) = \gamma_{(M,B^+,B^-)}(a)$ $(a \in \widetilde{A})$ is defined by

$$\gamma(a) = \begin{cases} -w(a) & (a \in A^\circ), \\ w(\overline{a}) & (a = (u, v) \in M^\circ, \overline{a} = (v, u) \in M), \\ -\omega^+(B^+, u, v) & (a = (u, v) \in A^+), \\ -\omega^-(B^-, u, v) & (a = (v, u) \in A^-), \\ 0 & (a \in F^+ \cup S^+ \cup F^- \cup S^-). \end{cases} \tag{5.1}$$

THEOREM 5.2. *A feasible solution* (M, B^+, B^-) *for VIAP(k) is optimal if and only if there exists in* $\widetilde{G}_{(M,B^+,B^-)}$ *no negative cycle with respect to the arc length* $\gamma_{(M,B^+,B^-)}$.

Remark 5.1. The definition of $F^\pm$ could be replaced by

$$F^+ = \{(u, s^+) \mid u \in \partial^+ M \cup (V^+ - B^+)\}, \quad F^- = \{(s^-, u) \mid u \in \partial^- M \cup (V^- - B^-)\}$$

without affecting the above theorem. The present definition is more convenient for the algorithm to be developed in part II [25].

Remark 5.2. When $k = r^+ = r^-$, the auxiliary graph $\widetilde{G}_{(M,B^+,B^-)}$ contains the auxiliary graph $\widetilde{G}_M$ of §4 as a subgraph.

5.2. Proofs. We formulate VIAP(k) as a valuated independent assignment problem on $G_k = (V_k^+, V_k^-; A_k)$ with valuated matroids $\mathbf{M}_k^+ = (V_k^+, \mathcal{B}_k^+, \omega_k^+)$ and $\mathbf{M}_k^- = (V_k^-, \mathcal{B}_k^-, \omega_k^-)$ having a common rank $r^+ + r^- - k$. The graph $G_k = (V_k^+, V_k^-; A_k)$ is defined as follows:

$$
\begin{aligned}
&V_k^+ = V^+ \cup U_k^+, \qquad U_k^+ \equiv \{u_i^+ \mid 1 \le i \le r^- - k\},\\
&V_k^- = V^- \cup U_k^-, \qquad U_k^- \equiv \{u_i^- \mid 1 \le i \le r^+ - k\},\\
&A_k = A \cup \{(u, u_i^-) \mid u \in V^+, u_i^- \in U_k^-\} \cup \{(u_i^+, u) \mid u \in V^-, u_i^+ \in U_k^+\}.
\end{aligned}
$$

The valuated matroid $\mathbf{M}_k^+$ is the direct sum of $\mathbf{M}^+$ and the free matroid on U_k^+ with trivial valuation (which is zero); i.e., $\mathcal{B}_k^+ = \{B \cup U_k^+ \mid B \in \mathcal{B}^+\}$ and $\omega_k^+(B \cup U_k^+) = \omega^+(B)$ for $B \in \mathcal{B}^+$, similarly for $\mathbf{M}_k^-$. The weight $w_k : A_k \to R$ is defined by

$$
w_k(a) = \begin{cases} w(a) & (a \in A),\\ 0 & (a \in A_k - A). \end{cases}
$$

With an independent assignment M_k in G_k we can create a feasible solution (M, B^+, B^-) for VIAP(k) by defining $M = M_k \cap A$, $B^+ = \partial^+ M_k - U_k^+$, and $B^- = \partial^- M_k - U_k^-$. Conversely, from (M, B^+, B^-) feasible for VIAP(k) we can construct an independent assignment M_k in G_k. Moreover, we have $\Omega(M, B^+, B^-) = \Omega_k(M_k)$, where $\Omega_k(M_k) \equiv w_k(M_k) + \omega_k^+(\partial^+ M_k) + \omega_k^-(\partial^- M_k)$.

Through this reduction of VIAP(k) to the valuated independent assignment problem, Theorems 4.1 and 4.3 translate into Theorems 5.1 and 5.2, respectively.

Acknowledgments. The author is grateful to Andreas Dress for a stimulating comment on [23] on the occasion of the 15th International Symposium on Mathematical Programming at Ann Arbor, August 1994, which motivated the present work. He also thanks Satoru Iwata for careful reading of the manuscript and for fruitful discussions, which resulted in the extension given in §5. The discussion with András Sebő was also fruitful, which led to Remark 3.3.

REFERENCES

[1] R. A. Brualdi, *Comments on bases in dependence structures*, Bull. Austral. Math. Soc., 1 (1969), pp. 161–167.

[2] A. W. M. Dress and W. Terhalle, *Well-layered maps and the maximum-degree $k \times k$-subdeterminant of a matrix of rational functions*, Appl. Math. Lett., 8 (1995), pp. 19–23.

[3] ———, *Well-layered maps—A class of greedily optimizable set functions*, Appl. Math. Lett., 8 (1995), pp. 77–80.

[4] ———, *Rewarding maps—On greedy optimization of set functions*, Adv. in Appl. Math., 16 (1995), pp. 464–483.

[5] A. W. M. Dress and W. Wenzel, *Valuated matroid: A new look at the greedy algorithm*, Appl. Math. Lett., 3 (1990), pp. 33–35.

[6] ———, *Valuated matroids*, Adv. Math., 93 (1992), pp. 214–250.

[7] J. Edmonds, *Submodular functions, matroids and certain polyhedra*, in Combinatorial Structures and Their Applications, R. Guy, H. Hanai, N. Sauer, and J. Schönsheim, eds., Gordon and Breach, New York, 1970, pp. 69–87.

[8] ———, *Matroid intersection*, Ann. Discrete Math., 14 (1979), pp. 39–49.

[9] U. Faigle, *Matroids in combinatorial optimization*, in Combinatorial Geometries, N. White, ed., Cambridge University Press, London, 1987, pp. 161–210.

[10] A. Frank, *A weighted matroid intersection algorithm*, J. Algorithms, 2 (1981), pp. 328–336.

[11] ———, *An algorithm for submodular functions on graphs*, Ann. Discrete Math., 16 (1982), pp. 97–120.

[12] S. Fujishige, *A primal approach to the independent assignment problem*, J. Oper. Res. Soc. Japan, 20 (1977), pp. 1–15.

[13] S. FUJISHIGE, *An algorithm for finding an optimal independent linkage*, J. Oper. Res. Soc. Japan, 20 (1977), pp. 59–75.
[14] ———, *Submodular Functions and Optimization*, Ann. Discrete Math. 47, North-Holland, Amsterdam, 1991.
[15] M. IRI, *A practical algorithm for the Menger-type generalization of the independent assignment problem*, Math. Prog. Study, 8 (1978), pp. 88–105.
[16] ———, *Applications of matroid theory*, in Mathematical Programming–The State of the Art, A. Bachem, M. Grötschel, and B. Korte, eds., Springer-Verlag, Berlin, 1983, pp. 158–201.
[17] M. IRI AND N. TOMIZAWA, *An algorithm for finding an optimal "independent assignment"*, J. Oper. Res. Soc. Japan, 19 (1976), pp. 32–57.
[18] S. KROGDAHL, *The dependence graph for bases in matroids*, Discrete Math., 19 (1977), pp. 47–59.
[19] J. P. S. KUNG, *Basis-exchange properties*, in Theory of Matroids, N. White, ed., Cambridge University Press, London, 1986, pp. 62–75.
[20] E. L. LAWLER, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.
[21] L. LOVÁSZ AND M. PLUMMER, *Matching Theory*, North-Holland, Amsterdam, 1986.
[22] K. MUROTA, *Systems Analysis By Graphs and Matroids—Structural Solvability and Controllability*, Algorithms and Combinatorics 3, Springer-Verlag, Berlin, 1987.
[23] ———, *Combinatorial relaxation algorithm for the maximum degree of subdeterminants: Computing Smith-McMillan form at infinity and structural indices in Kronecker form*, Applicable Algebra in Engineering, Communication and Computing, 6 (1995), pp. 251–273.
[24] ———, *Finding optimal minors of valuated bimatroids*, Appl. Math. Lett., 8 (1995), pp. 37–42.
[25] ———, *Valuated matroid intersection* II: *Algorithms*, SIAM J. Discrete Math., 9 (1996), pp. 562–576.
[26] ———, *Fenchel-type Duality for Matroid Valuations*, Report 95839-OR, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, Germany, 1995.
[27] ———, *Submodular Flow Problem with a Nonseparable Cost Function*, Report 95843-OR, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, Germany, 1995.
[28] ———, *Convexity and Steinitz's exchange property*, Adv. Appl. Math., to appear. Extended abstract in the Proceedings of Integer Programming and Combinatorial Optimization, V, June 1996, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1996.
[29] ———, *Structural approach in systems analysis by mixed matrices—An exposition for index of DAE*, in Proc. ICIAM 95, Hamburg, K. Kirchgässner, O. Mahrenholtz, and R. Mennicken, eds., Mathematical Research, Academie Verlag, Berlin, 1996.
[30] A. RECSKI, *Matroid Theory and Its Applications in Electric Network Theory and In Statics*, Algorithms and Combinatorics 6, Springer-Verlag, Berlin, 1989.
[31] H. H. ROSENBROCK, *State-space and Multivariable Theory*, Nelson, London, 1970.
[32] A. SCHRIJVER, *Matroids and linking systems*, J. Combin. Theory Ser. B, 26 (1979), pp. 349–369.
[33] W. TERHALLE, *Ein kombinatorischer Zugang zu p-adischer Geometrie: Bewertete Matroide, Bäume und Gebäude*, Doctor's dissertation, Universität Bielefeld, Germany, 1992.
[34] M. VIDYASAGAR, *Control System Synthesis: A Factorization Approach*, MIT Press, Cambridge, MA, 1985.
[35] U. ZIMMERMANN, *Minimization on submodular flows*, Discrete Appl. Math., 4 (1982), pp. 303–323.
[36] ———, *Negative circuits for flows and submodular flows*, Discrete Appl. Math., 36 (1992), pp. 179–189.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 562–576, November 1996

004

VALUATED MATROID INTERSECTION II: ALGORITHMS*

KAZUO MUROTA†

Abstract. Based on the optimality criteria established in part I [*SIAM J. Discrete Math.*, 9 (1996), pp. 545–561] we show a primal-type cycle-canceling algorithm and a primal–dual-type augmenting algorithm for the valuated independent assignment problem: given a bipartite graph $G = (V^+, V^-; A)$ with arc weight $w : A \to \mathbf{R}$ and matroid valuations ω^+ and ω^- on V^+ and V^-, respectively, find a matching $M(\subseteq A)$ that maximizes $\sum\{w(a) \mid a \in M\} + \omega^+(\partial^+ M) + \omega^-(\partial^- M)$, where $\partial^+ M$ and $\partial^- M$ denote the sets of vertices in V^+ and V^- incident to M. The proposed algorithms generalize the previous algorithms for the independent assignment problem as well as for the weighted matroid intersection problem, including those due to Lawler [*Math. Prog.*, 9 (1975), pp. 31–56], Iri and Tomizawa [*J. Oper. Res. Soc. Japan*, 19 (1976), pp. 32–57], Fujishige [*J. Oper. Res. Soc. Japan*, 20 (1977), pp. 1–15], Frank [*J. Algorithms*, 2 (1981), pp. 328–336], and Zimmermann [*Discrete Appl. Math.*, 36 (1992), pp. 179–189].

Key words. weighted matroid intersection problem, independent assignment problem, valuated matroid, combinatorial optimization

AMS subject classifications. 90C35, 90C27, 90B80

1. Introduction.

Part I of this paper [24] has introduced an extension of the independent assignment problem using the concept of the valuated matroid of Dress and Wenzel [4], [5]. A matroid $\mathbf{M} = (V, \mathcal{B})$ defined on a ground set V by the family of bases $\mathcal{B}$ is called a valuated matroid if it is equipped with a function $\omega : \mathcal{B} \to R$ that enjoys the exchange axiom: for $B, B' \in \mathcal{B}$ and $u \in B - B'$ there exists $v \in B' - B$ such that $B - u + v \in \mathcal{B}$, $B' + u - v \in \mathcal{B}$, and

$$\omega(B) + \omega(B') \leq \omega(B - u + v) + \omega(B' + u - v). \tag{1.1}$$

Here R is a totally ordered additive group. (Typically $R = \mathbf{R}$ (reals), $\mathbf{Q}$ (rationals), or $\mathbf{Z}$ (integers).) The problem considered in part I is as follows:

> VALUATED INDEPENDENT ASSIGNMENT PROBLEM. Given a bipartite graph $G = (V^+, V^-; A)$, valuated matroids $\mathbf{M}^+ = (V^+, \mathcal{B}^+, \omega^+)$ and $\mathbf{M}^- = (V^-, \mathcal{B}^-, \omega^-)$, and arc weight $w : A \to R$, find a matching $M(\subseteq A)$ that maximizes
>
> $$\Omega(M) \equiv w(M) + \omega^+(\partial^+ M) + \omega^-(\partial^- M)$$
>
> subject to the constraint
>
> $$\partial^+ M \in \mathcal{B}^+, \quad \partial^- M \in \mathcal{B}^-,$$
>
> where $\partial^+ M$ and $\partial^- M$ denote the sets of vertices in V^+ and V^- incident to M.

We assume $r^+ = r^-$ for the feasibility of the problem, where r^+ and r^- denote the ranks of $\mathbf{M}^+$ and $\mathbf{M}^-$. A special case of this problem is

* Received by the editors January 18, 1995; accepted for publication (in revised form) October 30, 1995.

† Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-01, Japan (murota@kurims.kyoto-u.ac.jp). This research was completed while the author was at Forschungsinstitut für Diskrete Mathematik, Universität Bonn.

INTERSECTION PROBLEM. Given a pair of valuated matroids $\mathbf{M}^1 = (V, \mathcal{B}^1, \omega^1)$ and $\mathbf{M}^2 = (V, \mathcal{B}^2, \omega^2)$ defined on a common ground set V and a weight function $w : V \to R$, find a common base $B \in \mathcal{B}^1 \cap \mathcal{B}^2$ that maximizes $w(B) + \omega^1(B) + \omega^2(B)$.

Also considered in part I [24] is the following more general form of the valuated independent assignment problem (VIAP).

VIAP(k). Maximize

$$\Omega(M, B^+, B^-) \equiv w(M) + \omega^+(B^+) + \omega^-(B^-)$$

subject to the constraint that M is a matching of size k, and

$$\partial^+ M \subseteq B^+ \in \mathcal{B}^+, \quad \partial^- M \subseteq B^- \in \mathcal{B}^-.$$

Obviously, VIAP(k) with $k = r^+ = r^-$ is the VIAP above, although VIAP(k) does not presuppose that $r^+ = r^-$. VIAP(k) with trivial valuations ($\omega^\pm \equiv 0$ on $\mathcal{B}^\pm$) is nothing but the problem of finding a maximum-weight independent matching of size k. For the intersection problem (with valuations) VIAP(k) takes the following form:

maximize $w(I) + \omega^1(B^1) + \omega^2(B^2)$ under the constraint that I is a common independent set of size k, and B^1 (respectively, B^2) is a base of $\mathbf{M}^1$ (respectively, $\mathbf{M}^2$) containing I.

Part I has established the optimality criteria to the above problems.

Part II gives a primal-type cycle-canceling algorithm and a primal–dual-type augmenting algorithm for the valuated independent assignment problem, both of which run in strongly polynomial time with oracles for the valuations $\omega^\pm$. Our cycle-canceling algorithm is based on the negative-cycle criterion (Theorem 4.3 of part I) and is an extension of Fujishige's [9] for the ordinary independent assignment problem. It can be polished up to a strongly polynomial algorithm using the minimum-ratio-cycle strategy invented by Zimmermann [28]. Our augmenting algorithm, on the other hand, solves VIAP(k) for $k = 0, 1, 2, \ldots$ and is an extension of the well-established primal-dual algorithm for the ordinary independent assignment problem and the weighted matroid intersection problem due to Iri and Tomizawa [15] and Lawler [18], [19]. (See also Frank [7]; for the submodular flow problem see Cunningham and Frank [3], Frank [8], and Fujishige [11], [12].)

Remark 1.1. Extensions of the proposed algorithms to the linkage-type problems as considered in Fujishige [10] and Iri [14] should be obvious from the results of this paper combined with those in [10] and [14]. In this paper no further explicit accounts will be made about this.

Let us recall some notation and lemmas from part I which are constantly referred to in this part. Let $\mathbf{M} = (V, \mathcal{B}, \omega)$ be a valuated matroid. For $B \in \mathcal{B}$, $v \in V - B$, and $u \in V$,

$$C(B, v): \quad \text{fundamental circuit of } v \text{ relative to } B \quad (=\text{unique circuit contained in } B + v), \tag{1.2}$$

$$\omega(B, u, v) = \omega(B - u + v) - \omega(B), \tag{1.3}$$

where $\omega(B, u, v) \neq -\infty$ if and only if $u \in C(B, v)$, assuming the convention that $\omega(B') = -\infty$ if $B' \notin \mathcal{B}$. For $B \in \mathcal{B}$ and $B' \subseteq V$,

$$G(B, B') : \text{exchangeability graph with vertex bipartition}(B - B', B' - B) \text{ and arc set}(u, v) \mid u \in B - B', v \in B' - B, u \in C(B, v);$$

each arc (u, v) is given weight $\omega(B, u, v)$,

$\widehat{\omega}(B, B')$: weight of a maximum-weight perfect matching in $G(B, B')$,

Unique-max condition : there exists exactly one maximum-weight perfect matching in $G(B, B')$.

The following two lemmas are the driving wheels for the technical developments (cf. Lemmas 3.4 and 3.8 of part I).

LEMMA 1.1 ("upper-bound lemma"). *For $B, B' \in \mathcal{B}$,*

$$\omega(B') \leq \omega(B) + \widehat{\omega}(B, B'). \tag{1.4}$$

LEMMA 1.2 ("unique-max lemma"). *Let $B \in \mathcal{B}$ and $B' \subseteq V$ with $|B'| = |B|$. If (B, B') satisfies the unique-max condition, then $B' \in \mathcal{B}$ and*

$$\omega(B') = \omega(B) + \widehat{\omega}(B, B'). \tag{1.5}$$

2. Cycle-canceling algorithms.

2.1. Algorithms. In Theorem 4.3 of part I we have shown a negative cycle criterion for the optimality with reference to an auxiliary graph $\widetilde{G}_M = (\widetilde{V}, \widetilde{A})$ with $\widetilde{V} = V^+ \cup V^-$ and $\widetilde{A} = A^\circ \cup M^\circ \cup A^+ \cup A^-$, where

$$\begin{aligned}
A^\circ &= \{a \mid a \in A\} \qquad \text{(copy of } A\text{)},\\
M^\circ &= \{\overline{a} \mid a \in M\} \qquad (\overline{a}\text{: reorientation of } a),\\
A^+ &= \{(u, v) \mid u \in B^+, v \in V^+ - B^+, u \in C^+(B^+, v)\},\\
A^- &= \{(v, u) \mid u \in B^-, v \in V^- - B^-, u \in C^-(B^-, v)\}.
\end{aligned}$$

Here $B^+ = \partial^+ M$, $B^- = \partial^- M$, and $C^+(B^+, v)$ and $C^-(B^-, v)$ are defined as in (1.2). The arc length $\gamma_M(a)$ $(a \in \widetilde{A})$ is defined by

$$\gamma_M(a) = \begin{cases} -w(a) & (a \in A^\circ),\\ w(\overline{a}) & (a = (u, v) \in M^\circ, \overline{a} = (v, u) \in M),\\ -\omega^+(B^+, u, v) & (a = (u, v) \in A^+),\\ -\omega^-(B^-, u, v) & (a = (v, u) \in A^-), \end{cases} \tag{2.1}$$

where $\omega^+(B^+, u, v)$ and $\omega^-(B^-, u, v)$ are as in (1.3).

Theorem 4.3 of part I as well as its proof suggests an algorithm for solving the valuated independent assignment problem as follows. The validity of this procedure follows from Theorem 4.3 and Lemma 4.5 of part I.

CYCLE-CANCELING ALGORITHM. Starting from an arbitrary independent assignment M, repeat (i)–(ii) below while there exists a negative cycle in $\widetilde{G}_M$:

(i) Find a negative cycle Q having the smallest number of arcs in the auxiliary graph $\widetilde{G}_M$ (with respect to the arc length γ_M).

(ii) Modify the current independent matching along the cycle Q by

$$\overline{M} = (M - \{a \in M \mid \overline{a} \in Q \cap M^\circ\}) \cup (Q \cap A^\circ).$$

This is a straightforward extension of the primal algorithm of Fujishige [9] for the ordinary independent assignment problem, which extends the classical idea of Klein

[17] and which is further extended by Fujishige [11] and by Zimmermann [27] for the submodular flow problem. (See also Fujishige [12] and the references therein.)

The above algorithm assumes an initial independent assignment M, which can be found by the existing algorithms for the (unweighted) matroid intersection. For each M, the graph $\widetilde{G}_M$ can be constructed with $r^+(|V^+|-r^+)$ evaluations of ω^+ and $r^-(|V^-|-r^-)$ evaluations of ω^-, where r^+ and r^- are the ranks of $\mathbf{M}^+$ and $\mathbf{M}^-$, respectively. (We have $r^+ = r^-$ for a feasible problem.) When the valuated matroids are associated with polynomial matrices as in Example 3.2 of part I, $\omega^\pm$ can be evaluated by the method of interpolation or by an algorithm of combinatorial relaxation type (see Murota [23] for details); or, more directly, $\omega^\pm(\cdot,\cdot,\cdot)$ can be determined by pivoting operations on the matrices if arithmetic operations on rational functions can be performed.

A negative cycle having the smallest number of arcs in (i) can be found easily by a variant of the standard shortest-path algorithm. It should however be worth noting that the minimality of the number of arcs is not really necessary, and in fact this observation adds more flexibility to the algorithm, as we will see soon. Recalling the notation

$$\overline{B}^+ = B^+ - \{\partial^+ a \mid a \in Q \cap A^+\} + \{\partial^- a \mid a \in Q \cap A^+\}, \tag{2.2}$$

$$\overline{B}^- = B^- - \{\partial^- a \mid a \in Q \cap A^-\} + \{\partial^+ a \mid a \in Q \cap A^-\}, \tag{2.3}$$

we call a cycle Q in $\widetilde{G}_M$ admissible if both $(B^+, \overline{B}^+)$ and $(B^-, \overline{B}^-)$ satisfy the unique-max condition in $\mathbf{M}^+$ and $\mathbf{M}^-$, respectively. The admissibility of Q guarantees (by the unique-max lemma) that the modified matching $\overline{M}$ remains to be an independent assignment.

In the proof of Lemma 4.4 of part I it has been shown that if a negative cycle Q is not admissible, a family of cycles, denoted Q'_k $(k = 1, \ldots, q)$, is naturally defined and that at least one of its members is a negative cycle. Following the terminology of Zimmermann [28] (for the submodular flow problem) let us call each Q'_k an induced cycle. The above observations lead to the following refinements of Lemmas 4.4 and 4.5 of part I.

LEMMA 2.1. *Let Q be a negative cycle in $\widetilde{G}_M$. Either Q is admissible or else it induces a negative cycle having a smaller number of arcs than Q. In particular, a negative cycle having the smallest number of arcs is admissible.*

LEMMA 2.2. *For an admissible cycle Q in $\widetilde{G}_M$, $\overline{M}$ is an independent assignment with $\Omega(\overline{M}) \geq \Omega(M) - \gamma_M(Q)$.*

Remark 2.1. In the case of the ordinary independent assignment problem, the admissibility of a cycle defined above agrees with the admissibility in the sense of [28] and with the feasibility in the sense of [2]. This is due to the observation made in Remark 3.2 of part I.

The algorithm finds the optimal independent assignment in a finite number of steps since there exist a finite number of independent assignments in the given graph and the objective function value $\Omega(M)$ increases monotonically; we have seen

$$\Omega(\overline{M}) \geq \Omega(M) - \gamma_M(Q) \qquad (> \Omega(M)). \tag{2.4}$$

(It may be emphasized, however, that the gain in $\Omega(M)$ can be larger than $-\gamma_M(Q)$, a phenomenon which cannot occur in the ordinary independent assignment problem. See the proof of Lemma 4.5 of part I.) However, the number of iterations of the loop (i)–(ii) is not bounded by a polynomial in the problem size as is also the case with

the original form of the primal algorithm for the ordinary independent assignment problem.

Recently Zimmermann [28] has shown (for the submodular flow problem) that, when $R \subseteq \mathbf{R}$, a judicious choice of a negative cycle renders the number of iterations bounded by r^+ $(= r^-)$. The idea is to introduce an auxiliary weight function α on $\widetilde{A}$ and to select a cycle Q of minimum ratio $\gamma_M(Q)/\alpha(Q)$ (satisfying some extra condition). In what follows we shall show that this idea carries over to our problem, making the number of iterations of the loop (i)–(ii) of our algorithm bounded by r^+ $(= r^-)$.

We maintain a subset $M^\bullet$ of $\widetilde{A}$, called the active arc set, and define $\alpha : \widetilde{A} \to \{0, 1\}$ by

$$\alpha(a) = \begin{cases} 1 & (a \in M^\bullet), \\ 0 & (a \in \widetilde{A} - M^\bullet). \end{cases}$$

An arc is said to be active if it belongs to $M^\bullet$. A cycle Q $(\subseteq \widetilde{A})$ is called a minimum-ratio cycle with respect to (γ_M, α) if $\gamma_M(Q)/\alpha(Q)$ takes the minimum value among all cycles with $\alpha(Q) > 0$. We assume $R \subseteq \mathbf{R}$ till the end of §2.

CYCLE-CANCELING ALGORITHM WITH MINIMUM-RATIO CYCLE. Starting from an arbitrary independent assignment M and active arc set defined by $M^\bullet = M^\circ$ $(\equiv \{\overline{a} \mid a \in M\})$, repeat (i)–(iii) below while there exists a negative cycle in $\widetilde{G}_M$:

(i) Find an admissible minimum-ratio cycle Q in the auxiliary graph $\widetilde{G}_M$ (with respect to (γ_M, α)).

(ii) Modify the current active arc set by

$$\overline{M^\bullet} = M^\bullet - (Q \cap M^\circ)$$

and the function α accordingly.

(iii) Modify the current independent matching along the cycle Q by

$$\overline{M} = (M - \{a \in M \mid \overline{a} \in Q \cap M^\circ\}) \cup (Q \cap A^\circ).$$

The following properties are maintained throughout the computation:

- Any negative cycle in $\widetilde{G}_M$ contains an active arc (cf. Lemma 2.8).
- M is an independent assignment. (That is, $\partial^+ M \in \mathcal{B}^+$, $\partial^- M \in \mathcal{B}^-$.)

Because of the first property, the minimum-ratio cycle in (i) is well defined, as long as $\widetilde{G}_M$ contains a negative cycle. In (ii), on the other hand, the active arc set $M^\bullet$ decreases monotonically, at least by one element in each iteration. This implies the termination of the algorithm in at most r^+ $(= r^-)$ iterations, whereas the obtained matching M is an optimal independent assignment by the second property and Theorem 4.3 of part I.

An admissible minimum-ratio cycle can be found in a polynomial time in the problem size as follows. By an algorithm of Megiddo [22], a minimum-ratio cycle Q can be generated in $O(|\widetilde{V}|^2|\widetilde{A}| \log |\widetilde{V}|)$ time. We can test for the admissibility of Q on the basis of Lemma 3.5 of part I by means of an algorithm for the weighted bipartite matching problem. This takes $O(|\widetilde{V}|^3)$ or less time. In case Q is not admissible, it induces a (nonempty) family of minimum-ratio cycles each having a smaller number

of arcs than Q, as will be shown later in Lemma 2.5. We pick up any one of the induced minimum-ratio cycles and repeat the above procedure. After repeating not more than $|\widetilde{V}|$ times we are guaranteed to obtain an admissible minimum-ratio cycle.

Summarizing the above arguments we have the following theorem, where $R \subseteq \mathbf{R}$ is assumed.

THEOREM 2.3. *The cycle-canceling algorithm with minimum-ratio cycle selection is a strongly polynomial time algorithm (modulo a polynomial number of evaluations of $\omega^{\pm}$).*

In connection to Lemma 2.2 it may be noted (cf. Lemma 2.6) that the minimum-ratio cycle selection yields an equality in (2.4) (in contrast to the original version of the algorithm).

Finally we mention two other variants of the cycle-canceling algorithm using the minimum-mean-cycle strategy, which was introduced by Goldberg and Tarjan [13] for the minimum-cost flow problem and adapted to the submodular flow problem by Cui and Fujishige [2]. The mean length of a cycle Q in $\widetilde{G}_M$ is $\gamma_M(Q)$ divided by $|Q|$ (= the number of arcs in Q), and a minimum-mean cycle means a cycle having the minimum mean length. Note that a minimum-mean cycle is a minimum-ratio cycle for $\alpha \equiv 1$.

The variants suggested here are to select, as in [2], a minimum-mean cycle Q in $\widetilde{G}_M$ according to one of the rules:

(N) select a minimum-mean cycle having the smallest number of arcs,

(L) select a minimum-mean cycle such that $q_M(\partial^+ a) = \pi(\partial^- a)$ for each arc in it, where $\pi : \widetilde{V} \to \{1, 2, \ldots, |\widetilde{V}|\}$ is a fixed one-to-one mapping (= ordering of $\widetilde{V}$) and

$$q_M(u) = \min\{\pi(v) \mid \text{arc } (u, v) \text{ lies on a minimum-mean cycle in } \widetilde{G}_M\}.$$

It is observed in [2] that such minimum-mean cycles can be found in $O(|\widetilde{V}|\,|\widetilde{A}|)$ time using an algorithm of Karp [16]. (See also McCormick [21] and Orlin and Ahuja [25] for algorithms for minimum-mean cycles.) The validity of these variants can be shown similarly as that of the minimum-ratio cycle algorithm. (See Remark 2.2.) Again we have an equality in (2.4).

2.2. Validity of the minimum-ratio cycle algorithm. We shall show the validity of the cycle-canceling algorithm using the minimum-ratio cycle selection. Basically we follow the arguments in [13], [28] while establishing two lemmas (Lemmas 2.5 and 2.7) specific to our problem. We abbreviate γ_M to γ for notational simplicity.

For $\varepsilon \geq 0$ an independent assignment M is said to be ε-optimal (with respect to α) if there exists a function $p : \widetilde{V} \to \mathbf{R}$ such that

$$\gamma_p(a) \equiv \gamma(a) + p(\partial^+ a) - p(\partial^- a) \geq -\varepsilon\alpha(a) \qquad (a \in \widetilde{A}). \tag{2.5}$$

Noting (2.5) is equivalent to saying that the modified arc length $\widehat{\gamma}(a) = \gamma(a) + \varepsilon\alpha(a)$ admits a function p such that

$$\widehat{\gamma}(a) + p(\partial^+ a) - p(\partial^- a) \geq 0 \qquad (a \in \widetilde{A}),$$

we see that the existence of p with (2.5) is also equivalent to

$$\gamma(Q) \geq -\varepsilon\alpha(Q) \qquad (Q : \text{ negative cycle}). \tag{2.6}$$

This implies obviously that $\alpha(Q) > 0$ for any negative cycle Q; that is,

$$\text{any negative cycle in } \widetilde{G}_M \text{ contains an active arc.} \tag{2.7}$$

Conversely suppose (2.7) is true and

(2.8) there exists a negative cycle.

Then the "minimum cycle ratio"

$$\mu = \min\left\{\frac{\gamma(Q)}{\alpha(Q)} \mid Q: \text{ cycle with } \alpha(Q) > 0\right\} \tag{2.9}$$

is well defined and M is ε-optimal with $\varepsilon = -\mu > 0$. Hence we have the following statement.

LEMMA 2.4. *Condition* (2.7) *is satisfied if and only if* M *is* ε*-optimal for some* $\varepsilon \geq 0$.

Proof. In addition to the above argument, note that the case $\varepsilon = 0$ corresponds to an optimal M for which (2.7) is vacuously true due to Theorem 4.3 of part I. □

Under condition (2.7) we define $\varepsilon(M)$ to be the minimum value of $\varepsilon \geq 0$ for which M is ε-optimal. The above argument shows, under (2.8), that

$$\varepsilon(M) = -\mu. \tag{2.10}$$

The following lemma substantiates step (i) of the minimum-ratio cycle algorithm.

LEMMA 2.5. *Assuming* (2.7) *and* (2.8) *let* Q *be a minimum-ratio cycle. Either* Q *is admissible or else it induces a minimum-ratio cycle having a smaller number of arcs than* Q. *In particular, a minimum-ratio cycle having the smallest number of arcs is admissible.*

Proof. We modify the proof of Lemma 4.4 of part I. Let $\overline{B}^+$ and $\overline{B}^-$ be defined by (2.2) and (2.3). Suppose that Q is not admissible, and assume without loss of generality that $(B^+, \overline{B}^+)$ does not satisfy the unique-max condition. Take a maximum-weight perfect matching M' in $G(B^+, \overline{B}^+)$ for $\mathbf{M}^+$ as well as the potential function $\widehat{p}$ in Lemma 3.5 of part I. Put $Q' = (Q - A^+) \cup M'$, which is a collection of disjoint cycles, say $Q' = \cup_{j=1}^{l} Q'_j$. Then $\alpha(Q') = \alpha(Q)$ (since $\alpha(M') = \alpha(Q \cap A^+) = 0$), and

$$\gamma(Q') = \gamma(Q) + [\gamma(M') - \gamma(Q \cap A^+)] \tag{2.11}$$

holds. By the choice of M' we have $\gamma(Q') \leq \gamma(Q)$, which implies

$$\gamma(Q')/\alpha(Q') \leq \gamma(Q)/\alpha(Q) = \mu. \tag{2.12}$$

We claim that the equality holds in (2.12). In fact, (2.12) shows

$$\gamma(Q') = \sum_{j=1}^{l} \gamma(Q'_j) \;\leq\; \mu\, \alpha(Q') = \mu \sum_{j=1}^{l} \alpha(Q'_j),$$

whereas $\gamma(Q'_j) \geq \mu\alpha(Q'_j)$ for all j by (2.7) and (2.9). With the equality in (2.12) we obtain $\gamma(Q') = \gamma(Q)$ since $\alpha(Q') = \alpha(Q)$.

It then follows from (2.11) that

$$\gamma(Q \cap A^+) = \gamma(M') = -\widehat{\omega}^+(B^+, \overline{B}^+). \tag{2.13}$$

Hence, putting

$$M'' = Q \cap A^+ = \{(u_i, v_i) \mid i = 1, \ldots, m\},$$

we have $M'' \subseteq A^*$, where

$$A^* = \{(u,v) \mid u \in B^+ - \overline{B}^+, v \in \overline{B}^+ - B^+, \omega^+(B^+,u,v) - \widehat{p}(u) + \widehat{p}(v) = 0\},$$

and $\widehat{p}$ is the potential function in Lemma 3.5 of part I. (Note: (u_i, v_i) denoted arcs in M' in the proof of Lemma 4.4 of part I.)

Since $(B^+, \overline{B}^+)$ does not satisfy the unique-max condition, there exist distinct indices i_k $(k = 1, \ldots, q; q \geq 2)$ such that $(u_{i_k}, v_{i_{k+1}}) \in A^*$ for $k = 1, \ldots, q$, where $i_{q+1} = i_1$. Then

$$\omega^+(B^+, u_{i_k}, v_{i_{k+1}}) = \widehat{p}(u_{i_k}) - \widehat{p}(v_{i_{k+1}}) \qquad (k = 1, \ldots, q), \tag{2.14}$$

$$\omega^+(B^+, u_{i_k}, v_{i_k}) = \widehat{p}(u_{i_k}) - \widehat{p}(v_{i_k}) \qquad (k = 1, \ldots, q), \tag{2.15}$$

$$\sum_{k=1}^{q} \gamma(u_{i_k}, v_{i_{k+1}}) = \sum_{k=1}^{q} \gamma(u_{i_k}, v_{i_k}) \tag{2.16}$$

hold true, where (2.15) is due to $M'' \subseteq A^*$.

For $k = 1, \ldots, q$, let $P(v_{i_{k+1}}, u_{i_k})$ denote the path on Q from $v_{i_{k+1}}$ to u_{i_k}, and let Q_k be the directed cycle formed by arc $(u_{i_k}, v_{i_{k+1}})$ and path $P(v_{i_{k+1}}, u_{i_k})$. By a similar argument as in the proof of Lemma 4.4 of part I we obtain

$$\sum_{k=1}^{q} (\gamma(Q_k) - \mu\alpha(Q_k)) = q'(\gamma(Q) - \mu\alpha(Q)) = 0$$

for some q' with $1 \leq q' < q$, which shows $\gamma(Q_k) - \mu\alpha(Q_k) = 0$ for each k. Therefore Q_k is a minimum-ratio cycle for k with $\alpha(Q_k) > 0$, while such k exists since $\sum_{k=1}^{q} \alpha(Q_k) = \alpha(Q) > 0$. □

LEMMA 2.6. *Assuming* (2.7) *and* (2.8), *let* Q *be an admissible minimum-ratio cycle. Then* $\overline{M}$ *is an independent assignment with* $\Omega(\overline{M}) = \Omega(M) - \gamma_M(Q)$.

Proof. This is the same as the proof of Lemma 4.5 of part I, except that (2.13) is used. □

LEMMA 2.7. *Assuming* (2.7) *and* (2.8), *let* Q *be an admissible minimum-ratio cycle. Then* $\varepsilon(\overline{M}) \leq \varepsilon(M)$ *for* $\overline{M} = (M - \{a \in M \mid \overline{a} \in Q \cap M^\circ\}) \cup (Q \cap A^\circ)$.

Proof. Put $\varepsilon = \varepsilon(M)$, which is equal to $-\mu$ by (2.10). By the ε-optimality of M,

$$\gamma_p(a) \equiv \gamma(a) + p(\partial^+ a) - p(\partial^- a) \geq -\varepsilon\alpha(a) \qquad (a \in \widetilde{A})$$

holds for some p. Note that

$$\gamma_p(a) = -\varepsilon\alpha(a) \qquad (a \in Q). \tag{2.17}$$

Denote by $G_{\overline{M}} = (\overline{V}, \overline{A})$ the auxiliary graph for $\overline{M}$, with obvious additional notations $\overline{A} = A^\circ \cup \overline{M}^\circ \cup \overline{A}^+ \cup \overline{A}^-$, $\overline{\gamma}$, and $\overline{\alpha}$. We will show

$$\overline{\gamma}_p(a) \equiv \overline{\gamma}(a) + p(\partial^+ a) - p(\partial^- a) \geq -\varepsilon\overline{\alpha}(a) \qquad (a \in \overline{A}) \tag{2.18}$$

for the same p. This is obvious for $a \in \overline{M}^\circ - M^\circ$ since $\overline{\alpha}(a) = 0$ and its reorientation $\overline{a} \in Q \cap A^\circ$ satisfies $\gamma_p(\overline{a}) = 0$.

In what follows we show (2.18) for $a \in \overline{A}^+$; the proof for the remaining case with $a \in \overline{A}^-$ is similar. We abbreviate ω^+, V^+, B^+, and $\overline{B}^+$ to ω, V, B, and $\overline{B}$, respectively. Then (2.18) for $a \in \overline{A}^+$ can be written as

$$\omega(\overline{B}, u, v) \leq p(u) - p(v) \qquad (u \in \overline{B}, v \in V - \overline{B}) \tag{2.19}$$

since $\omega(\overline{B}, u, v) = -\infty$ if $(u, v) \notin \overline{A}^+$.

Recalling the definition

$$\gamma(a) = -\omega(B, u, v) \qquad (a = (u, v) \in A^+)$$

and noting $\alpha(a) = 0$ $(a \in A^+)$, we see from (2.5) that

$$\omega(B, u, v) \leq p(u) - p(v) \qquad (u \in B, v \in V - B). \tag{2.20}$$

(Note that $(u, v) \notin A^+$ implies $\omega(B, u, v) = -\infty$.) Equation (2.17) shows that this is satisfied with equality for $(u, v) \in Q \cap A^+$. Hence

$$\widehat{\omega}(B, \overline{B}) = \sum_{u \in B - \overline{B}} p(u) - \sum_{v \in \overline{B} - B} p(v). \tag{2.21}$$

For $u \in \overline{B}$ and $v \in V - \overline{B}$ put $B' = \overline{B} - u + v$. It follows from Lemma 1.1 (upper-bound lemma), Lemma 1.2 (unique-max lemma), (2.20), and (2.21) that

$$\begin{aligned}
\omega(\overline{B}, u, v) &= \omega(B') - \omega(\overline{B}) \\
&\leq \widehat{\omega}(B, B') - \widehat{\omega}(B, \overline{B}) \\
&\leq \left[\sum_{u' \in B - B'} p(u') - \sum_{v' \in B' - B} p(v') \right] - \left[\sum_{u' \in B - \overline{B}} p(u') - \sum_{v' \in \overline{B} - B} p(v') \right] \\
&= p(u) - p(v).
\end{aligned}$$

Thus (2.19) is established. It may be remarked that the essence of (2.19) lies in Lemma 3.9 of part I. □

Combining Lemmas 2.4 and 2.7 we see that condition (2.7) is preserved in updating an independent matching in step (iii) of the minimum-ratio cycle algorithm. That is, we have the following.

LEMMA 2.8. *Assuming* (2.7) *and* (2.8), *let* Q *be an admissible minimum-ratio cycle. Then the condition* (2.7) *is satisfied by* $\overline{M}$.

We have justified all the claims about the cycle-canceling algorithm with minimum-ratio cycle selection.

Remark 2.2. The validity of the variants using the minimum-mean cycle can be shown similarly. To be specific, we have the following.

LEMMA 2.9. *Let* Q *be a minimum-mean cycle selected by rule (N) or (L). Then* Q *is admissible and* $\overline{M}$ *is an independent assignment with* $\Omega(\overline{M}) = \Omega(M) - \gamma_M(Q)$.

Proof. Put $\alpha \equiv 1$ in the proof of Lemma 2.5. This shows that if Q were not admissible, each induced cycle Q_k would be a minimum-mean cycle. Case (N): We have $|Q_k| < |Q|$, which is a contradiction. Case (L): The choice of Q implies that $\pi(v_{i_k}) < \pi(v_{i_{k+1}})$ for $k = 1, \ldots, q$, which is a contradiction since $i_{q+1} = i_1$. The proof of the second half is the same as the proof of Lemma 2.6. □

It should be emphasized, however, that no polynomial bound on the number of iterations can be deduced from the above lemma.

3. Augmenting algorithm.

3.1. Algorithms. Our augmenting algorithm solves VIAP(k) for $k = 0, 1, 2, \ldots$ with the aid of the auxiliary graph $\widetilde{G}_{(M, B^+, B^-)} = (\widetilde{V}, \widetilde{A})$ introduced in §5 of part I. The vertex set $\widetilde{V}$ is given by

$$\widetilde{V} = V^+ \cup V^- \cup \{s^+, s^-\},$$

where s^+ and s^- are new vertices referred to as the source vertex and the sink vertex, respectively. The arc set $\widetilde{A}$ consists of eight disjoint parts:

$$\widetilde{A} = (A^\circ \cup M^\circ) \cup (A^+ \cup F^+ \cup S^+) \cup (A^- \cup F^- \cup S^-),$$

where

$$\begin{aligned}
A^\circ &= \{a \mid a \in A\} \qquad \text{(copy of } A\text{)},\\
M^\circ &= \{\overline{a} \mid a \in M\} \qquad (\overline{a}\text{: reorientation of } a),\\
A^+ &= \{(u,v) \mid u \in B^+, v \in V^+ - B^+, u \in C^+(B^+,v)\},\\
F^+ &= \{(u,s^+) \mid u \in V^+\},\\
S^+ &= \{(s^+,v) \mid v \in B^+ - \partial^+ M\},\\
A^- &= \{(v,u) \mid u \in B^-, v \in V^- - B^-, u \in C^-(B^-,v)\},\\
F^- &= \{(s^-,u) \mid u \in V^-\},\\
S^- &= \{(v,s^-) \mid v \in B^- - \partial^- M\}.
\end{aligned}$$

The arc length $\gamma(a) = \gamma_{(M,B^+,B^-)}(a)$ $(a \in \widetilde{A})$ is defined by

$$\gamma(a) = \begin{cases} -w(a) & (a \in A^\circ),\\ w(\overline{a}) & (a = (u,v) \in M^\circ, \overline{a} = (v,u) \in M),\\ -\omega^+(B^+,u,v) & (a = (u,v) \in A^+),\\ -\omega^-(B^-,u,v) & (a = (v,u) \in A^-),\\ 0 & (a \in F^+ \cup S^+ \cup F^- \cup S^-). \end{cases} \tag{3.1}$$

LEMMA 3.1. *Let (M,B^+,B^-) be a feasible solution of VIAP(k). Problem VIAP($k+1$) has a feasible solution if and only if there exists a directed path in $\widetilde{G}_{(M,B^+,B^-)}$ from s^+ to s^-.*

Proof. First note that the graph $\widetilde{G}_{(M,B^+,B^-)}$ does not depend on w nor on $\omega^\pm$, except for the arc length. Then the claim follows from a standard result for the independent matching problem (e.g., [1], [12, Thm. 4.7]). □

Suppose that (M,B^+,B^-) is optimal for VIAP(k) and that VIAP($k+1$) is feasible. It follows from Lemma 3.1 that there is a (directed) path in $\widetilde{G}_{(M,B^+,B^-)}$ from the source s^+ to the sink s^- and from Theorem 5.2 of part I that there is a shortest path from s^+ to s^- with respect to γ. Let P be (the set of arcs on) a shortest path from s^+ to s^- having the smallest number of arcs. Then the following theorem holds true; the proof is given later.

THEOREM 3.2. *Let (M,B^+,B^-) be optimal for VIAP(k) and P be a shortest path, from the source s^+ to the sink s^- in $\widetilde{G}_{(M,B^+,B^-)}$, having the smallest number of arcs. Then $(\overline{M},\overline{B}^+,\overline{B}^-)$ defined by*

$$\overline{M} = M - \{a \in M \mid \overline{a} \in P \cap M^\circ\} + (P \cap A^\circ), \tag{3.2}$$

$$\overline{B}^+ = B^+ - \{\partial^+ a \mid a \in P \cap A^+\} + \{\partial^- a \mid a \in P \cap A^+\}, \tag{3.3}$$

$$\overline{B}^- = B^- - \{\partial^- a \mid a \in P \cap A^-\} + \{\partial^+ a \mid a \in P \cap A^-\}, \tag{3.4}$$

is optimal for VIAP($k+1$).

With this theorem, we obtain the following algorithm of augmenting type that solves VIAP(k) for $k = 0,1,2,\ldots$. At the beginning of the algorithm we set $M = \emptyset$

and find a maximum-weight base B^+ of $\mathbf{M}^+$ with respect to ω^+ and a maximum-weight base B^- of $\mathbf{M}^-$ with respect to ω^-. Obviously this choice gives the optimal solution to VIAP(0).

> AUGMENTING ALGORITHM (OUTLINE). Starting from the empty matching M and maximum-weight bases B^+ and B^- of $\mathbf{M}^+$ and $\mathbf{M}^-$, respectively, repeat (i)–(ii) below for $k = 0, 1, 2, \ldots$:
> (i) Find a shortest path P having the smallest number of arcs (from s^+ to s^- in $\widetilde{G}_{(M,B^+,B^-)}$ with respect to the arc length $\gamma_{(M,B^+,B^-)}$).
> [Stop if there is no path from s^+ to s^-.]
> (ii) Update (M, B^+, B^-) to $(\overline{M}, \overline{B}^+, \overline{B}^-)$ by (3.2), (3.3), (3.4).

Note that the graph $\widetilde{G}_{(M,B^+,B^-)}$ can be constructed in a similar manner as the graph $\widetilde{G}_M$ for the cycle-canceling algorithm.

Just like the primal-dual algorithm for the ordinary minimum-cost flow problem and the independent assignment problem, the algorithm outlined above can be made more efficient by the explicit use of a potential function $p : \widetilde{V} \to R$, the use of which has been invented independently by Tomizawa [26] and by Edmonds and Karp [6].

Suppose again that (M, B^+, B^-) is optimal for VIAP(k). By Theorem 5.2 of part I there is a potential $p : \widetilde{V} \to R$ such that

$$\gamma_p(a) \equiv \gamma(a) + p(\partial^+ a) - p(\partial^- a) \geq 0 \qquad (a \in \widetilde{A}). \tag{3.5}$$

This condition is equivalent to the following set of conditions appearing in Theorem 5.1 of part I:

$$w(a) - p(\partial^+ a) + p(\partial^- a) \begin{cases} \leq 0 & (a \in A), \\ = 0 & (a \in M), \end{cases} \tag{3.6}$$

$$B^+ \text{ is a maximum-weight base of } \mathbf{M}^+ \text{ with respect to } \omega^+[p^+], \tag{3.7}$$

$$B^- \text{ is a maximum-weight base of } \mathbf{M}^- \text{ with respect to } \omega^-[-p^-], \tag{3.8}$$

$$p(u) \geq p(v) \qquad (u \in V^+,\ v \in B^+ - \partial^+ M), \tag{3.9}$$

$$p(u) \leq p(v) \qquad (u \in V^-,\ v \in B^- - \partial^- M), \tag{3.10}$$

where it should be recalled from part I that $p^\pm$ denotes the restriction of p to $V^\pm$ and that

$$\omega^+[p^+](B) = \omega^+(B) + \sum_{v \in B} p^+(v) = \omega^+(B) + \sum_{v \in B} p(v) \qquad (B \in \mathcal{B}^+),$$

$$\omega^-[-p^-](B) = \omega^-(B) - \sum_{v \in B} p^-(v) = \omega^-(B) - \sum_{v \in B} p(v) \qquad (B \in \mathcal{B}^-)$$

are called the similarity transformations of $\omega^\pm$.

In the following algorithm we maintain a potential function p in addition to (M, B^+, B^-), and a shortest path is sought with respect to the modified arc length γ_p, which is nonnegative by virtue of (3.5). At the beginning of the algorithm the potential p is chosen as

$$p(v) = \begin{cases} 0 & (v \in V^+ + s^+), \\ -\max_{a \in A} w(a) & (v \in V^- + s^-), \end{cases} \tag{3.11}$$

which is easily seen to be legitimate. In the general steps p is updated to

$$\overline{p}(v) = p(v) + \Delta p(v) \qquad (v \in \widetilde{V}) \tag{3.12}$$

based on the length $\Delta p(v)$ of the shortest path from the source s^+ to v with respect to the modified arc length γ_p.

AUGMENTING ALGORITHM (WITH POTENTIAL).

(Step 0)

(i) Set $M = \emptyset$.

(ii) Define p by (3.11).

(iii) Find maximum-weight bases B^+ and B^- of $\mathbf{M}^+$ and $\mathbf{M}^-$, respectively.

(Step 1) Repeat (i)–(iii) below for $k = 0, 1, 2, \ldots$:

(i) Find a shortest path P having the smallest number of arcs (from s^+ to s^- in $\widetilde{G}_{(M,B^+,B^-)}$ with respect to the modified arc length γ_p).
[Stop if there is no path from s^+ to s^-.]

(ii) For each $v \in \widetilde{V}$ compute the length $\Delta p(v)$ of the shortest path from s^+ to v in $\widetilde{G}_{(M,B^+,B^-)}$ with respect to the modified arc length γ_p; update p to $\overline{p}$ by (3.12).

(iii) Update (M, B^+, B^-) to $(\overline{M}, \overline{B}^+, \overline{B}^-)$ by (3.2), (3.3), (3.4).

Remark 3.1. In the description of the algorithm above, we have assumed that $\Delta p(v)$ takes a finite value for all v in order to focus on the main ideas, which is common in the literature on matroid intersection algorithms. In actual implementations, however, this issue should be taken care of in an appropriate manner.

3.2. Validity of the augmenting algorithm. We show that $(\overline{M}, \overline{B}^+, \overline{B}^-, \overline{p})$ satisfies conditions (3.6)–(3.10). It then follows from Theorem 5.1 of part I that $(\overline{M}, \overline{B}^+, \overline{B}^-)$ is optimal for VIAP$(k+1)$. Theorem 3.2 also follows from this.

First note that $\overline{M}$ is a matching of size $k+1$ and that

$$\begin{aligned} \gamma_{\overline{p}}(a) &\equiv \gamma(a) + \overline{p}(\partial^+ a) - \overline{p}(\partial^- a) \\ &= \gamma_p(a) + \Delta p(\partial^+ a) - \Delta p(\partial^- a) \ \geq 0 \qquad (a \in \widetilde{A}) \end{aligned} \tag{3.13}$$

by the definition of Δp.

LEMMA 3.3.

$$w(a) - \overline{p}(\partial^+ a) + \overline{p}(\partial^- a) \begin{cases} \leq 0 & (a \in A), \\ = 0 & (a \in \overline{M}). \end{cases}$$

Proof. The first part follows from (3.13) for $a \in A^\circ$, while the second is due to

$$\gamma_p(a) + \Delta p(\partial^+ a) - \Delta p(\partial^- a) = \gamma(a) + \overline{p}(\partial^+ a) - \overline{p}(\partial^- a) = 0 \qquad (a \in M \cup P). \qquad \square$$

LEMMA 3.4.

$$\overline{p}(u) \geq \overline{p}(v) \ \ (u \in V^+, v \in \overline{B}^+ - \partial^+ \overline{M}), \quad \overline{p}(u) \leq \overline{p}(v) \ \ (u \in V^-, v \in \overline{B}^- - \partial^- \overline{M}).$$

Proof. The inequality (3.13) for $a = (u, s^+), (v, s^+), (s^+, v)$ implies $\overline{p}(u) - \overline{p}(s^+) \geq 0$ and $\overline{p}(v) - \overline{p}(s^+) = 0$. The proof for the second claim is similar. $\square$

Let

$$\{(u_i^+, v_i^+) \mid i = 1, \ldots, l^+\} = P \cap A^+,$$
$$\{(v_i^-, u_i^-) \mid i = 1, \ldots, l^-\} = P \cap A^-,$$

where $l^+ = |P \cap A^+|$, $l^- = |P \cap A^-|$, and the indices are chosen so that

$$u_1^+, v_1^+ \ , \ u_2^+, v_2^+ \ , \ \cdots \ , \ u_{l^+}^+, v_{l^+}^+$$

represents the order in which they appear on P and similarly for

$$v_{l^-}^-, u_{l^-}^- \ , \ \cdots \ , \ v_2^-, u_2^- \ , \ v_1^-, u_1^- .$$

We see

$$\overline{B}^+ = B^+ - \{u_1^+, \ldots, u_{l^+}^+\} + \{v_1^+, \ldots, v_{l^+}^+\} \supseteq \partial^+ \overline{M},$$
$$\overline{B}^- = B^- - \{u_1^-, \ldots, u_{l^-}^-\} + \{v_1^-, \ldots, v_{l^-}^-\} \supseteq \partial^- \overline{M}.$$

LEMMA 3.5. (1) $(B^+, \overline{B}^+)$ *and* $(B^-, \overline{B}^-)$ *satisfy the unique-max condition in* $\mathbf{M}^+$ *and* $\mathbf{M}^-$, *respectively.*

(2)

$$\widehat{\omega}(B^+, \overline{B}^+) = \sum_{i=1}^{l^+} \left(\overline{p}(u_i^+) - \overline{p}(v_i^+)\right),$$
$$\widehat{\omega}(B^-, \overline{B}^-) = -\sum_{i=1}^{l^-} \left(\overline{p}(u_i^-) - \overline{p}(v_i^-)\right).$$

Proof We prove the case "+" only and omit the superscript "+."

(1) By (3.13) for $a \in A^+$, we have

$$\omega(B, u_i, v_j) \leq \overline{p}(u_i) - \overline{p}(v_j) \qquad (1 \leq i, j \leq l).$$

Here we have an equality if $i = j$ and a strict inequality if $i < j$ by the definitions of $\overline{p}$ and P. Then the unique-max property follows from Lemma 3.5 of part I.

(2) We see from the above that $\overline{p}$ is the optimal dual variable in the sense of matching theory [20], and hence

$$\widehat{\omega}(B, \overline{B}) = \sum_{i=1}^{l} (\overline{p}(u_i) - \overline{p}(v_i)). \qquad \square$$

LEMMA 3.6.

$$\omega^+[\overline{p}^+](\overline{B}^+) \geq \omega^+[\overline{p}^+](B_1^+) \qquad (B_1^+ \in \mathcal{B}^+),$$
$$\omega^-[-\overline{p}^-](\overline{B}^-) \geq \omega^-[-\overline{p}^-](B_1^-) \qquad (B_1^- \in \mathcal{B}^-).$$

Proof. Again we prove the case "+" only. By Lemma 3.1 of part I it suffices to show

$$\omega[\overline{p}](\overline{B} - u + v) \leq \omega[\overline{p}](\overline{B}) \qquad (u \in \overline{B}, v \in V - \overline{B}).$$

Note first that

$$\omega[\overline{p}](\overline{B}-u+v)-\omega[\overline{p}](\overline{B})=\omega(\overline{B}-u+v)-\omega(\overline{B})-\overline{p}(u)+\overline{p}(v). \tag{3.14}$$

Here we have

$$\omega(\overline{B}-u+v)-\omega(B)\leq\widehat{\omega}(B,\overline{B}-u+v)\leq\sum_{u'\in B}\overline{p}(u')-\sum_{v'\in\overline{B}-u+v}\overline{p}(v')$$

by the upper-bound lemma (Lemma 1.1) and (3.13) for $a\in A^{+}$, and

$$\omega(\overline{B})-\omega(B)=\widehat{\omega}(B,\overline{B})=\sum_{i=1}^{l}(\overline{p}(u_i)-\overline{p}(v_i))$$

by Lemma 3.5 and the unique-max lemma (Lemma 1.2). Therefore the RHS of (3.14) is bounded by

$$\sum_{u'\in B}\overline{p}(u')-\sum_{v'\in\overline{B}-u+v}\overline{p}(v')-\sum_{i=1}^{l}(\overline{p}(u_i)-\overline{p}(v_i))-\overline{p}(u)+\overline{p}(v)=0. \qquad \square$$

Thus we have shown (3.6) in Lemma 3.3, (3.7) and (3.8) in Lemma 3.6, and (3.9) and (3.10) in Lemma 3.4. This completes the proof of Theorem 3.2.

Acknowledgments. The author expresses his gratitude to Satoru Iwata for bringing to his attention a number of important references concerning the primal approach and for suggesting a substantial improvement on the primal-dual algorithm. Discussion with Uwe Zimmermann on the minimum-ratio cycle algorithm was helpful to improve its presentation.

REFERENCES

[1] W. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*, to appear.

[2] W. Cui and S. Fujishige, *A primal algorithm for the submodular flow problem with minimum-mean cycle selection*, J. Oper. Res. Soc. Japan, 31 (1988), pp. 431–440.

[3] W. H. Cunningham and A. Frank, *A primal-dual algorithm for submodular flows*, Math. Oper. Res., 10 (1985), pp. 251–262.

[4] A. W. M. Dress and W. Wenzel, *Valuated matroid: A new look at the greedy algorithm*, Appl. Math. Lett., 3 (1990), pp. 33–35.

[5] ———, *Valuated matroids*, Adv. Math., 93 (1992), pp. 214–250.

[6] J. Edmonds and R. M. Karp, *Theoretical improvements in algorithmic efficiency for network flow problems*, J. Assoc. Comput. Mach., 19 (1972), pp. 248–264.

[7] A. Frank, *A weighted matroid intersection algorithm*, J. Algorithms, 2 (1981), pp. 328–336.

[8] ———, *An algorithm for submodular functions on graphs*, Ann. Discrete Math., 16 (1982), pp. 97–120.

[9] S. Fujishige, *A primal approach to the independent assignment problem*, J. Oper. Res. Soc. Japan, 20 (1977), pp. 1–15.

[10] ———, *An algorithm for finding an optimal independent linkage*, J. Oper. Res. Soc. Japan, 20 (1977), pp. 59–75.

[11] ———, *Algorithms for solving the independent-flow problems*, J. Oper. Res. Soc. Japan, 21 (1978), pp. 189–204.

[12] ———, *Submodular Functions and Optimization*, Annals of Discrete Mathematics 47, North-Holland, Amsterdam, 1991.

[13] A. V. Goldberg and R. E. Tarjan, *Finding minimum-cost circulations by canceling negative cycles*, J. Assoc. Comput. Mach., 36 (1989), pp. 873–886.

[14] M. IRI, *A practical algorithm for the Menger-type generalization of the independent assignment problem*, Mathematical Programming Study, 8 (1978), pp. 88–105.
[15] M. IRI AND N. TOMIZAWA, *An algorithm for finding an optimal "independent assignment"*, J. Oper. Res. Soc. Japan, 19 (1976), pp. 32–57.
[16] R. M. KARP, *A characterization of the minimum mean cycle in a digraph*, Discrete Math., 23 (1978), pp. 309–311.
[17] M. KLEIN, *A primal method for minimal cost flows*, Management Science, 14 (1967), pp. 205–220.
[18] E. L. LAWLER, *Matroid intersection algorithms*, Math. Prog., 9 (1975), pp. 31–56.
[19] ———, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.
[20] L. LOVÁSZ AND M. PLUMMER, *Matching Theory*, North-Holland, Amsterdam, 1986.
[21] S. T. MCCORMICK, *Approximate binary search algorithms for mean cuts and cycles*, Oper. Res. Lett., 14 (1993), pp. 129–132.
[22] N. MEGIDDO, *Combinatorial optimization with rational objective functions*, Math. Oper. Res., 4 (1979), pp. 414–424.
[23] K. MUROTA, *Computing the degree of determinants via combinatorial relaxation*, SIAM J. Comput., 24 (1995), pp. 765–796.
[24] ———, *Valuated matroid intersection* I: *Optimality criteria*, SIAM J. Discrete Math., 9 (1996), pp. 545–561.
[25] J. B. ORLIN AND R. K. AHUJA, *New scaling algorithms for the assignment and minimum cycle mean problems*, Math. Prog., 54 (1992), pp. 41–56.
[26] N. TOMIZAWA, *On some techniques useful for solution of transportation network problems*, Networks, 1 (1971), pp. 173–194.
[27] U. ZIMMERMANN, *Minimization on submodular flows*, Discrete Appl. Math., 4 (1982), pp. 303–323.
[28] ———, *Negative circuits for flows and submodular flows*, Discrete Appl. Math., 36 (1992), pp. 179–189.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 577–596, November 1996

005

PLANE EMBEDDINGS OF 2-TREES AND BICONNECTED PARTIAL 2-TREES*

ANDRZEJ PROSKUROWSKI†, MACIEJ M. SYSŁO‡, AND PAWEŁ WINTER§

Abstract. We consider different plane embeddings of partial 2-trees and give an efficient algorithm constructing a minimum cardinality cover of faces, where each face is covered by exactly one vertex. These tasks are facilitated by a unique tree representation of plane embeddings of 2-trees.

Key words. planar embeddings, treewidth, 2-trees, covering, minimum cost perfect matching

AMS subject classifications. 05C10, 05C30, 05C70

1. Introduction.

1.1. Motivation. Partial 2-trees constitute a nontrivial class of planar graphs that includes outerplanar graphs and series-parallel graphs. They admit efficient algorithms solving many inherently hard problems on general graphs [8, 1]. This property of algorithmic tractability follows from the tree-like structure of partial 2-trees and 2-trees, which are graphs imbedding partial 2-trees [12]. We propose a tree representation of 2-trees that is very useful in algorithmic treatment of these graphs.

The central algorithmic problem considered here is connected with plane embeddings of 2-trees and partial 2-trees. The union of minimal separators of any 2-tree has a very distinct structure which implies a one-to-one correspondence between certain subgraphs of partial 2-trees (outerplanar subgraphs pivotal for plane embedding) and the corresponding subgraphs of the imbedding 2-trees. This intermediate result complements the study of interior graphs of maximal outerplanar graphs in [4]. It also leads in a fairly straightforward manner to a formula counting the number of plane embeddings of biconnected 2-trees. (This problem for general planar graphs has been considered by MacLane [5].)

We investigate the existence of restricted covers of faces of a plane graph by vertices. This notion has been introduced in [9] and investigated in [10]. In the case of maximal outerplanar graphs, a tree representation of a plane embedding was used both to count the number of different embeddings (see [11]) and to construct an embedding admitting a perfect face independent vertex cover of all graph faces, FIVC. The proposed tree representation of 2-trees is crucial for constructing a perfect FIVC for 2-trees and partial 2-trees.

1.2. Definitions. We will deal with simple, loopless combinatorial graphs. An edge is *incident* with its *end vertices*, which are mutually *adjacent*. A *simple path* between two vertices u and v is a sequence of edges such that each of their end vertices (other than u or v) is incident with exactly two neighboring edges. If $u = v$, we have a *simple cycle*. A graph is *connected* if there is a path between any two of

* Received by the editors September 7, 1993; accepted for publication (in revised form) November 10, 1995.

† Department of Computer and Information Science, University of Oregon, Eugene, OR 97403 (andrzej@cs.uoregon.edu). The work of this author was supported in part by National Science Foundation grant NSF-CCR-9213439.

‡ Institute of Computer Science, University of Wrocław, ul. Przesmyckiego 20, 51151 Wrocław, Poland (syslo@ii.uni.wroc.pl).

§ Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark (pawel@diku.dk).

its vertices. In a connected graph, a subset S of vertices is a *separator* if its removal disconnects the graph. A tree is a connected acyclic graph. A graph G is *outerplanar* if there is an embedding of G in the plane such that all vertices lie on the boundary of the infinite region of the plane (the outer face). Thus, an outerplanar graph has no subgraph homeomorphic to (a subdivision of) the complete bipartite graph $K_{2,3}$. The set of *boundary cycles* consists of the boundaries of the faces, regions of the plane in a plane embedding. We identify a plane embedding of a graph with the set of its boundary cycles. A subgraph of G *induced* by a subset S of the vertices of G consists of S and all edges of G with both end vertices in S.

A 2-*tree* is either the complete graph on 3 vertices (the *triangle* K_3) or a graph with $n>3$ vertices obtained from a 2-tree G on $n-1$ vertices by adding a new vertex adjacent exactly to both end vertices of an edge of G. (An alternative definition involves construction of a 2-tree as the union of two smaller 2-trees that have an edge in common.) Every minimal separator of a 2-tree consists of the end vertices of an edge [7].

A *partial* 2-*tree* is a subgraph of a 2-tree (it can be imbedded in a 2-tree) with the same set of vertices. The class of partial 2-trees is identical with a slight generalization of series-parallel graphs, which are graphs with *treewidth* at most 2. It is well known that a graph is a partial 2-tree if and only if it contains no homeomorph of K_4. For emphasis, we will call 2-trees *full*. Also, we will distinguish between an *embedding* of a planar graph in the plane and an *imbedding* of a partial 2-tree in a full 2-tree.

We will use the following classification of edges in a full or partial 2-tree H. An edge $e=(a,b)$ is called *exterior* if $\{a,b\}$ is not a separator of H; otherwise, it is called *interior*. An edge $e=(a,b)$ is called *strongly interior* if the graph $H-\{a,b\}$ has more than two connected components and *weakly interior* otherwise. A strongly interior edge $e=(a,b)$ is *terminal* if and only if at most one of the graphs $G_i = H - \bigoplus_{j\neq i} C_j$ is not outerplanar. (Here, C_i's are the connected components of $H-\{a,b\}$, and $\bigoplus$ denotes the disjoint union of graphs.)

LEMMA 1.1. *Every nonouterplanar* 2-*tree* H *has a terminal strongly interior edge.*

Proof. If a 2-tree H is not outerplanar, it has a strongly interior edge since it contains a homeomorph of $K_{2,3}$. Assume that there is no such terminal edge. Then, there is a strongly interior edge e that separates H into at least two nonouterplanar components: C, which has the maximum size over all strongly interior edges and the corresponding components, and D. The latter has a strongly interior edge f separating H into connected components, one of which is nonouterplanar and properly includes C, thus contradicting the definition of C as maximum size. (This argument should give an intuition about the name of the terminal strongly interior edge.) □

Outerplanar 2-trees are known as *maximal outerplanar* graphs (*mops*), which are also identical with all triangulations of polygons.

2. Structure of 2-trees and their separators.

2.1. Interior graphs of 2-trees. Hedetniemi, Proskurowski, and Sysło [4] define the interior graph of a *mop* as the union of its interior edges (see also [6]). They completely characterize the interior graphs of mops and show that such a graph is a union of mops and attached to them caterpillars. We obtain a similar result for partial 2-trees.

LEMMA 2.1. *Any tree is the interior graph of some* 2-*tree.*

Proof. (This is proved by induction on the number of vertices.) By inspection, the lemma is true for $n=2$ and $n=3$ vertices. For $n\geq 3$, consider a tree T with

$n+1$ vertices. Unless $T = K_{1,n}$ (a star) we can split T into smaller trees T_1 and T_2 by removing an edge e so that $|T_i| + 1 \leq n$ $(i = 1, 2)$. By the inductive hypothesis, each of the trees T_i augmented by e is the interior graph of a 2-tree G_i $(i = 1, 2)$. A 2-tree G obtained from G_1 and G_2 by identifying the copies of e in each of them has T as its interior graph. As for the remaining case, $K_{1,n}$ is the interior graph of the 2-tree with a universal vertex and n remaining vertices inducing a path (a wheel without one external edge). □

THEOREM 2.2. *A connected partial 2-tree H is the interior graph of some 2-tree if and only if it has no induced cycles of length greater than 3.*

Proof. Sufficiency: any such H has biconnected components, H_i, that are either edges or 2-trees. A 2-tree component H_i is the interior graph of a 2-tree G_i obtained from H_i by adding a triangle (a vertex adjacent to both end vertices of an edge) to each exterior edge. An edge component H_i is the interior graph of a 2-tree G_i obtained by adding two triangles to H_i. Consider the union of such graphs G_i (identifying the corresponding copies of articulation vertices of H). For each articulation point v of G, choose from each connected component of $G-v$ an edge e of H incident to v and connect the other end vertices of these edges by a path. This results in a 2-tree that has H as its interior graph.

Necessity: let a partial 2-tree H be the interior graph of a 2-tree G. As a chordal graph, G has no induced chordless cycles other than triangles. Removing all vertices of degree 2 from G does not introduce any induced cycles. Since those vertices are incident to all exterior edges of G, their removal results in the interior graph, H, also without induced cycles of length greater than 3. □

The necessity result of [4] that the interior graph of a mop is a union of mops and caterpillars is an immediate corollary of Theorem 2.2 (since caterpillars are the only acyclic interior graphs of mops). However, the outerplanarity constitutes a nontrivial hindrance for the sufficiency of this condition.

2.2. Tree representation of 2-trees. We will represent a plane embedding of a 2-tree by a rooted, ordered tree with sibling nodes (children of the same parent) partitioned into two sets. To reach this goal, we first define a unique *associated graph* $D(G)$ of a given 2-tree G. $D(G)$ is the intersection graph of triangles of G over the set of edges. Thus, the vertices (called *nodes*) of $D(G)$ correspond to triangles of G, and edges of $D(G)$ correspond to edges of G that are in at least two triangles (an example can be found in Fig. 1b).

It can easily be verified that each node v of $D(G)$ is in at most three maximal cliques. Furthermore, there is at least one node in $D(G)$ that belongs to exactly one maximal clique. We will call such a node (and the corresponding triangle of G) *pendant.*

We will now give an algorithm constructing a graph that represents a plane embedding G_p of G (G_p is assumed to be specified by the set of boundary cycles). Let r be a pendant node in $D(G)$. Let T_r denote the directed tree rooted at r and obtained by the breadth-first traversal of $D(G)$ (Fig. 3b). Consider a node $v \in T_r$. Let T_v denote the maximal subtree of T_r rooted at v. Each node in T_v corresponds to a triangle in G. Let $G(v)$ denote a subgraph of G consisting of vertices and edges in the triangles corresponding to nodes in T_v. Let us define the natural relation of *inclusion* between triangles of a plane embedding: a triangle u is included in a triangle v if at least one of u's vertices is strictly inside the region bordered by v. We partition the children of v into the following three subsets:

- $In(v)$: nodes with the corresponding triangles included in v,

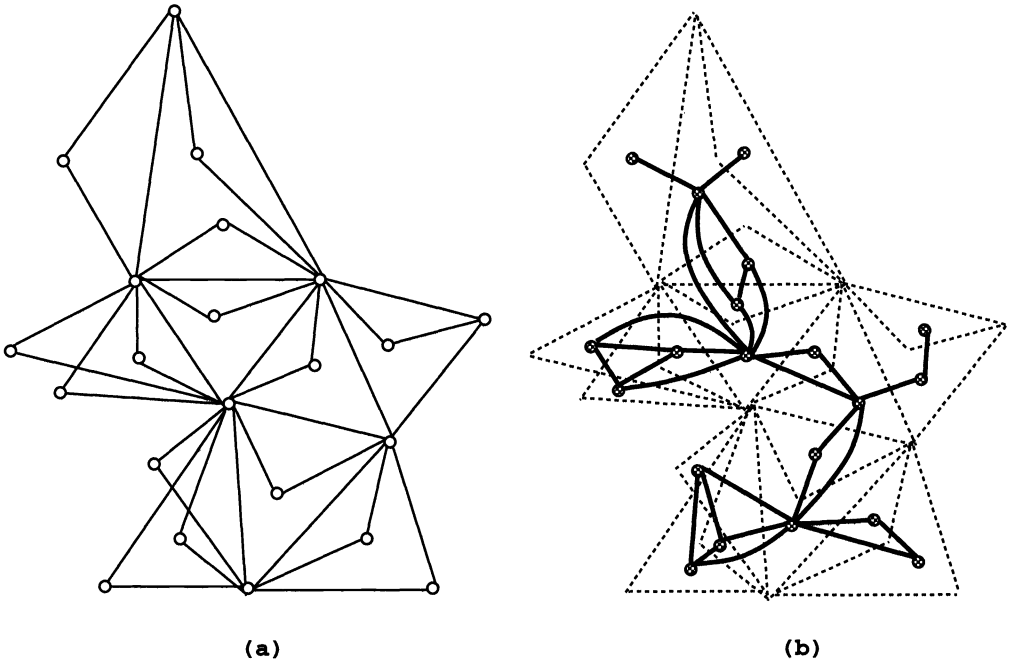


FIG. 1. *G and its associated graph D(G).*

- $Out(v)$: nodes with the corresponding triangles not related to v,
- $Cov(v)$: nodes with the corresponding triangles including v.

These nodes belong to at most two maximal cliques of $D(G)$ that also contain v. If $v = r$, then these nodes belong to just one clique. Further partition the nodes in $In(v)$ into $In_1(v)$ and $In_2(v)$ depending on to which of the two maximal cliques they belong. Let $Out_1(v)$ and $Out_2(v)$ (respectively, $Cov_1(v)$ and $Cov_2(v)$) be a similar partition of $Out(v)$ (respectively, $Cov(v)$). Order the nodes in each of the sets $In_1(v)$, $Out_1(v)$, $Cov_1(v)$, $In_2(v)$, $Out_2(v)$, $Cov_2(v)$ according to the relationship of the inclusion in the plane between the corresponding triangles of G (the triangle that includes all the others first). From now on we will treat these sets as ordered lists and $||$ will denote their concatenation; the equality will take under consideration the order of elements.

Note that either $Cov_1(v) = \emptyset$ or $Cov_2(v) = \emptyset$ due to the fact that G_p is planar. In fact, the planarity of G_p forces the set of nodes $U = \{u \in D(G) | Cov(u) \neq \emptyset\}$ to be on a single directed path in T_r. Let v be a node of $D(G)$ with $Cov(v) \neq \emptyset$ that is farthest away from the root r. Assume that $Cov_1(v) = \emptyset$ and $Cov_2(v) \neq \emptyset$ and let z_1 denote the first node in $Cov_2(v)$ (Fig. 2a). Consider the plane embedding (Fig. 2b) where

- the triangle z_1 is drawn outside of v (i.e., z_1 is removed from $Cov_2(v)$ and added in front of $Out_2(v)$),
- triangles corresponding to nodes in $Out_1(z_1)$ and $Out_2(z_1)$ are drawn inside z_1 in the same order (i.e., $Out_1(z_1)$ and $Out_2(z_1)$ become $In_1(z_1)$ and $In_2(z_1)$, respectively),
- triangles corresponding to nodes in $In_1(z_1)$ and $In_2(z_1)$ are drawn outside z_1 in the same order (i.e., $In_1(z_1)$ and $In_2(z_1)$ become $Out_1(z_1)$ and $Out_2(z_1)$,

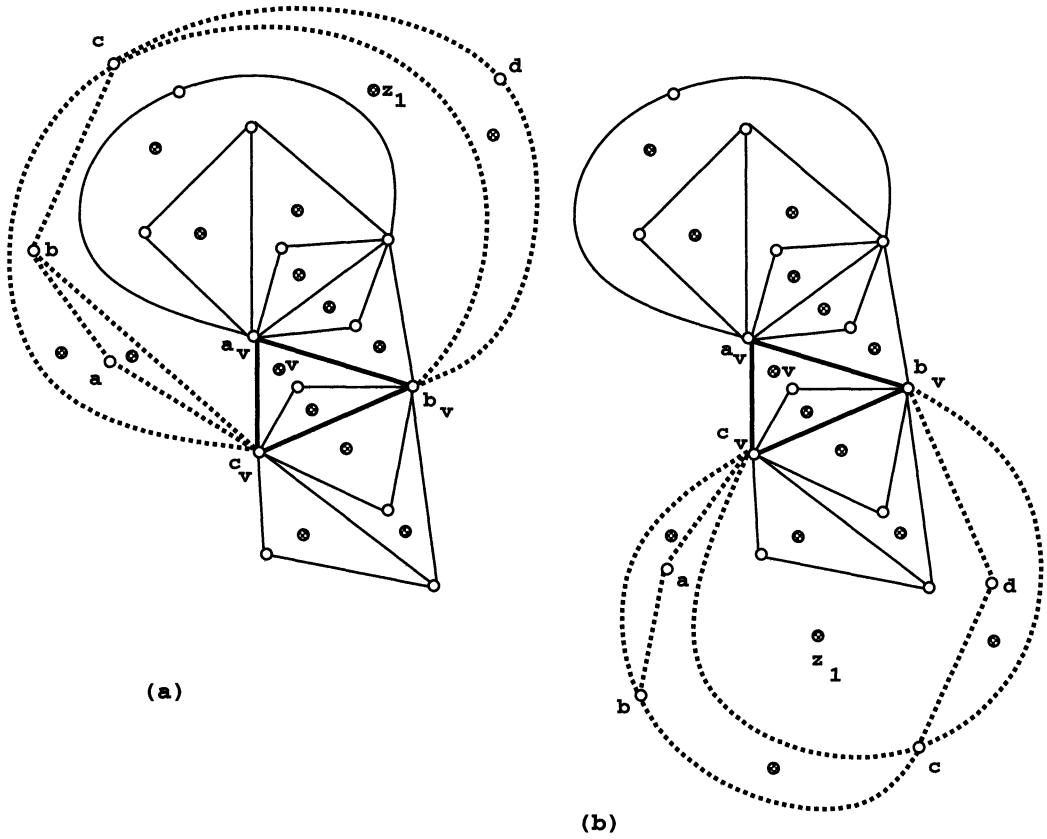


FIG. 2. *Identical plane embeddings.*

respectively).

Although the two drawings are different, they represent the same plane embedding. Since the above transformation can be applied repeatedly, we shall henceforth assume that $U = \emptyset$ (or, equivalently, $Cov_1(v) = \emptyset$ and $Cov_2(v) = \emptyset$ for all $v \in T_r$).

The tree T_r together with the ordered partitions $In_1(v)$, $Out_1(v)$, $In_2(v)$, $Out_2(v)$ for every node v in T_r other than r will be called an *in-graph* of T_r *rooted at* r (Fig. 3c). It will be denoted by $\vec{T}_r$. The root r will have all children in one partition $In(r), Out(r)$.

LEMMA 2.3. *Every plane embedding G_p of a full 2-tree G can be represented by an in-graph $\vec{T}_r$, where r is a pendant node in $D(G)$.*

Proof. The proof follows immediately from the above discussion. □

When drawing in-graphs as in Fig. 3c, we will use open (respectively, bold) circles to indicate nodes of *Out*-subsets (respectively, *In*-subsets). The order in the subsets will be indicated by directed dashed paths.

LEMMA 2.4. *Every in-graph $\vec{T}_r$ of G defines a unique plane embedding G_p of G.*

Proof. The embedding associated with $\vec{T}_r$ is obtained in the following manner (Fig. 4):

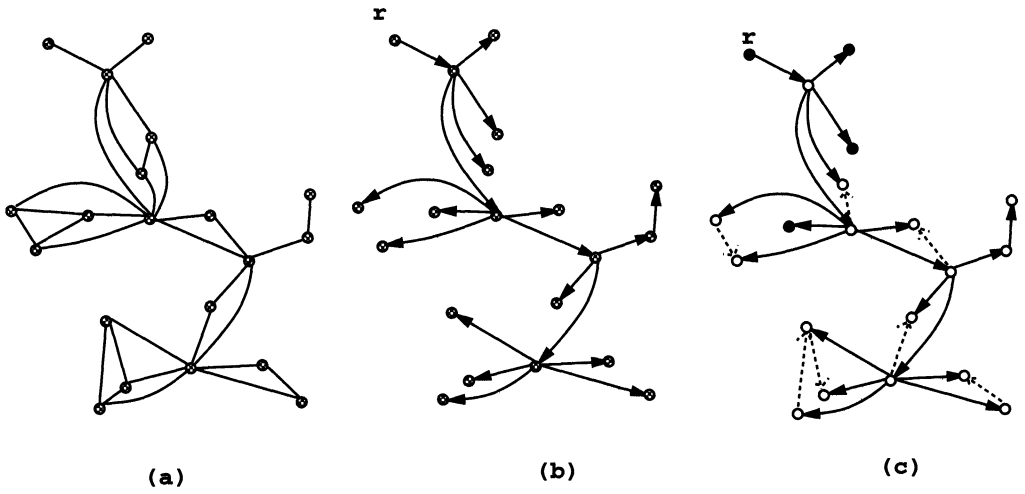


FIG. 3. *$D(G)$, its breadth-first tree T_r, and an in-graph $\overrightarrow{T}_r$.*

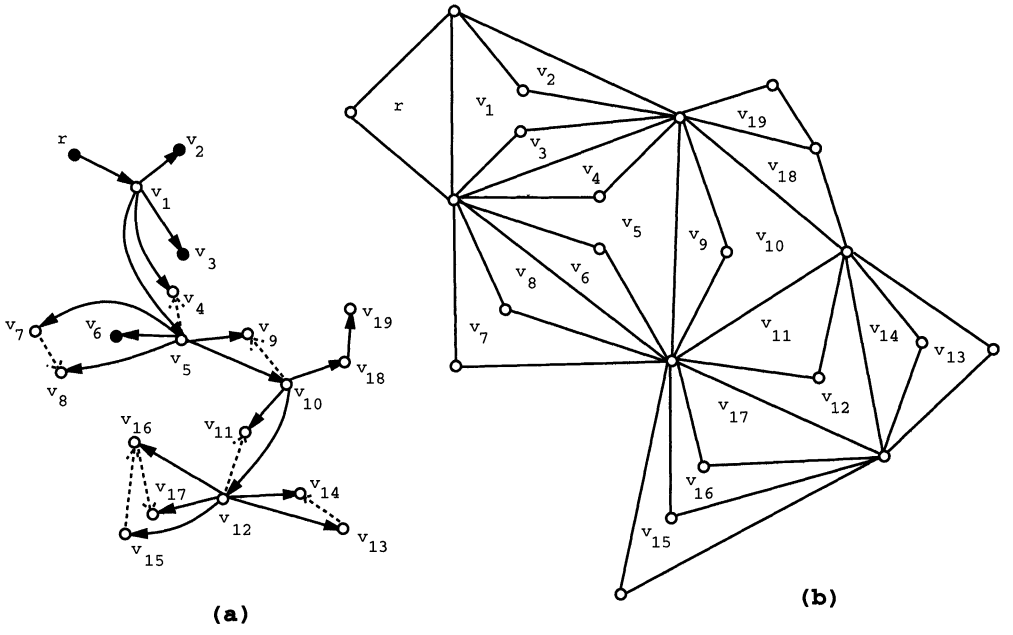


FIG. 4. *$\overrightarrow{T}_r$ and the corresponding embedding G_p.*

- Draw the triangle r.
- Traverse the nodes of $\overrightarrow{T}_r$ in any parent-first order. When leaving a node v, draw in the nested fashion triangles corresponding to $In_1(v)$ and $In_2(v)$ (respectively, $Out_1(v)$ and $Out_2(v)$) inside (respectively, outside) the triangle v. The ordering of triangles is given by the directed paths (first node corresponding to the outermost triangle). It is always possible to place triangles without violating the planarity of the already embedded subgraph. Note that

triangles drawn when traversing $In_1(v)$ and $In_2(v)$ (and their successors) will all be inside v. The unique face inside v that contains all three vertices on its boundary will be considered as the one corresponding to v. □

Let r be a pendant node of a pendant clique in $D(G)$. Let $\vec{T}_r$ be an in-graph rooted at r. Let us define a transformation of $G_p(v)$ called, informally, "turning inside-out" of G_p at v that swaps the In and Out subsets while preserving the order of their elements.

More formally, for a vertex v of T_r,

- $In'_1(v) = Out_1(v)$, $In'_2(v) = Out_2(v)$,
- $Out'_1(v) = In_1(v)$, $Out'_2(v) = In_2(v)$.

For $v = r$, the subscripts are immaterial. Define the in-graph $\phi(\vec{T}_r)$ obtained from $\vec{T}_r$ by turning G_p inside-out at r.

LEMMA 2.5. *Two distinct in-graphs, $\vec{T}_r$ and $\vec{T'}_r$, both rooted at the same pendant node r in $D(G)$, represent the same plane embedding of G if and only if $\phi(\vec{T}_r) = \vec{T'}_r$.*

Proof. It can easily be verified that if $\phi(\vec{T}_r) = \vec{T'}_r$, then the respective plane embeddings are identical.

Let In and Out be associated with T_r, and let In' and Out' be associated with T'_r. For the implication in the other direction, note that if $In_1(v) = In'_1(v)$, $Out_1(v) = Out'_1(v)$, $In_2(v) = In'_2(v)$, $Out_2(v) = Out'_2(v)$ for all nonroot nodes $v \in \vec{T}_r$; then there is nothing to prove. Therefore, let v be a node of $\vec{T}_r$ with at least one of the equalities violated. Assume that v is selected such that for all nodes on the path between v and the root r, the above equalities are satisfied. Let $v = \{a_v, b_v, c_v\}$ be the triangle in G corresponding to the node v. Assume without loss of generality (w.l.o.g.) that $In_1(v) \neq In'_1(v)$ or $Out_1(v) \neq Out'_1(v)$. Let $In_1(v) = \{x_1, x_2, ..., x_k\}$, $In'_1(v) = \{x'_1, x'_2, ..., x'_{k'}\}$, $Out_1(v) = \{y_1, y_2, ..., y_l\}$, and $Out'_1(v) = \{y'_1, y'_2, ..., y'_{l'}\}$. Let $u = \{a_u, b_u, c_u\}$ be the triangle in G corresponding to a node $u \in In_1(v)||Out_1(v) = In'_1(v)||Out'_1(v)$. Assume w.l.o.g. that $a_u = a_v$ and $c_u = c_v$ for all $u \in In_1(v)||Out_1(v)$.

Assume that $k \geq 1$ and consider the face corresponding to x_k in the plane embedding given by $\vec{T}_r$. This face contains no vertices b_u, $u \in In_1(v)||Out_1(v) \setminus \{x_k\}$. Neither does it contain b_v. Since $\vec{T}'_r$ represents the same embedding, x_k is either $x'_{k'}$ or $y'_{l'}$. By a similar argument, if $l \geq 1$, then y_l is either $y'_{l'}$ or $x'_{k'}$.

Assume that $k \geq 2$ and let x_i, x_{i+1} $(1 \leq i < k)$ be a pair of consecutive nodes in $In_1(v)$. Hence, the plane embedding must have a face with both b_{x_i} and $b_{x_{i+1}}$ on its boundary. Consequently, x_i and x_{i+1} must appear next to each other in either $In'_1(v)$ or $Out'_1(v)$. Similar arguments apply if $l \geq 2$ and y_j, y_{j+1} $(j \geq 1)$ is a pair of consecutive nodes in $Out_1(v)$.

It follows from the above arguments that either

- $In_1(v) = In'_1(v)$, $Out_1(v) = Out'_1(v)$ or
- $In_1(v) = Out'_1(v)$, $Out_1(v) = In'_1(v)$, and $In_1(v)||Out_1(v) \neq \emptyset$.

Assume that v is not the root and $In_1(v) = Out'_1(v)$ and $Out_1(v) = In'_1$. Assume that $In_1(v) \neq \emptyset$. Since x_1 is in front of $In_1(v)$, no face with b_{x_1} on its boundary can contain a vertex not in $G(v)$. But x_1 is also in front of $Out'_1(v)$ implying that at least one face with b_{x_1} on its boundary must contain a vertex outside $G(v)$, a contradiction. If $In_1(v) = \emptyset$ then $Out_1(v) \neq \emptyset$ leads to a similar contradiction. □

3. Counting plane embeddings.

3.1. Plane embeddings of 2-trees. A *frame* in a 2-tree H is a maximal (with respect to subgraph inclusion) outerplanar subgraph of H that does not contain any strongly interior edge as an interior edge. Any frame is also a mop. Strongly interior edges of a 2-tree partition it into frames (if one allows multiple copies of those edges).

We will first prove that a biconnected partial 2-tree has the same plane embeddings as any full 2-tree that imbeds it (modulo embeddings of its frames). Since the frames are outerplanar and the plane embeddings problem for outerplanar graphs has been solved [11], solving the problem for full 2-trees will imply the solution for partial 2-trees.

LEMMA 3.1. *A biconnected partial 2-tree G contains all exterior edges of any full 2-tree imbedding with the same set of vertices.*

Proof. Removal of an exterior edge from a 2-tree introduces an articulation point. If there were an imbedding H of G missing an exterior edge, it would separable and so would any partial graph of H. This contradicts the biconnectivity of G. □

Strongly interior edges of a 2-tree H partition H into maximal outerplanar components (frames) in the following sense: in every nonouterplanar 2-tree, there is a terminal strongly interior edge, say, $e = (a, b)$. Add the outerplanar graphs G_i (connected components C_i of $H - \{a, b\}$ augmented by $\{a, b\}$ and the adjacent edges) to the set of frames and remove the corresponding components C_i to obtain a 2-tree H'. Repeat the operation until only one edge remains.

LEMMA 3.2. *Any 2-tree imbedding H of a biconnected partial 2-tree G contains the same set of strongly interior edges.*

Proof. The lemma follows from the uniqueness of the set of exterior edges (see Lemma 3.1). If (a, b) is a strongly interior edge in an imbedding H of G, then the removal of $\{a, b\}$ disconnects G into more than two components. Since $H - \{a, b\}$ consists of at least three connected components, G contains three disjoint paths between a and b. A 2-tree imbedding of G in which any two of the three paths are connected by a path not using a or b would have a subgraph homeomorphic to K_4. This would contradict the absence of such a subgraph in partial (and thus also in full) 2-trees. Thus, $\{a, b\}$ is a separator in any 2-tree imbedding of G. □

3.2. Counting plane embeddings of 2-trees. Using the tree representation, it is a rather straightforward task to count all plane embeddings of 2-trees. Let v denote a node of an in-graph $\vec{T}_r$ of a 2-tree G, $v \neq r$. Let $In_1(v)||Out_1(v) = \{x_1, x_2, \ldots, x_{k_v}\}$ ($k_v \geq 0$) and $In_2(v)||Out_2(v) = \{y_1, y_2, \ldots, y_{l_v}\}$ ($l_v \geq 0$). There are $k_v!$ permutations of $In_1(v)||Out_1(v)$. Each permutation can be split in $k_v + 1$ ways such that the first i elements, $0 \leq i \leq k_v$, belong to $In_1(v)$ while the remaining $k_v - i$ elements belong to $Out_1(v)$. It follows immediately from Lemma 2.4 that the number of plane embeddings of G is

$$\pi(G) = \frac{1}{2} \Pi_{v \in \vec{T}_r} (l_v + 1)!(k_v + 1)!.$$

The coefficient $\frac{1}{2}$ is due to the fact that plane embeddings obtained by turning inside-out $In(r)$ and $Out(r)$ subsets at the root r are identical.

3.3. Counting plane embeddings of partial 2-trees. For a planar graph G, let $\pi(G)$ be the number of plane embeddings (i.e., embeddings with different sets of boundary cycles). Let $\pi'(G)$ be the number of plane embeddings of G when the outer face containing a specified edge is distinguished. (It is easy to see that this number is

independent of the particular edge chosen. When the graph G has at least two faces, then $\pi'(G) = 2\pi(G)$ since every edge is in exactly two faces.)

LEMMA 3.3. *Let $e = (a, b)$ be an interior edge of a 2-tree H with l components C_i of $H - \{a, b\}$. Let $H_i = H - \bigoplus_{j \neq i} C_j$. Then*

$$\pi(H) = \frac{1}{2} l! \prod_{1 \leq i \leq l} \pi'(H_i).$$

Proof. We will use the idea of permuting the components H_i to produce all embeddings of H while avoiding duplication by omitting embeddings related in a manner similar to the mapping ϕ of the preceding subsection. The proof will follow by induction on l:

(i) $l = 2$. Assume that H is drawn with a vertical edge e separating C_1 on the left of e from C_2 on the right of e and consider given embeddings of H_1 and H_2. The same embedding of H can be found among plane drawings of H with both C_1 and C_2 on one side of e. On the other hand, every embedding of H_1 and H_2 with the distinguished outer side of e multiplicatively contributes a new set of boundary cycles. Thus, $\pi(H) = \pi'(H_1) \cdot \pi'(H_2) = \frac{1}{2} l! \prod \pi'(H_i)$.

(ii) $l > 2$. Let x be an end vertex of e. For each H_i, choose an arbitrary edge e_i incident with x. An embedding of H is uniquely given by the position of e_l in the ordering of e_i around x and the given embeddings of the H_i's $(1 \leq i < l)$ fixing the "outer side" of e. Assuming $\pi(H - C_l) = \frac{1}{2}(l-1)! \prod_{1 \leq i < l} \pi'(H_i)$, each embedding of H_l contributes multiplicatively to the number of different sets of boundary cycles and the above observation proves the desired formula, since there are l possible positions for e_l. □

Note that the outerplanar case (with the embedding count given in [11] as $\pi(H) = 2^{f-2}$, where $f > 1$ is the number of interior faces in H) is a simple corollary of Lemma 3.3, since every separating edge of an outerplanar graph gives $l = 2$ and the absence of such an edge gives the base case of $\pi(H) = 1$. Since all mops of a given size have the same number of interior faces, the number of plane embeddings of a mop is completely determined by its size.

LEMMA 3.4. *Given a biconnected partial 2-tree G, the number of plane embeddings is the same for every 2-tree H imbedding G.*

Proof. By Lemma 2.3, any two 2-tree imbeddings of G differ at most on some subset of weakly interior edges. Yet, the sizes of the corresponding frames are identical. Since the frames of a 2-tree are maximal outerplanar, it follows by Lemma 3.3 that the number of plane embeddings of H is determined by the size of frames of H and their interactions through strongly interior edges of H. These are identical for all imbeddings of G. □

From these lemmas follows immediately a formula counting the number of plane embeddings for a partial 2-tree with minimal separators that induce edges.

THEOREM 3.5. *Let $\{a, b\}$ be a separator of a biconnected partial 2-tree G with l components C_i of $G - \{a, b\}$. Let $G_i = G - \bigoplus_{j \neq i} C_j$. Then*

(i) *if (a, b) is an edge of G, then $\pi(G) = \frac{1}{2} l! \prod_{1 \leq i \leq l} \pi'(G_i)$,*

(ii) *otherwise, $\pi(G) = \frac{1}{2}(l-1)! \prod_{1 \leq i \leq l} \pi'(G_i)$.*

Proof. Since the proof of (i) is almost identical with the proof of Lemma 3.3, we omit it.

In (ii), if $l = 2$, then either G is outerplanar or one can find a strongly interior edge as in (i). Let $\{a, b\}$ be a minimal separator of G not inducing an edge. Since

we assume that the number of connected components of $G - \{a, b\}$ is at least 3, (a, b) is an edge in any 2-tree imbedding H of G. We notice that in this case, the $l \geq 3$ components can be "permuted" in $\frac{1}{2}(l-1)!$ ways. Each subgraph G_i of G will be defined as $G - \bigoplus_{j \neq i} C_j$ *augmented* by the edge (a, b). (In the previous case of the separator inducing an edge, the edge $e = (a, b)$ acts as an extra component.) □

4. Perfect FIVC for full 2-trees. Let $G = (V, E)$ be a biconnected planar graph. A subset S of vertices is called a *perfect face-independent vertex cover* (or *perfect* FIVC, for short) of G if there exists a plane embedding G_p of G in which every face has exactly one vertex in S. A set W of cycles of a graph H is called a *perfect vertex-independent face cover* (or *perfect* VIFC for short) if there is a plane embedding H_p of H such that W is a subset of boundary cycles of faces and every vertex of H is in exactly one cycle of W. A perfect VIFC in this plane embedding of H is simply a 2-factor of H which consists of facial cycles. In the geometric dual G_p^* of G_p, a perfect FIVC of G corresponds to a set of faces of G_p^* which is a perfect VIFC of the vertices of G_p^*. The problems of the existence of perfect FIVCs and perfect VIFCs are NP-complete in general; see [3, 2]. When restricted to outerplanar graphs, both the existence problem and the problem of finding a perfect FIVC are solvable in linear time; see [10].

In this section, we describe a polynomial time algorithm that, given a full 2-tree G, finds a plane embedding G_p of G that admits a perfect FIVC or decides that no such embedding exists. In fact, the algorithm finds a plane embedding with a minimum cardinality perfect FIVC if a perfect FIVC exists. The algorithm follows an approach similar to that of [10]. It processes in a bottom-up manner the breadth-first search tree T_r of the associated graph $D(G)$.

If T_r consists of the root r alone, the problem is trivial. In the following, we assume that T_r contains at least two nodes. Let $v \in T_r$, $v \neq r$. Recall that T_v is the maximal subtree of T_r rooted at v and $G(v)$ is the corresponding subgraph of G. Consider the edge (u, v) of T_r entering v ($(u, v) \notin T_v$). Assume that the corresponding edge in G (and in $G(v)$) is (a_v, b_v). Define the following minimum cardinality covers among all plane embeddings of $G(v)$.

- $I(v)$ = face-independent vertex cover of all but the exterior face.
- $B(v)$ = face-independent vertex cover of all but the face corresponding to v and the exterior face.
- $F(v)$ = face-independent vertex cover of all but the face corresponding to v.
- $L(v)$ = face-independent vertex cover of all faces with the face corresponding to v and the exterior face covered by a_v.
- $R(v)$ = face-independent vertex cover of all faces with the face corresponding to v and the exterior face covered by b_v.
- $E(v)$ = face-independent vertex cover of all faces using neither a_v nor b_v.

Initially, for each pendant node $v = \{a_v, b_v, c_v\}$ of T_r, $v \neq r$, where (a_v, b_v) is a common edge with another triangle, we let the corresponding covers be (see Fig. 5) the following:

- $I(v)$ = undefined; it is impossible to cover the face corresponding to v without covering the exterior face.
- $B(v) = \emptyset$.
- $F(v)$ = undefined; it is impossible to cover the exterior face without covering the face corresponding to v.
- $L(v) = \{a_v\}$.
- $R(v) = \{b_v\}$.

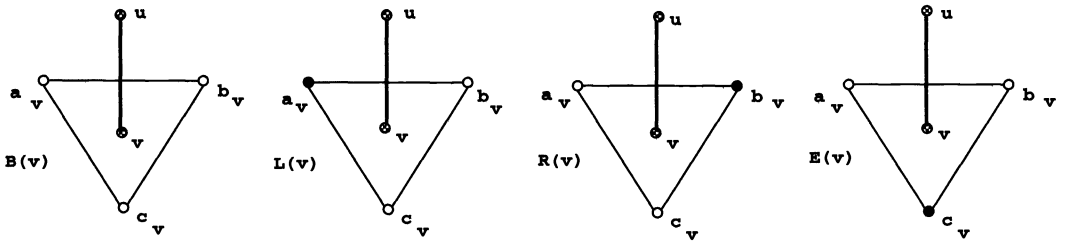


FIG. 5. *Existing covers of $G(v)$; v is a pendant node in T_r.*

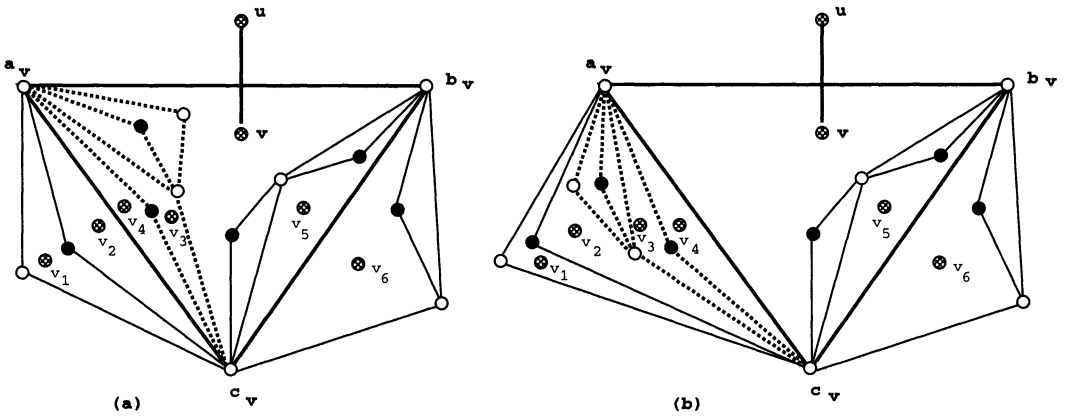


FIG. 6. *Two different plane embeddings of $G(v)$ admitting the same $I(v)$.*

- $E(v) = \{c_v\}$.

Any node for which the above six covers have been determined is said to be *labeled.* Hence, all pendant nodes of T_r other than r are initially the only labeled nodes. Assume that an unlabeled node v is chosen such that all its children in T_r are labeled. In the remainder of this section, we explain how to determine the six covers for v. Let $In_1(v)||Out_1(v) = \{x_1, x_2, ..., x_k\}$ and $In_2(v)||Out_2(v) = \{y_1, y_2, ..., y_l\}$ denote the children of v in T_v.

4.1. FIVC for all but the exterior face ($I(v)$). Suppose that $I(v)$ exists. Let $In_1(v), Out_1(v), In_2(v), Out_2(v)$ denote the ordered partition of children of v such that the corresponding embedding of $G(v)$ admits this $I(v)$. Either $In_1(v) \neq \emptyset$ or $In_2(v) \neq \emptyset$; otherwise, it would be impossible to cover the face corresponding to v by $I(v)$. Suppose that the face corresponding to v is covered by a vertex from the triangle corresponding to a node in T_{v_i}, $v_i \in In_2(v)$. Assume that $In_1(v) \neq \emptyset$ (Fig. 6a). The first node in $In_1(v)$ is a root of a subtree of T_v. None of the vertices of triangles corresponding to nodes in this subtree can cover the face corresponding to v; otherwise, this face would be covered twice. Consider the plane embedding of $G(v)$ obtained by adding $In_1(v)$ to the end of $Out_1(v)$, i.e., the embedding with $Out'_1(v) = Out_1(v)||In_1(v)$ and $In'_1(v) = \emptyset$. $I(v)$ is still an FIVC of $G(v)$ for all but the exterior face (Fig. 6b).

We can therefore first assume that $In_1(v) = \emptyset$ and search for the minimum cardinality $I_1(v)$ (or decide that it does not exist) among such embeddings of G. Then, the minimum cardinality $I_2(v)$ for embeddings of G with $In_2(v) = \emptyset$ is found

or it is decided that it does not exist. The smallest of these two is the desired $I(v)$. If none of them exists, then neither does $I(v)$.

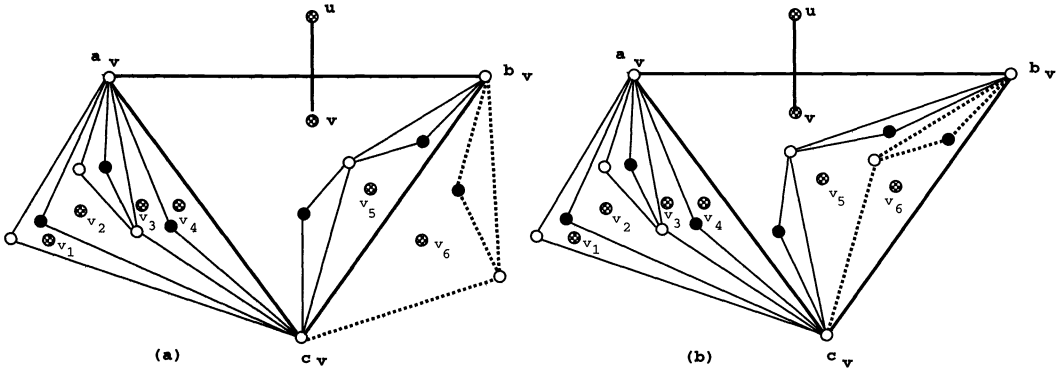


FIG. 7. *Two different plane embeddings of $G(v)$ admitting the same $I(v)$.*

Our analysis will determine feasible *covering sequences*, i.e., sets of vertex covers of T_x, for every child x of v. Let us assume w.l.o.g. that $In_1(v) = \emptyset$. Assume that $Out_2(v) \neq \emptyset$ (Fig. 7a). Consider the embedding of $G(v)$ obtained by adding $Out_2(v)$ to the end of $In_2(v)$, i.e., the embedding with $In_2'(v) = In_2(v)||Out_2(v)$ and $Out_2'(v) = \emptyset$. $I(v)$ is still an FIVC of $G(v)$ for all but the exterior face (Fig. 7b). Hence, when looking for the minimum cardinality $I(v)$ with $In_1(v) = \emptyset$, we can assume that $Out_2(v) = \emptyset$.

Suppose therefore that $In_1(v) = Out_2(v) = \emptyset$. Let $Out_1(v) = \{x_1, x_2, ..., x_k\}$ and $In_2(v) = \{y_1, y_2, ..., y_l\}$ with the indicated orders admitting $I(v)$. Then $G(x_1)$ must be covered by either $I(x_1)$ or $B(x_1)$; otherwise, the exterior face would be covered. If $G(x_1)$ is covered by $I(x_1)$, then $G(x_2)$ must be covered by either $I(x_2)$ or $B(x_2)$. If $G(x_1)$ is covered by $B(x_1)$, then $G(x_2)$ must be covered by either $F(x_2)$ or $E(x_2)$. If $G(x_2)$ is covered by $F(x_2)$, then $G(x_3)$ must be covered by either $F(x_3)$ or $E(x_3)$. If $G(x_2)$ is covered by $E(x_2)$, then $G(x_3)$ must be covered by either $I(x_3)$ or $B(x_3)$. Note that $G(x_k)$ must be covered by either $I(x_k)$ or $E(x_k)$. Hence, the covering sequence of $Out_1(v) = \{x_1, x_2, ..., x_k\}$ must be formed as a path in the forest shown in Fig. 8a with leaves being either $I(x_k)$ or $E(x_k)$.

Suppose that $I(x_i)$ $(3 \leq i \leq k)$ is preceded by $E(x_{i-1})$ (Fig. 9a with $i = 3$). Then we can place x_i in front of $Out_1(v)$ without affecting $I(v)$. Hence, we can assume that all $I(x_i)$ covers occur only in the beginning of the sequence $Out_1(v)$ (Fig. 9b).

By turning $G_p(z)$ inside-out at every child z of x_i, the cover $I(x_i)$ becomes $F(x_i)$ (and vice versa). Consequently, we can assume that no x_i $(2 \leq i \leq k-1)$ is covered by $F(x_i)$. Otherwise, we could place x_i in front of $Out_1(v)$ and cover it by $I(x_i)$ (Fig. 10 with $i = 3$).

It follows that at least one optimal covering sequence of $Out_1(v)$ in $I(v)$ is a path in the pruned forest shown in Fig. 8b with leaves being either $I(x_k)$ or $E(x_k)$.

Given these restrictions, we can now determine the optimal covering sequence of $Out_1(v)$ in $I(v)$, provided that one exists.

Consider a complete undirected graph K with $x_1, x_2, ..., x_k$ as its vertices. With every edge (x_i, x_j), associate the cost

$$\min\{|I(x_i)| + |I(x_j)|, |B(x_i)| + |E(x_j)|, |E(x_i)| + |B(x_j)|\}.$$

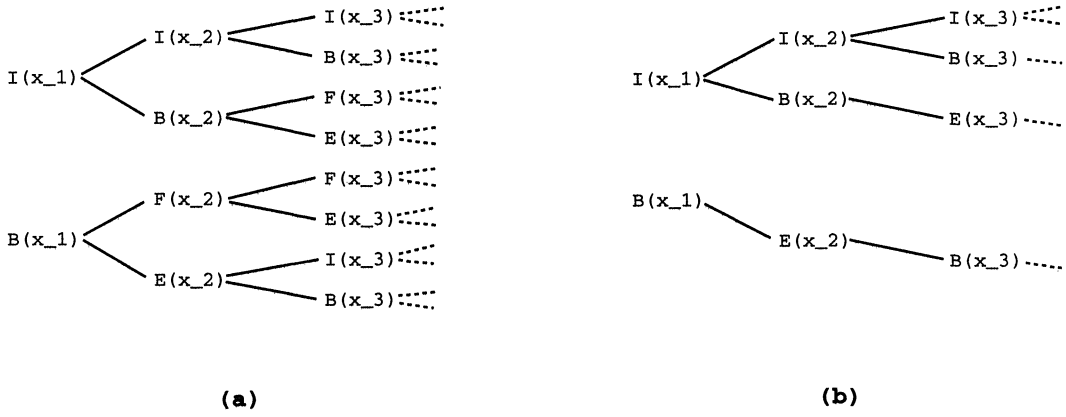


FIG. 8. *Covering sequences of* $Out_1(v) = \{x_1, x_2, ..., x_k\}$ *for* $I(v)$.

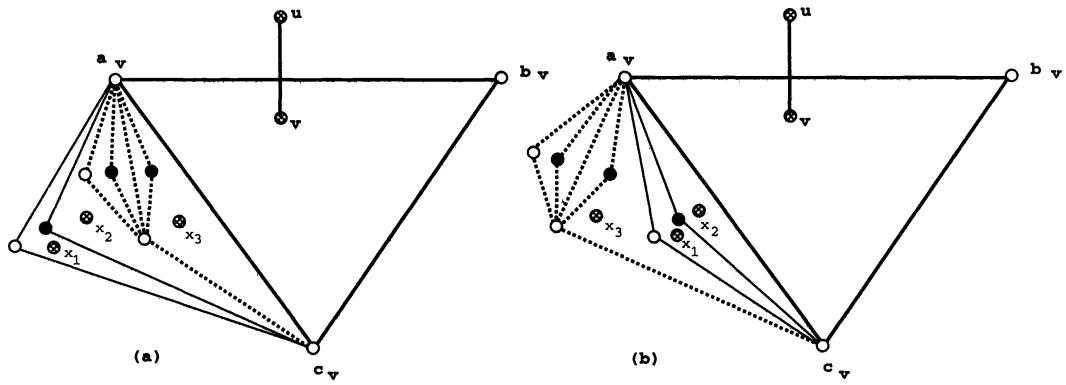


FIG. 9. *Equivalent covering sequences of* $Out_1(v)$ *in* $I(v)$.

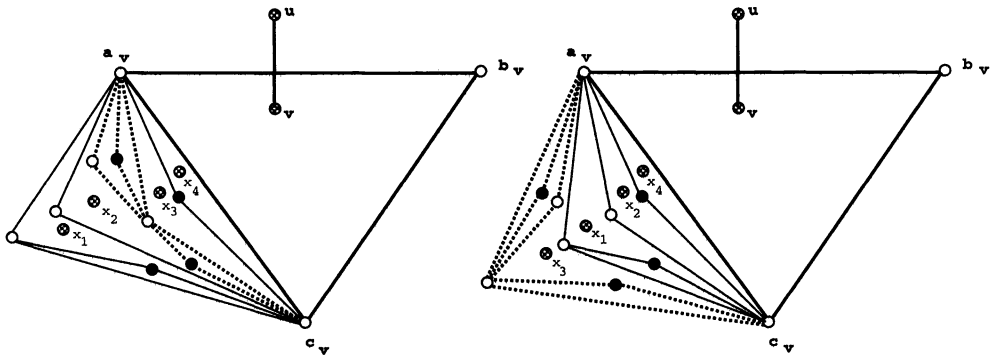


FIG. 10. *Equivalent covering sequences of* $Out_1(v)$ *in* $I(v)$.

For undefined covers, their cardinality is defined to be ∞. This cost gives the minimum cardinality of partial FIVCs of T_{x_i} and T_{x_j} that are compatible if x_i and x_j were to be placed consecutively in a plane embedding of G.

Suppose first that k is even. Solve the minimum cost perfect matching problem on K. End vertices of edges in this matching with costs determined by the first minimization term are placed in front of $Out_1(v)$ in any order. The remaining end vertices are then placed pairwise. The order within each such pair (x_i, x_j) depends on whether the edge cost was determined by the second minimization term (x_i precedes x_j) or by the third minimization term (x_j precedes x_i).

If k is odd, at least one of the subgraphs $G(x_m)$ $(1 \leq m \leq k)$ must be covered by $I(x_m)$. For each choice of x_m, we need to solve the minimum cost perfect matching problem M_m on the complete subgraph of K induced by $Out_1(v) \setminus \{x_m\}$. Select the matching M_m such that its cost together with $|I(x_m)|$ is minimized. Place this x_m in front of $Out_1(v)$. The remaining nodes of $Out_1(v)$ are ordered as in the case k even.

Let us now look at how to cover $In_2(v)$. $G(y_1)$ must be covered by either $F(y_1)$ or $E(y_1)$. If $G(y_1)$ is covered by $F(y_1)$, then $G(y_2)$ must be covered by either $F(y_2)$ or $E(y_2)$. If $G(y_1)$ is covered by $E(y_1)$, then $G(y_2)$ must be covered by either $I(y_2)$ or $B(y_2)$. If $G(y_2)$ is covered by $I(y_2)$, then $G(y_3)$ must be covered by either $I(y_3)$ or $B(y_3)$. If $G(y_2)$ is covered by $B(y_2)$, then $G(y_3)$ must be covered by either $F(y_3)$ or $E(y_3)$. Note that $G(y_l)$ must be covered by either $I(y_l)$ or $E(y_l)$. Hence, the covering sequence of $y_1, y_2, ..., y_l$ must be a path in the forest shown in Fig. 11a with leaves being either $I(y_l)$ or $E(y_l)$.

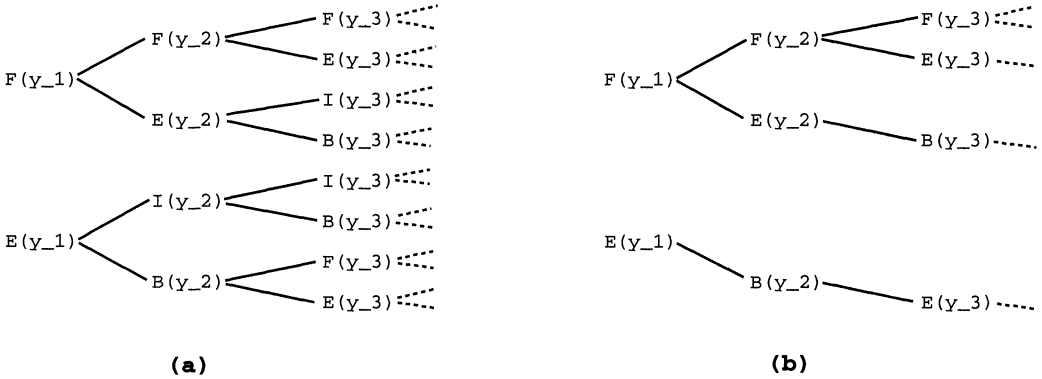


FIG. 11. *Covering sequences of* $In_2(v) = \{y_1, y_2, ..., y_l\}$ *for* $I(v)$.

Suppose that $F(y_i)$ $(3 \leq i \leq l-1)$ is preceded by $B(y_{i-1})$ (Fig. 12a with $i = 3$). Then we can place $F(y_i)$ in front of $In_2(v)$ without affecting $I(v)$. Hence, we can assume that $F(x_i)$ occurs only in the beginning of the covering sequence $In_2(v)$ (Fig. 12b).

As already mentioned, by turning $I(y_i)$ inside-out, we obtain $F(y_i)$ (and vice versa). Consequently, we can assume that no y_i $(3 \leq i \leq l-1)$ is covered by $I(y_i)$. If it were, we could place y_i in front of $In_2(v)$ and cover it by $F(y_i)$ (Fig. 13 with $i = 3$).

It follows that at least one optimal covering sequence of $In_2(v)$ in $I(v)$ is a path in the pruned tree shown in Fig. 11b with leaves being $E(x_l)$.

Given these restrictions, we can now determine the optimal covering sequence of $In_2(v)$ in $I(v)$, provided that one exists.

Consider a complete undirected graph K with $y_1, y_2, ..., y_l$ as its vertices. With

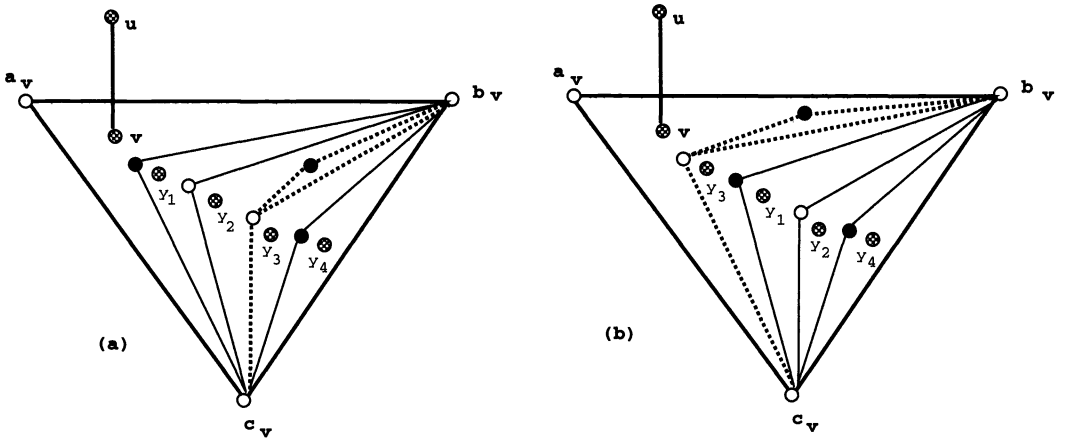


FIG. 12. *Equivalent covering sequences of* $In_2(v)$ *in* $I(v)$.

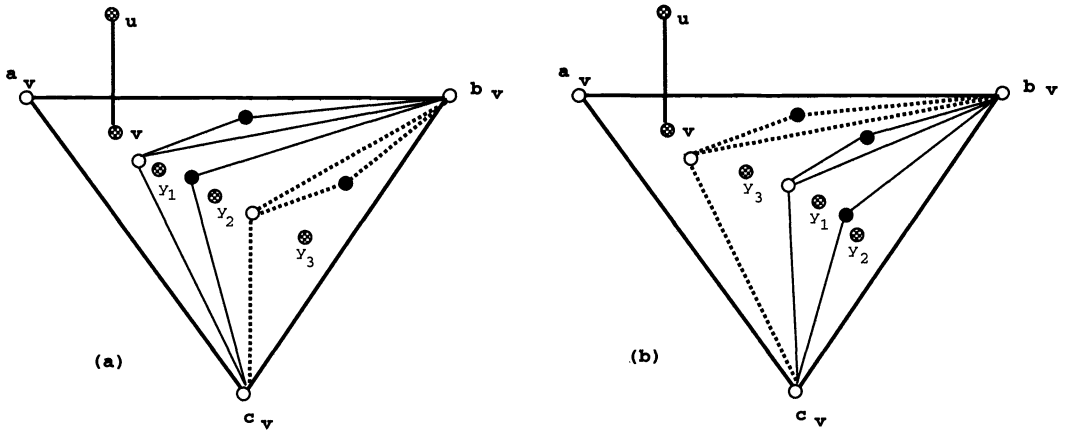


FIG. 13. *Equivalent covering sequences of* $In_2(v)$ *in* $I(v)$.

every edge (y_i, y_j) of K we associate the cost

$$\min\{|F(y_i)| + |F(y_j)|, |B(y_i)| + |E(y_j)|, |E(y_i)| + |B(y_j)|\}.$$

Suppose first that l is even. Then there is a pair of nodes y_m, y_n $(1 \leq m < n \leq l)$ which have covers $F(y_m)$, $E(y_n)$. For each choice of the pair y_m, y_n, solve the minimum cost perfect matching problem M_{mn} on the complete subgraph of K induced by $In_2(v) \setminus \{y_m, y_n\}$. Select the matching M_{mn} such that its cost together with $\min\{|F(y_m)| + |E(y_n)|, |E(y_m)| + |F(y_n)|\}$ is minimized. End vertices of edges in M_{mn} with their costs determined by the first minimization term $(|F(y_i)| + |F(y_j)|)$ are placed in front of $In_2(v)$ (in any order). If $|F(y_m)| + |E(y_n)| < |E(y_m)| + |F(y_n)|$, then y_m is placed in $In_2(v)$, followed by y_n. If this is not the case, then y_n is followed by y_m. The remaining end vertices of edges in M_{mn} are then placed pairwise. The order within each pair depends on whether the edge cost was determined by the second minimization term (y_i precedes y_j) or by the third minimization term (y_j precedes y_i).

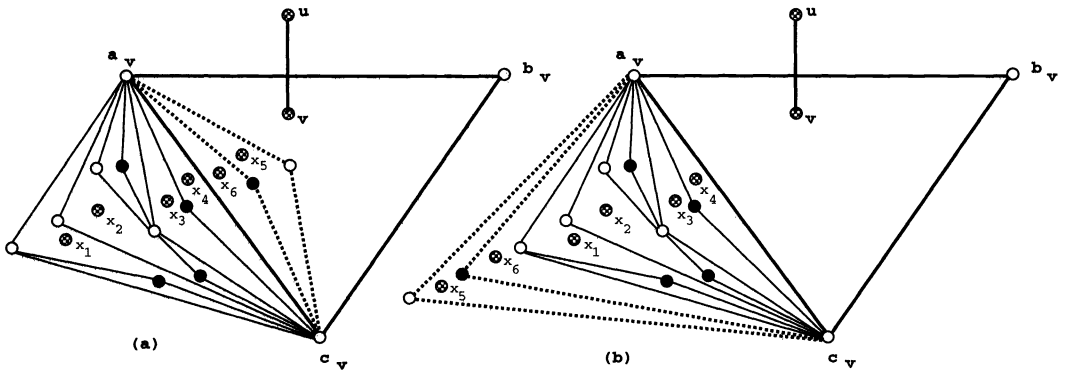


FIG. 14. *Equivalent covering sequences of* $In_1(v)$ *and* $Out_1(v)$ *in* $B(v)$.

Suppose now that l is odd. At least one of the subgraphs $G(y_m)$ $(1 \leq m \leq l)$ must be covered by $E(y_m)$. For each choice of y_m, we need to solve the minimum cost perfect matching problem M_m on the subgraph of K induced by $In_2(v) \setminus \{y_m\}$. Select the matching M_m such that its cost together with $|E(y_m)|$ is minimized. End vertices of edges in M_m with their cost determined by the first minimization term are placed in front of $In_2(v)$, followed by y_m. The remaining end vertices of edges in M_m are then placed pairwise. The order within each pair depends on whether the edge cost was determined by the second minimization term (y_i precedes y_j) or by the third minimization term (y_j precedes y_i).

4.2. FIVC for all but the face corresponding to v and to the exterior face ($B(v)$). We employ an argument similar to that of the preceding section. Suppose that $B(v)$ exists. Let $In_1(v), Out_1(v), In_2(v), Out_2(v)$ denote the ordered partition of children of v such that the corresponding embedding of $G(v)$ admits this $B(v)$. We can assume that $In_1(v) = In_2(v) = \emptyset$. Suppose, to the contrary, that $In_1(v) \neq \emptyset$ (Fig. 14a). Consider the embedding of $G(v)$ obtained by adding $In_1(v)$ at the front of $Out_1(v)$; i.e., $Out'_1(v) = In_1(v)||Out_1(v)$ and $In'_1(v) = \emptyset$. $B(v)$ is still an FIVC of $G(v)$ covering all faces except for the face corresponding to v and to the exterior face (Fig. 14b).

Suppose therefore that $In_1(v) = In_2(v) = \emptyset$. The order of nodes within $Out_1(v)$ and $Out_2(v)$ and the corresponding covering sequences can be determined in the same manner as the order and covering sequence of $Out_1(v)$ for $I(v)$ (Fig. 8).

4.3. FIVC for all but the face corresponding to v ($F(v)$). Suppose that $F(v)$ exists. Let $In_1(v), Out_1(v), In_2(v), Out_2(v)$ denote the ordered partition of children of v such that the corresponding embedding of $G(v)$ admits a minimum cardinality $F(v)$. We can then assume that $In_1(v) = In_2(v) = \emptyset$. If this is not the case, we can consider the embedding with $In'_1(v) = \emptyset$, $Out'_1(v) = Out_1(v)||In_1(v)$, $In'_2(v) = \emptyset$, $Out'_2(v) = Out_2(v)||In_2(v)$ (see Fig. 6). $F(v)$ is still an FIVC of $G(v)$ for all but the face corresponding to v.

Suppose therefore that $In_1(v) = In_2(v) = \emptyset$. The exterior face is covered either by the covering sequence of $Out_1(v)$ or by the covering sequence of $Out_2(v)$. The minimum sum of the corresponding cover cardinalities determines the desired FIVC as follows.

Assume that the exterior face is covered by a covering sequence of $Out_1(v)$. This

covering sequence of $Out_1(v)$ must be the same as the covering sequence of $In_2(v)$ for $I(v)$ (Fig. 11). The covering sequence of $Out_2(v)$ must be the same as the covering sequence of $Out_1(v)$ for $I(v)$ (Fig. 8).

Assume that the exterior face is covered by the covering sequence of $Out_2(v)$. The covering sequence of $Out_1(v)$ must be the same as the covering sequence of $Out_1(v)$ for $I(v)$ (Fig. 8). The covering sequence of $Out_2(v)$ must be the same as the covering sequence of $In_2(v)$ for $I(v)$ (Fig. 11).

4.4. FIVC for all faces using a_v ($L(v)$). Suppose that $L(v)$ exists. Let $In_1(v), Out_1(v), In_2(v), Out_2(v)$ denote the ordered partition of children of v such that the corresponding embedding of $G(v)$ admits this $L(v)$.

The assumption that $In_1(v) = \emptyset$, $In_2(v) = \emptyset$ can easily be verified. The covering sequence of $Out_1(v)$ must be $L(x_1), L(x_2), ..., L(x_k)$. The covering sequence of $Out_2(v)$ can be determined in the same manner as the covering sequence of $In_2(v)$ for $I(v)$ (see Fig. 11).

4.5. FIVC for all faces using b_v ($R(v)$). Suppose that $R(v)$ exists. Let $In_1(v), Out_1(v), In_2(v), Out_2(v)$ denote the ordered partition of children of v such that the corresponding embedding of $G(v)$ admits this $R(v)$.

The assumption that $In_1(v) = \emptyset$, $In_2(v) = \emptyset$ can easily be verified. The covering sequence of $Out_2(v)$ must be $R(x_1), R(x_2), ..., R(x_k)$. The covering sequence of $Out_1(v)$ can be determined in the same manner as the covering sequence of $In_2(v)$ for $I(v)$ (see Fig. 11).

4.6. FIVC for all faces using neither a_v nor b_v ($E(v)$). Suppose that $E(v)$ exists. Let $In_1(v), Out_1(v), In_2(v), Out_2(v)$ denote the ordered partition of children of v such that the corresponding embedding of $G(v)$ admits this $E(v)$.

Suppose first that the exterior face and the face corresponding to v are covered by c_v. We can then assume that $In_1(v) = In_2(v) = \emptyset$. Let $Out_1(v) = \{x_1, x_2, ..., x_k\}$ and $Out_2(v) = \{y_1, y_2, ..., y_l\}$. The covering sequence of $Out_1(v)$ must be $R(x_1), R(x_2), ..., R(x_k)$. Similarly, the covering sequence of $Out_2(v)$ must be $L(y_1), L(y_2), ..., L(y_l)$.

Assume now that c_v is not in $E(v)$. If $In_1(v)||Out_1(v) = \emptyset$ and $In_2(v)||Out_2(v) = \emptyset$, then $E(v)$ does not exist. Otherwise, we need to distinguish between the following cases:

- The exterior face is covered by a covering sequence of $Out_1(v)$, and the face corresponding to v is covered by a covering sequence of $In_2(v)$. We can assume that $Out_2(v) = In_1(v) = \emptyset$. Covering sequences of $Out_1(v)$ and $In_2(v)$ must be the same as the covering sequence of $In_2(v)$ for $I(v)$ (Fig. 11).
- The exterior face is covered by a covering sequence of $Out_2(v)$, and the face corresponding to v is covered by a covering sequence of $In_1(v)$. This case is analogous to the previous case.
- The exterior face is covered by a covering sequence of $Out_1(v)$, and the face corresponding to v is covered by a covering sequence of $In_1(v)$. Then we can assume that $In_2(v) = \emptyset$. Let $Out_2(v) = \{y_1, y_2, ..., y_l\}$. The covering sequence of $Out_2(v)$ must be the same as the covering sequence of $Out_1(v)$ for $I(v)$. Let $In_1(v) = \{x_1, x_2, ..., x_p\}$ and $Out_1(v) = \{x_{p+1}, x_{p+2}, ..., x_k\}$, $1 \leq p < k$. If $p = 1$ then $G(x_1)$ is covered by $E(x_1)$. Suppose that $p > 1$. $G(x_1)$ must be covered by either $F(x_1)$ or by $E(x_1)$; otherwise, the face corresponding to v in the embedding of $G(v)$ would be uncovered. Suppose that $G(x_1)$ is covered by $E(x_1)$ (Fig. 15a). Then $G(x_2)$ must be covered by either $I(x_2)$ or

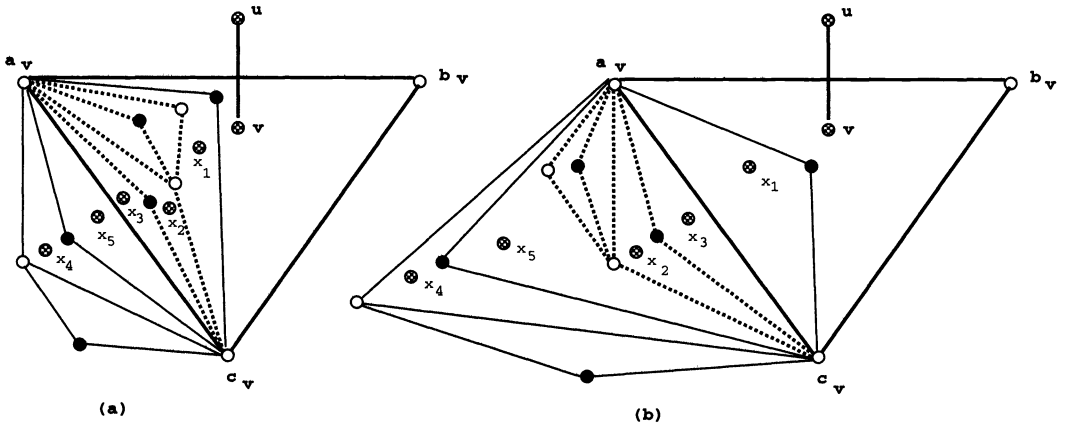


FIG. 15. *Equivalent covering sequences of* $In_1(v)$ *and* $Out_1(v)$ *in* $E(v)$.

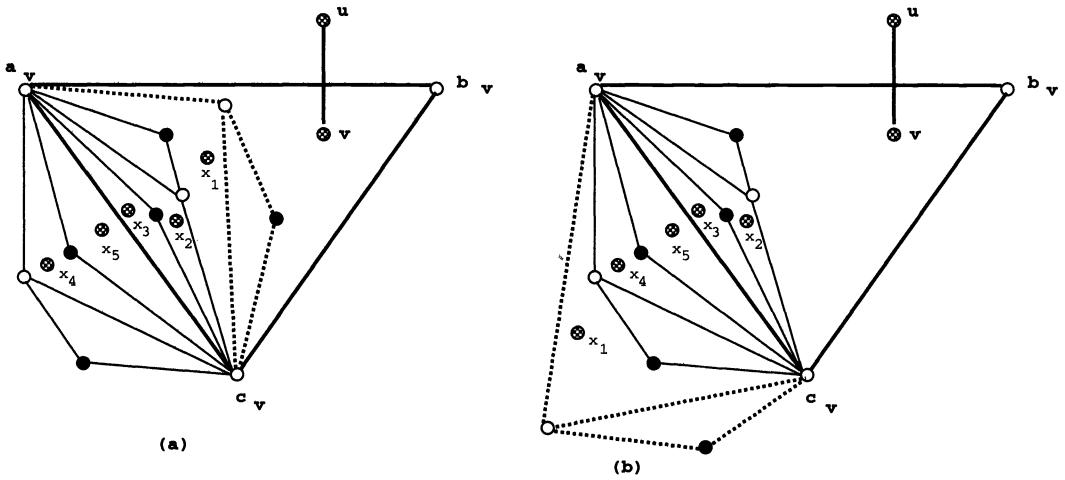


FIG. 16. *Equivalent covering sequences of* $In_1(v)$ *and* $Out_1(v)$ *in* $E(v)$.

$B(x_2)$. Let $Out_1'(v) = Out_1(v)||\{x_2, x_3, ..., x_p\}$ and $In_1'(v) = \{x_1\}$. $E(v)$ is still an FIVC for all faces (Fig. 15b).

Suppose that $G(x_1)$ is covered by $F(x_1)$ (Fig. 16a). Then $G(x_2)$ must be covered by either $F(x_2)$ or $E(x_2)$. Let $Out_1'(v) = \{x_1\}||Out_1(v)$ and $In_1'(v) = \{x_2, x_3, ..., x_p\}$. $E(v)$ is still an FIVC for all faces (Fig. 16b). Hence, the cardinality of $In_1(v)$ has been reduced by one. Either $|In_1(v)| = 1$ or one of the above two cases is applicable.

For each choice of a node x in $In_1(v)||Out_1(v)$ as x_1 (the only node in $In_1(v)$), we find the covering sequence of nodes in $Out_1(v)$. It can be determined in the same manner as the covering sequence of $In_2(v)$ for $I(v)$ (see Fig. 11). We select the covering sequence whose cardinality together with $|E(x)|$ is the smallest.

- The exterior face is covered by a covering sequence of $Out_2(v)$, and the face corresponding to v is covered by a covering sequence of $In_2(v)$. This case is analogous to the previous case.

The smallest among the above five covers is the desired $E(v)$.

4.7. Final FIVC determination. Suppose that $I(r)$, $B(r)$, $F(r)$, $L(r)$, $R(r)$, $E(r)$ have been determined for the root node r. Then the minimum cardinality perfect FIVC for G is the smallest of the covers $L(r)$, $R(r)$, $E(r)$.

5. Covering faces of partial 2-trees. The FIVC problem for a plane embedding of partial 2-trees is solvable in an almost identical manner to that of 2-trees. Given a partial 2-tree H, one has to produce an imbedding in a 2-tree G and then process G as for full 2-trees, with small modifications. To avoid repetition, we only will give the basic case of covering the subgraph $G(v)$ for some node v of the in-graph of G, by $I(v)$, i.e., a set of vertices covering all faces of $G(v)$ except for the exterior one. We will refer to the analysis of §4.1.

Let us assume that triangles corresponding to the children $x_1, \ldots, x_k$ of v in $T(G)$ (constituting, w.l.o.g., $Out_1(v)$) have as the common base an *added* edge, (a_v, c_v), which is in G but not in H and that triangles corresponding to children $y_1, \ldots, y_l$ of v (constituting $In_2(v)||Out_2(v)$) have as the common base the other edge of the triangle v of G, (b_v, c_v), which is also in H. We will consider two cases of constructing an $I(v)$ FIVC, depending on the manner in which the interior face corresponding to v is covered.

Suppose first that the interior face corresponding to v is covered by a vertex from $G(x_k)$. Then the analysis of the covering sequence of $Out_1(v)$ is the same as in the (full) 2-tree case for $I(v)$. By considering the result of setting $Out_2'(v) = Out_2(v)||In_2(v)$ (see Fig. 6), we can assume that $In_2'(v) = \emptyset$. Thus, the analysis of the covering sequence of $Out_2'(v)$ is the same as for $Out_1(v)$ in the 2-tree case.

Suppose next that the interior face corresponding to v is covered by a vertex from $G(y_1)$. We can then assume that $Out_2'(v) = \emptyset$, setting $In_2'(v)$ to $In_2(v)||Out_2(v)$ (see Fig. 7). The covering sequence of $In_2'(v)$ must be the same as the covering sequence of $In_2(v)$ in the (full) 2-tree case. The covering sequence of $Out_1(v)$ is almost the same as in the $I(v)$ covering of 2-trees. The only difference is that it must end in either $B(x_k)$ or $F(x_k)$. Since every $F(x_i)$ can be replaced by $I(x_i)$ and placed in front of the sequence, we can assume that the sequence ends in $B(x_k)$. In the remainder of this section, we discuss the analysis of the covering sequence of $Out_1(v)$ in this case.

If k is even, then the feasible sequences are of the form

$$I(x_1), \ldots, I(x_i), B(x_{i+1}), E(x_{i+2}), B(x_{i+3}), E(x_{i+4}), \ldots E(x_{k-1}), B(x_k),$$

where i is odd.

To find the minimum cardinality FIVC we define a graph K as for the 2-tree case, with similarly weighted edges. For each choice of two vertices x_m and x_n of K, we find M_{mn}, the minimum weight perfect matching for the subgraph of K induced by the remaining vertices of K. We then select the solution that minimizes the sum of the cost of M_{mn} and $|I(x_m)| + |B(x_n)|$ and construct the minimum cardinality FIVC as in the 2-tree case (with x_m and x_n as the vertices x_i, and x_{i+1}, respectively).

If k is odd, the feasible sequences are similar as above but the number i of the initial I covers is even. To find the minimum cardinality FIVC, one has to remove a vertex x_m of K, find the solution M_m for the matching problem for the subgraph of K induced by the remaining vertices, and then join with the removed vertex as x_k, minimizing the sum of the cost of M_m and $|B(x_m)|$.

The other five types of covers for partial 2-trees can be constructed in a similar manner, based on the analyses of the full 2-tree cases.

REFERENCES

[1] S. ARNBORG AND A. PROSKUROWSKI, *Linear time algorithms for NP-hard problems on graphs embedded in k-trees*, Discrete Appl. Math., 23 (1989), pp. 11–24.
[2] M. FELLOWS, Personal communication, 1986.
[3] M. FELLOWS, F. HICKLING, AND M. M. SYSŁO, *A topological parameterization and hard graphs problems*, Congr. Numer., 59 (1987), pp. 69–78.
[4] S. M. HEDETNIEMI, A. PROSKUROWSKI, AND M. M. SYSŁO, *Interior graphs of maximal outerplane graphs*, J. Combin. Theory Ser. B, 38 (1985), pp. 156–167.
[5] S. MAC LANE, *A structural characterization of planar combinatorial graphs*, Duke Math. J., 3 (1937), pp. 460–472.
[6] A. PROSKUROWSKI, *Separating subgraphs in k-trees: Cables and caterpillars*, Discrete Math., 49 (1984), pp. 275–285.
[7] D. ROSE, *On simple characterization of k-trees*, Discrete Math., 7 (1974), pp. 317–322.
[8] K. TAKAMIZAWA, T. NISHIZEKI, AND N. SAITO, *Linear-time computability of combinatorial problems on series-parallel graphs*, J. Assoc. Comput. Mach., 29 (1982), pp. 623–641.
[9] M. M. SYSŁO, *Independent face and vertex covers in plane graphs*, Banach Center Publ., 25 (1989), pp. 177–185.
[10] M. M. SYSŁO AND P. WINTER, *Independent covers in outerplanar graphs*, R. Karlsson, A. Lingas, eds., Lecture Notes in Computer Science 318, Springer-Verlag, Berlin, New York, 1988, pp. 242–254.
[11] ———, *In-trees and plane embeddings of outerplanar graphs*, BIT, 30 (1990), pp. 83–90.
[12] J. WALD AND C. J. COLBOURN, *Steiner trees, partial 2-trees, and minimum IFI networks*, Networks, 13 (1983), pp. 159–167.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 597–601, November 1996

006

ON THE TRELLIS COMPLEXITY OF THE DENSEST LATTICE PACKINGS IN $\mathbb{R}^n$*

IAN F. BLAKE† AND VAHID TAROKH†

Abstract. An inequality relating the trellis complexity of lattices to their dimension and Hermite parameter is established. Using this inequality, a conjecture of Forney is proved indicating that the trellis complexity of the densest lattice packings in $\mathbb{R}^n$ grows exponentially as a function of their coding gain.

Key words. rational lattices, coding gain, trellis complexity

AMS subject classification. 94B12

1. Introduction.

In two remarkable papers [2], [3], Forney observed that every rational lattice has a finite trellis diagram which can be employed for maximum likelihood decoding over the additive white Gaussian noise channel via the Viterbi algorithm. He also conjectured [3] that the trellis complexity of the densest lattice packings grows exponentially as a function of their coding gains. It is proved that this is indeed the case.

To formalize his conjectures, consider the category $\mathcal{L}$ of all lattices having finite trellis diagrams. This category trivially contains all the rational lattices, and it is well known that a lattice L of dimension n is in $\mathcal{L}$ if and only if there exists a sublattice of L of dimension n with a basis consisting of mutually orthogonal vectors.

For $L \in \mathcal{L}$ of dimension n, let $\mathcal{C}(L)$ denote the category of all finite trellis diagrams for L; then $\mathcal{C}(L)$ is nonempty. Let S and B, respectively, denote the minimum number of states and branches of elements of $\mathcal{C}(L)$. Define $\mathcal{S}(L)$, *the average state trellis complexity* of L, to be $(S-1)/n$ and $\mathcal{B}(L)$, *the average branch trellis complexity* of L, to be B/n [5].

For $L \in \mathcal{L}$, let $d(L)$ denote the minimum length of nonzero elements of L and $V(L)$ denote the fundamental volume of L [1]. For any lattice L, let $\delta(L) = d(L)^2/[V(L)]^{2/n}$ denote the coding gain of L. The coding gain is also known as Hermite's parameter in the number theory literature.

For $\gamma \geq 1$, define the functions

$$\mathcal{T}_1(\gamma) = \inf\{\mathcal{S}(L) \mid \delta(L) \geq \gamma \text{ and } L \in \mathcal{L}\},$$
$$\mathcal{T}_2(\gamma) = \inf\{\mathcal{B}(L) \mid \delta(L) \geq \gamma \text{ and } L \in \mathcal{L}\},$$

which are referred to as the *state trellis complexity* and the *branch trellis complexity* functions, respectively, [5].

Forney conjectured that $\mathcal{S}(L_n)$ grows at least exponentially in γ_n, where L_n is the densest (rational) lattice in dimension n and γ_n is the coding gain of L_n. He has also conjectured that the complexity functions grow at least exponentially in γ [4].

Here an explicit lower bound relating the trellis complexity of a lattice L, its gain, and its dimension is derived. This inequality will imply Forney's conjecture for any chain of densest lattices in $\mathbb{R}^n$, as well as any chain of lattices with relatively

* Received by the editors March 20, 1995; accepted for publication (in revised form) November 13, 1995. This research was supported in part by the National Sciences and Engineering Research Council of Canada grant A7382.

† Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.

high effective coding gain per dimension, a parameter that will be defined. It should be clear that a proof of the first conjecture gives supporting ground for the second conjecture.

2. Preliminaries. Using the notation of Forney [3], let the normalized density length profile (DLP) of an n-dimensional lattice L be the sequence $\mathbf{k}'(L)$ whose ith component $k_i'(L)$ is the maximum normalized log density

$$k'(L_i) = -\log V(L_i) + (i/n)\log V(L)$$

of any i-dimensional cross section L_i, $1 \leq i \leq n$, where log denotes the natural logarithm function. The normalized inverse DLP of L is defined to be the sequence $\tilde{\mathbf{k}}'(L)$ whose ith component $\tilde{k}_i'(L)$ is the minimum log density $-\log V(P_i(L)) + (i/n)\log V(L)$ of any i-dimensional projection $P_i(L)$ of L.

The following lemmas are due to Forney.

LEMMA 2.1. For any lattice L and for any $1 \leq i \leq n$,

$$-k_{n-i}'(L) = \tilde{k}_i'(L). \tag{1}$$

LEMMA 2.2. For any lattice L of dimension n and for $1 \leq i \leq n$,

$$k_i'(L) \leq (i/2)\log(\gamma_i/\delta(L)), \tag{2}$$

where γ_i is the coding gain of the densest lattice in dimension i. (γ_i is sometimes referred to as "the Hermite's constant in dimension i" in the number theory literature.)

If L is a lattice in $\mathcal{L}$, given an arbitrary trellis Γ for L, let $\mathbf{s}(\Gamma)$ denote a vector whose ith component $s_i(\Gamma)$, $0 \leq i \leq n$ is the number of states in level i of Γ. The following lemma is proved in [3].

LEMMA 2.3. For any lattice $L \in \mathcal{L}$ and any trellis diagram Γ for L,

$$\mathbf{s}(\Gamma) \geq \exp(\tilde{\mathbf{k}}'(L) - \mathbf{k}'(L)). \tag{3}$$

The following result is from [1] and is a combination of the Minkowski–Hlawka lower bound and the Kabatiansky–Levenshtein upper bound.

LEMMA 2.4. For n large, the coding gain γ_n of the densest lattice packing in $\mathbb{R}^n$ satisfies the inequality

$$n/2\pi e \leq \gamma_n \leq 1.744\, n/2\pi e. \tag{4}$$

3. The main results. The main results are presented by a sequence of lemmas.

LEMMA 3.1. Let $L \in \mathcal{L}$. Let Γ be a trellis diagram for L. For any i satisfying $1 \leq i \leq n-1$, we have

$$s_i(\Gamma) + s_{n-i}(\Gamma) \geq 2\exp(-k_i'(L))\exp(-k_{n-i}'(L)). \tag{5}$$

Proof. By Lemma 2.1, $-k_{n-i}'(L) = \tilde{k}_i'(L)$ and $-k_i'(L) = \tilde{k}_{n-i}'(L)$. Using these relations in Lemma 2.3 yields

$$s_i(\Gamma) \geq \exp(\tilde{k}_i'(L) - k_i'(L)) = \exp(-k_{n-i}'(L) - k_i'(L)), \tag{6}$$

$$s_{n-i}(\Gamma) \geq \exp(\tilde{k}_{n-i}'(L) - k_{n-i}'(L)) = \exp(-k_i'(L) - k_{n-i}'(L)), \tag{7}$$

and the result follows by adding the inequalities. □

LEMMA 3.2. Given $0 < \epsilon < 0.2$, there exists l sufficiently large such that for any $L \in \mathcal{L}$ of dimension $n > l$,

$$\mathcal{S}(L) > 2[M(L)]^{n/2} \int_{\epsilon}^{\frac{1}{2}-\epsilon} \exp\left(\frac{nH_e(x)}{2}\right) dx, \tag{8}$$

where $M(L) = 2\pi e\delta(L)/(1.744\, n)$, and

$$H_e(x) = -x \log x - (1-x)\log(1-x) \tag{9}$$

is the natural binary entropy function.

Proof. Given $0 < \epsilon < 0.2$, we choose m such that (4) holds for all $n \geq m$. Let $n \geq m/\epsilon$ and $j = \lceil \epsilon n \rceil$. It follows immediately from (5) that

$$\sum_{i=1}^{n} s_j(\Gamma) \geq \sum_{i=j}^{n-j} s_j(\Gamma) \geq 2 \sum_{i=j}^{\lfloor n/2 \rfloor} \exp(-k_i'(L)) \exp(-k_{n-i}'(L)), \tag{10}$$

where Γ is an arbitrary trellis of L.

It follows from (2) that

$$\sum_{i=j}^{\lfloor n/2 \rfloor} \exp(-k_i'(L)) \exp(-k_{n-i}'(L)) \geq \sum_{i=j}^{\lfloor n/2 \rfloor} (\delta(L)/\gamma_i)^{i/2} (\delta(L)/\gamma_{n-i})^{(n-i)/2}, \tag{11}$$

where γ_i is the maximum density of lattice packings in dimension i. By (4) and the assumption on m, it follows that $\gamma_i \leq i(1.744/2\pi e)$ for all $j \leq i \leq (n-j)$, and hence the right side of (11) is

$$\geq \sum_{i=j}^{\lfloor n/2 \rfloor} (2\pi e\delta(L)/1.744\, i)^{i/2} (2\pi e\delta(L)/1.744\, (n-i))^{(n-i)/2}. \tag{12}$$

For any lattice L, define $M(L) = 2\pi e\delta(L)/(1.744\, \dim L)$. Thus $M(L)$ is a measure of comparison between the coding gain of L with that given by the upper bound in (4) and is called *the effective gain per dimension* of L throughout this work. It now follows from the above that

$$\sum_{i=j}^{\lfloor n/2 \rfloor} \exp(-k_i'(L)) \exp(-k_{n-i}'(L)) \geq [M(L)]^{n/2} \sum_{i=j}^{\lfloor n/2 \rfloor} \exp((n/2) H_e(i/n)). \tag{13}$$

Clearly $\exp(nH_e(x)/2)$ is an increasing positive function on $(0, 0.5)$. Thus the area given by the curve $\exp(nH_e(x)/2)$, lines $y = \epsilon$, $y = \lfloor n/2 \rfloor / n$, and the x-axis is less than that given by the left side of the inequality below; that is,

$$(1/n) \sum_{i=j}^{\lfloor n/2 \rfloor} \exp((n/2) H_e(i/n)) > \int_{\epsilon}^{\frac{\lfloor n/2 \rfloor}{n}} \exp\left(\frac{nH_e(x)}{2}\right) dx.$$

Since $\lfloor n/2 \rfloor / n \geq 0.5 - 1/2n \geq 0.5 - \epsilon$, it follows from the above inequality that

$$(1/n) \sum_{i=j}^{\lfloor n/2 \rfloor} \exp(-k_i'(L) - k_{n-i}'(L)) \geq [M(L)]^{n/2} \int_{\epsilon}^{\frac{1}{2}-\epsilon} \exp\left(\frac{nH_e(x)}{2}\right) dx. \tag{14}$$

By (10) and (14),

$$\frac{s_1(\Gamma)+s_2(\Gamma)+\cdots+s_n(\Gamma)}{n} \geq 2[M(L)]^{n/2}\int_{\epsilon}^{\frac{1}{2}-\epsilon}\exp\left(\frac{nH_e(x)}{2}\right)dx, \tag{15}$$

where $H_e(x)$ is given in (9). By taking the minimum of the left side over all the possible trellis diagrams of L, it follows that

$$\mathcal{S}(L) \geq 2[M(L)]^{n/2}\int_{\epsilon}^{\frac{1}{2}-\epsilon}\exp\left(\frac{nH_e(x)}{2}\right)dx, \tag{16}$$

which gives the result.

COROLLARY 3.3. If $\{L_n,\ n=1,2,\ldots\}$ is a chain of rational lattices such that $\dim L_n = n$ and $\liminf_n \{M(L_n)\} > 0.5$, then $\mathcal{S}(L_n)$ grows exponentially as a function of $\delta(L_n)$.

Proof. Let $\beta = \liminf_n \{M(L_n)\}$, and choose θ and ν such that $0.5 < \theta < \nu < \beta$. By the definition of lim inf, then there exists m such that $n \geq m$ implies that $M(L_n) \geq \nu$.

It is known that the entropy function $H_e(x)$ is one-to-one and thus invertible on $(0, 0.5)$. Let $\zeta = \max(H_e^{-1}(-\log\theta), 0.3)$ and $\epsilon = (0.5-\zeta)/2$; then $0 < \epsilon < 0.1$.

By the previous theorem, there exists $l \geq m$ sufficiently large such that for $n > l$, the inequality (8) holds with L_n substituted for L. Since the function $\exp(nH_e(x)/2)$ is increasing in the interval $(0, 0.5)$, it follows that for $0.5 - 2\epsilon < x < 0.5 - \epsilon$,

$$\exp\left(\frac{nH_e(x)}{2}\right) \geq \exp\left(\frac{nH_e(0.5-2\epsilon)}{2}\right) \geq \exp\left(\frac{nH_e(\zeta)}{2}\right) \geq \theta^{-n/2}. \tag{17}$$

Since $0 < \epsilon < 0.1$,

$$\begin{aligned} 2[M(L_n)]^{n/2}\int_{\epsilon}^{\frac{1}{2}-\epsilon}\exp\left(\frac{nH_e(x)}{2}\right)dx &\geq 2[M(L_n)]^{n/2}\int_{0.5-2\epsilon}^{0.5-\epsilon}\exp\left(\frac{nH_e(x)}{2}\right)dx \\ &\geq 2[M(L_n)]^{n/2}\epsilon\theta^{-n/2} \geq 2\epsilon(\nu/\theta)^{n/2} \end{aligned}$$

for all $n \geq l$. By (4) and (8), since $\delta(L_n) \leq \gamma_n$, the above implies that

$$\mathcal{S}(L_n) \geq (\nu/\theta)^{n/2} \geq (\nu/\theta)^{\pi e\delta(L_n)/1.744}. \tag{18}$$

Since $\nu/\theta > 1$, the result follows. □

COROLLARY 3.4. For $n = 1, 2, 3, \ldots$, let L_n denote a rational lattice achieving the maximum center density of lattice packings in dimension n. Let γ_n denote the coding gain of L_n. Then $S(L_n)$ grows at least exponentially as a function of γ_n.

Remark. For an arbitrary dimension n, it is known that there exist rational lattices achieving the maximum possible center density of lattice packings in dimension n [6].

Proof. By (4), it follows that $M(L_n) \geq 1/1.744 > 0.573$ for n sufficiently large and thus $\liminf_n\{M(L_n)\} \geq 0.573$. Corollary 3.3 now gives the result.

4. Comments. Since $\mathcal{B}(L) \geq \mathcal{S}(L)$ for an arbitrary lattice, it follows that Corollary 3.3 is valid with $\mathcal{B}$ replacing $\mathcal{S}$. It may seem from (8) that in terms of the complexity of achieving a certain gain, lattices with lower effective gain per dimension are more promising. However, Tarokh [5] proves that even lattices of low effective gain

per dimension are not good when considering the best trade-off between the coding gain and trellis decoding complexity.

Acknowledgments. The authors thank G. D. Forney, B. M. Hochwald, and C. L. Stewart for valuable comments and suggestions.

REFERENCES

[1] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, Springer-Verlag, Berlin, New York, 1993.

[2] G. D. Forney, *Coset codes–part* II*: Binary lattices and related codes*, IEEE Trans. Inform. Theory, 34 (1988), pp. 1152–1187.

[3] ———, *Density/length profiles and trellis complexity of lattices*, IEEE Trans. Inform. Theory, 40 (1995), pp. 1753–1772.

[4] ———, Private communication, 1995.

[5] V. Tarokh, *Trellis Complexity Versus The Coding Gain of Lattice-Based Communication Systems*, Ph.D. thesis, The University of Waterloo, Waterloo, Ontario, Canada, 1995.

[6] M. A. Tsfasman and S. G. Vladut, *Algebraic-Geometric Codes*, Kluwer Academic Publishers, Norwell, MA, 1991.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 602–624, November 1996

007

THE GRAPHICAL ASYMMETRIC TRAVELING SALESMAN POLYHEDRON: SYMMETRIC INEQUALITIES *

SUNIL CHOPRA† AND GIOVANNI RINALDI ‡

Abstract. A present trend in the study of the symmetric traveling salesman polytope is to use, as a relaxation of the polytope, the graphical traveling salesman polyhedron (GTSP). Following a parallel approach for the asymmetric traveling salesman polytope, we define the graphical asymmetric traveling salesman problem on a general digraph D and its associated polyhedron GATSP(D). We give basic polyhedral results and lifting theorems for GATSP(D) and we give a general condition for a facet-defining inequality for GTSP to yield a symmetric facet-defining inequality for GATSP. Using this approach we show that all known major families of facet-defining inequalities of GTSP define facets of GATSP. Finally, we discuss possible extension of these results to the asymmetric traveling salesman polytope.

Key words. traveling salesman problem, directed graph, polyhedron, facet, linear inequality

AMS subject classifications. 05C35, 68E10

1. Introduction.

Let $D = (V, A)$ be a directed graph. We denote by $a = (u, v)$ the arc of A having tail u and head v. For every nonempty proper subset W of V, we denote by $\delta_D^+(W)$ and $\delta_D^-(W)$ the sets of arcs defined by $\delta_D^+(W) = \{(u, v) \mid u \in W, v \notin W\}$ and $\delta_D^-(W) = \{(u, v) \mid v \in W, u \notin W\}$, respectively. When W is a singleton node $\{w\}$, we simply write $\delta_D^+(w)$ and $\delta_D^-(w)$. Let $\mathbb{R}^A$ be the space of all real vectors whose components are indexed by A. For x in $\mathbb{R}^A$ we denote by x_a (or $x(u, v)$) the component of x indexed by $a = (u, v)$; for $J \subseteq A$ we denote by $x(J)$ the sum $\sum_{a \in J} x_a$.

A *family of arcs* of D is a collection F of elements of A. Several copies of the same element of A may appear in the collection. For every element a of A, we call *multiplicity* of a in F the number of times a appears in F. A set of arcs of A is a family where every element has multiplicity 1. With every family F of arcs of D we associate a unique incidence vector $\chi^F \in \mathbb{R}^A$ by setting χ_a^F equal to the multiplicity of a in F for every $a \in A$. Let F_1 and F_2 be two families of arcs of D. We denote by $F_1 + F_2$, $F_1 - F_2$, and kF_1 the families having incidence vectors $\chi^{F_1} + \chi^{F_2}$, $\chi^{F_1} - \chi^{F_2}$, and $k\chi^{F_1}$, respectively.

Let F be a family of arcs of D. By $D^1[F]$ we denote the support digraph of F, i.e., the subgraph obtained from D by removing the arcs not contained in F. By $D[F]$ we denote the directed multigraph obtained by replicating every arc a as many times as the multiplicity of a in F.

Let $G = (V, W)$ be an undirected graph. We denote by $e = [u, v]$ the edge of E with endpoints u and v, and for $x \in \mathbb{R}^E$ we denote by x_e (or $x[u, v]$) the component of x indexed by e. For every nonempty proper subset W of V, we denote by $\delta_G(W)$ the set of edges defined by $\delta_G(W) = \{[u, v] \mid u \in W, v \notin W\}$, and if W is a singleton node $\{w\}$ we simply write $\delta_G(w)$. All the above definitions and notation on the families of

*Received by the editors November 20, 1991; accepted for publication (in revised form) November 13, 1995.

†J. L. Kellogg Graduate School of Management, Northwestern University, 2001 Sheridan, Evanston, IL 60208-2009 (schopra@casbah.acns.nwu.edu).

‡Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche, viale Manzoni 30, 00185 Rome, Italy (rinaldi@iasi.rm.cnr.it).

arcs apply also to the *families of edges* when the directed graph D and its arc set A are replaced with the undirected graph G and its edge set E, respectively.

A *tour* of D is a family T of arcs of D such that

(a) $|\delta^+_{D[T]}(u)| = |\delta^-_{D[T]}(u)|$ for all u in V;

(b) $D[T]$ is connected.

An *undirected tour* of G is a family Θ of edges of G such that

(a′) $|\delta_{G[\Theta]}(u)|$ is even for all u in V;

(b′) $G[\Theta]$ is connected.

In the following we use a roman or a Greek letter to denote a tour or an undirected tour, respectively. For the sake of simplicity we use the term "tour" also in the undirected case, every time this does not generate any ambiguity.

A tour T is *minimal* if no proper subfamily of T is a tour, i.e., if the removal of any directed cycle from $D[T]$ produces a disconnected multigraph. A tour of D is *extremal* if its incidence vector is not a convex combination of other tours of D. An extremal tour is minimal, but the converse is not necessarily true (see §2). An *equality tour* T for an inequality $cx \geq c_0$ of $\mathbb{R}^A$ is a tour with $c\chi^T = c_0$. The sets of all tours and of all extremal tours of D are denoted by $\mathcal{T}^*_D$ and $\mathcal{T}_D$, respectively.

The definitions and notations of the previous paragraph apply, mutatis mutandis, to the undirected tours.

Given arc weights $w_a \in \mathbb{R}$, the *graphical asymmetric traveling salesman problem* (*gatsp*) is to find a minimum weight tour in $\mathcal{T}^*_D$. The *graphical asymmetric traveling salesman polyhedron* associated with D (GATSP(D)) is the convex hull of the incidence vectors of the elements of $\mathcal{T}^*_D$; i.e.,

$$\mathrm{GATSP}(D) = \mathrm{conv}\,\{\chi^T \mid T \in \mathcal{T}^*_D\}\,.$$

A *directed Hamiltonian cycle* of D is a tour H with $|\delta^+_{D[H]}(u)| = 1$ for every u in V. Given a complete directed graph D_n, the *asymmetric traveling salesman problem* (*atsp*) is to find a minimum weight Hamiltonian cycle in D_n. We denote by $\mathcal{H}_n$ the set of all directed Hamiltonian cycles of D_n. The *asymmetric traveling salesman polytope* associated with D_n (ATSP(n)) is the convex hull of the incidence vectors of the elements of $\mathcal{H}_n$; i.e.,

$$\mathrm{ATSP}(n) = \mathrm{conv}\,\{\chi^H \mid H \in \mathcal{H}_n\}\,.$$

For a directed graph G, with edge weights $w_e \in \mathbb{R}$, the *graphical traveling salesman problem* (*gtsp*) and the *graphical traveling salesman polyhedron* associated with G (GTSP(G)) are defined similarly.

We refer to [3] and [4] for the necessary background in polyhedral theory and for a survey on the polyhedral aspects of the *traveling salesman problem*.

The counterpart of *gatsp* for undirected graphs (*gtsp*) has been studied in [1], [2], [6], and [7]. To the best of our knowledge *gatsp* has not been investigated yet. The advantages in studying *gatsp* as a relaxation to *atsp* are the same as those mentioned in [1] and [2] for *gtsp*. In particular,

(a) if the objective function coefficients satisfy the triangular inequality (like in most of the real-world applications), *gatsp* has a directed Hamiltonian cycle among the optimal solutions and so *atsp* can be solved as a *gatsp*;

(b) in some applications the requirement that a tour visit a node only once may not be necessary and so *gatsp* may be a more appropriate formulation for the problem;

(c) *gatsp* is defined for general (even not Hamiltonian) graphs, and for these graphs it is not necessary to add artificial arcs, as for the case when to formulate the problem *atsp* is used instead;

(d) several facets of GATSP are defined for very sparse graphs and the coefficients of the missing edges can be computed by a sequential lifting procedure; often these coefficients can be given in closed form (see Remark 3.1);

(e) it is often possible to extend results on the polyhedral structure of GATSP to obtain results on ATSP, like has been done for the undirected case (see [8] and [5]);

(f) several results on the polyhedral structure of GTSP can be extended to GATSP (see §3). An unnecessary duplication of work can be saved in this way in the proofs.

In §2 we give general results on GATSP, we define its basic facet-defining inequalities and give some general lifting theorems. In §3 we give conditions under which a facet-defining inequality for GTSP yields a facet-defining inequality for GATSP. We use these conditions to show that several families of symmetric inequalities define facets of GATSP. In §4 we discuss possible extensions of the results of §3 to ATSP and we show some pathological cases of facet-defining inequalities for GATSP that do not define facets of ATSP.

2. Basic properties of GATSP(D). It is straightforward to see that GATSP(D) is nonempty if and only if D is strongly connected.

For every tour T of D, $\chi^T(\delta^+_{D[T]}(u)) - \chi^T(\delta^-_{D[T]}(u)) = 0$ for all u in V, and $\chi^T(\delta^+_{D[T]}(W)) + \chi^T(\delta^-_{D[T]}(W)) \geq 2$ for every nonempty proper subset W of V. It follows that GATSP$(D) \subseteq \{x \in \mathbb{R}^A \mid F_D x = 0\}$, where F_D is the node-arc incidence matrix of D. Hence GATSP(D) is not a full-dimensional polyhedron.

THEOREM 2.1. *If D is strongly connected, the dimension of* GATSP(D) *is* $|A| - |V| + 1$.

Proof. Assume that GATSP(D) is nonempty and that the incidence vectors of all tours of D satisfy $\pi x = \pi_0$. Let T be a minimal tour of D and let $D^1[T]$ be its associated support subgraph of D. Let C be a directed cycle of $D^1[T]$. By minimality of T there is an arc $\overline{a} \in C$ with multiplicity 1 in T. By adding a linear combination of the equations $F_D x = 0$ to $\pi x = \pi_0$ we can obtain a new equation $\pi' x = \pi_0$, satisfied by the incidence vector of all tours of D, with $\pi'_a = 0$ for $a \in C - \{\overline{a}\}$. Consider the tour $T + C$. Its incidence vector satisfies $\pi' x = \pi_0$; thus $\pi'_{\overline{a}} = 0$. If f is a chord of C, there is a directed cycle C_f of $D^1[T]$ containing f and only edges of C and the incidence vector of $T + C_f$ satisfies $\pi' x = \pi_0$; thus $\pi'_f = 0$. Now we shrink the nodes of C into a single node and remove all loops from the resulting shrunk graph. We iteratively repeat this process to the shrunk graph until it is reduced to a single node. At this point we have produced an equation $\pi'' x = \pi_0$ as a linear combination of $\pi x = \pi_0$ and of the equations $F_D x = 0$ by imposing that $\pi''_a = 0$ for every arc a belonging to a spanning tree of $D^1[T]$, which has $\pi''_a = 0$ for every arc a in $D^1[T]$. Let f be any arc of D which is not in $D^1[T]$. There exists a directed cycle C_f containing f and only arcs of $D^1[T]$. The incidence vector of $T + C_f$ satisfies $\pi'' x = \pi_0$; thus $\pi''_f = 0$. This shows that the equation $\pi x = \pi_0$ is a linear combination of the flow equations and since the rank of F_D is $|V| - 1$, the theorem follows. □

Since GATSP is not full dimensional, two inequalities, $\pi' x \geq \pi'_0$ and $\pi'' x \geq \pi''_0$, are *equivalent*, i.e., define the same facet of the polyhedron, if and only if there exist $\xi > 0$ and $\lambda \in \mathbb{R}^{V'}$ such that $\pi' = \xi\pi'' + \lambda F'_D$ and $\pi'_0 = \xi\pi''_0$. Here V' denotes the node set $V - \{\overline{v}\}$ for some $\overline{v} \in V$, and F'_D denotes the submatrix of F_D obtained by taking the rows corresponding to V'. Consequently, recognizing when two facet-defining

inequalities for GATSP are equivalent is not as easy as in the case of a full-dimensional polyhedron. To overcome this problem we use the following definition of *standard* form for an inequality.

A facet-defining inequality for GATSP, $\pi x \geq \pi_0$ is in *standard form* if $F_D \pi = 0$. It follows that two inequalities in standard form defining facets of GATSP are equivalent if and only if one is obtained by multiplying the other by a nonzero number. For a complete digraph, any facet-defining inequality $\pi x \geq \pi_0$ has an equivalent inequality $(\pi + \lambda F'_{D_n})x \geq \pi_0$ in standard form, where the vector λ is defined by

$$\lambda_v = \frac{1}{2n} \sum_{u \neq v} (\pi(v,u) - \pi(u,v)) - \frac{1}{2n} \sum_{u \neq \overline{v}} (\pi(\overline{v},u) - \pi(u,\overline{v})) \quad \text{for } v \in V'.$$

A facet-defining inequality $\pi x \geq \pi_0$ for GATSP(D_n) is said to be *symmetric* if

$$\pi(u,v) = \pi(v,u) \quad \text{for every } (u,v) \in A. \tag{2.1}$$

Observe that a symmetric inequality is always in standard form. It follows that to verify whether or not an inequality is equivalent to a symmetric one, it is sufficient to check if an inequality in standard form, equivalent to it, satisfies (2.1). A facet-defining inequality for GATSP(D_n) is called *asymmetric* if any inequality in standard form equivalent to it does not satisfy (2.1).

In order to study the facets of a polyhedron it is interesting to know the dimension of the convex hull of its extreme points. If this dimension coincides with the dimension of the polyhedron, several facets (those that really matter in a polyhedral cutting-plane algorithm) are the convex hull of extreme points of the polyhedron. Otherwise, to describe any facet of the polyhedron, except at most one, some extreme rays have to be considered.

It is difficult to characterize the extreme points of GATSP(D) for a general digraph D. Every extreme tour has to be minimal, but the converse does not hold. See, for example, Figure 2.1 (a), where a minimal tour that can be expressed as a convex combination of the tours of the Figures 2.1 (b) and (c) is shown.

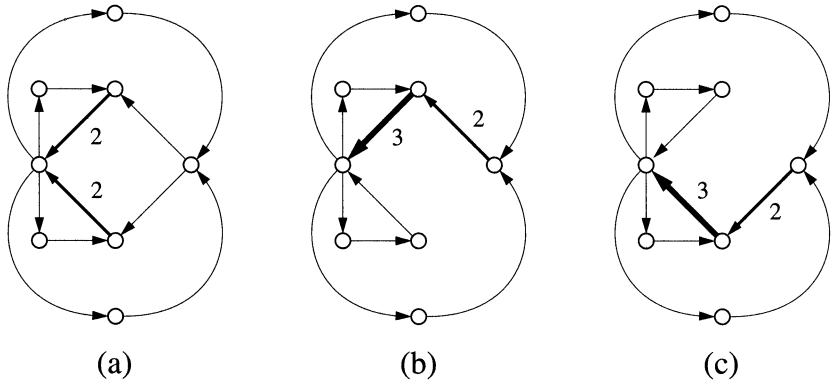


FIG. 2.1.

Also, the dimension of the convex hull of the extreme points of GATSP(D) seems hard to identify in general. As an example, consider the graphs D_a and D_b of Figure 2.2, which differ only by one arc. The dimension is zero for D_b and 2 for D_a.

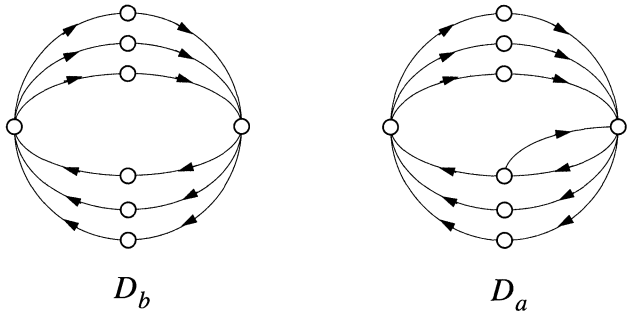


FIG. 2.2.

When D is a *bidirected* graph (that is, (u, v) in A implies (v, u) in A), the dimension depends upon the number of bridges. A *bridge* is a pair (u, v), (v, u) whose removal disconnects D. In the following we study facet-defining inequalities of GATSP(D), where D has a suitable bidirected graph as a subgraph, and so the case of a bidirected graph is of particular interest for us.

THEOREM 2.2. *If D is bidirected and has k bridges, the convex hull of the extreme points of* GATSP(D) *has dimension* $|A| - |V| - k + 1$.

Proof. The proof is similar to the one given in [1] for GTSP and is therefore omitted. □

The integer points of the polyhedron defined by

$$F_D x = 0, \tag{2.2}$$

$$x(\delta_D^+(W)) + x(\delta_D^-(W)) \geq 2 \quad \text{for all } \emptyset \neq W \subset V, \tag{2.3}$$

$$x \geq 0 \tag{2.4}$$

are the incidence vectors of all tours of D. We call (2.2) the *flow equations*, (2.3) the *cut inequalities*, and (2.4) the *trivial inequalities*. Observe that while in the directed case (2.2), (2.3), and (2.4) provide an integer linear programming formulation of *gatsp*, for the undirected case the condition that every node has an even degree in every tour cannot be stated in a similar simple way.

THEOREM 2.3. *For $a \in A$ the trivial inequality $x_a \geq 0$ is facet defining for* GATSP(D) *if and only if the subgraph of D obtained by deleting the arc a is strongly connected.*

Proof. Since we assume that GATSP(D) is not empty then D is strongly connected. If $D - \{a\}$ is not strongly connected then $x_a \geq 1$ is valid for every tour of D, and so $x_a \geq 0$ is not even supporting for GATSP(D). If $D - \{a\}$ is strongly connected then by Theorem 2.1 there are $|A| - |V| + 1$ affinely independent tours of the digraph $D - \{a\}$ which are also tours of D satisfying $x_a = 0$. Consequently, $x_a \geq 0$ is facet defining for GATSP(D). □

THEOREM 2.4. *For $\emptyset \neq W \subset V$, the cut inequality $x(\delta_D^+(W)) + x(\delta_D^-(W)) \geq 2$ is facet defining for* GATSP(D) *if and only if both the subgraphs of D induced by W and by $V - W$, respectively, are strongly connected.*

Proof. Since we assume that GATSP(D) is not empty, D is strongly connected and both the arc sets $x(\delta_D^+(W))$ and $x(\delta_D^-(W))$ are not empty. Assume that the subgraph of D induced by W is not strongly connected. Then $\delta_D^+(W') \subset \delta_D^+(W)$ for some $\emptyset \neq W' \subset W$, and so the inequality

$$x(\delta_D^+(W)) + x(\delta_D^-(W)) \geq 2 \tag{2.5}$$

is dominated by $x(\delta_D^+(W')) + x(\delta_D^-(W)) \geq 2$ and cannot be facet defining. Now assume that both the subgraphs of D induced by W and $V - W$, respectively, are strongly connected. Let A^W and $A^{\overline{W}}$ be the arc sets of the subgraphs of D induced by W and $V - W$, respectively. By hypothesis these two subgraphs have two sets $\mathcal{T}^W$ and $\mathcal{T}^{\overline{W}}$, respectively, of affinely independent tours with $|\mathcal{T}^W| = |A^W| - |W| + 2$ and $|\mathcal{T}^{\overline{W}}| = |A^{\overline{W}}| - |V - W| + 2$. Let $a^+ \in \delta^+(W)$, $a^- \in \delta^-(W)$, $T^W \in \mathcal{T}^W$, and $T^{\overline{W}} \in \mathcal{T}^{\overline{W}}$. For $a \in \delta^+(W)$ denote by C_a^+ a directed cycle containing the arcs a, a^-, and only arcs in the sets A^W and $A^{\overline{W}}$. Analogously, for $a \in \delta^-(W)$ denote by C_a^- a directed cycle containing the arcs a, a^+, and only arcs in the sets A^W and $A^{\overline{W}}$. The set

$$\begin{aligned}
&\left\{T^W + T^{\overline{W}} + C_a^+ \mid a \in \delta^+(W)\right\} \\
&\cup \left\{T^W + T^{\overline{W}} + C_a^- \mid a \in \delta^-(W) - \{a^-\}\right\} \\
&\cup \left\{T^W + C_{a^-}^- + T \mid T \in \mathcal{T}^{\overline{W}}\right\} \\
&\cup \left\{T^{\overline{W}} + C_{a^-}^- + T \mid T \in \mathcal{T}^W\right\}
\end{aligned}$$

contains $|A| - |V| + 1$ affinely independent tours of D satisfying (2.5) with equality. The result thus follows. □

An inequality $\pi x \geq \pi_0$ in $\mathbb{R}^A$ satisfies the *shortest path condition* if for every (u, v) in A and for every directed path P_{uv} connecting u to v, $\pi(u, v) \leq \pi(P_{uv})$.

PROPOSITION 2.5. *A facet-defining inequality for* GATSP(D) *either is a trivial inequality or satisfies the shortest path condition.*

Proof. Let $\pi x \geq \pi_0$ be a facet-defining inequality for GATSP(D) that does not satisfy the shortest path condition. Therefore, there exists (u, v) such that $\pi(u, v) > \pi(P_{uv})$. No equality tour for the inequality can contain (u, v): if there was such a tour T then the tour $T - (u, v) + P_{uv}$ would violate the inequality. Consequently all equality tours satisfy the trivial inequality $x(u, v) \geq 0$ with equality, and so $\pi x \geq \pi_0$ is equivalent to $x(u, v) \geq 0$. □

PROPOSITION 2.6. *Let $\pi x \geq \pi_0$ be a valid inequality for* GATSP(D). *Then $\pi(C) \geq 0$ for every directed cycle C of D.*

Proof. Let T be a tour of D. If C is a directed cycle with $\pi(C) < 0$ then the tour $T + kC$, for k sufficiently large, violates $\pi x \geq \pi_0$. □

The next two lemmas give liftings for facets of GATSP(D).

LEMMA 2.7. *Let $\pi^* x^* \geq \pi_0$ be a valid inequality for* GATSP(D^*) *such that D^* has a directed cycle C with a chord $\overline{a} = (u, v)$ and $\pi^*(C) = 0$. Form D from D^* by deleting $\overline{a}$. The inequality $\pi^* x^* \geq \pi_0$ is facet defining for* GATSP(D^*) *if it satisfies the shortest path property and $\pi x \geq \pi_0$ is facet defining for* GATSP(D), *where π is the restriction of π^* to the arcs of D.*

Proof. Let P_{uv} and P_{vu} be the directed paths contained in the directed cycle C and connecting u to v and v to u, respectively. The validity of $\pi^* x^* \geq \pi_0$ and Proposition 2.6 imply $\pi^*((u, v) + P_{vu}) \geq 0$. The shortest path property of $\pi^* x^* \geq \pi_0$ and the assumption that $\pi^*(P_{uv} + P_{vu}) = 0$ imply $\pi^*((u, v) + P_{vu}) \leq 0$, and so we have $\pi^*((u, v) + P_{vu}) = 0$. Since $\pi x \geq \pi_0$ is facet defining for GATSP(D), there exists a set $\mathcal{T}$ of $|A| - |V| + 1$ affinely independent tours of D that satisfy $\pi x \geq \pi_0$ with equality. Let T be a tour of $\mathcal{T}$. The tours of $\mathcal{T}$ with the tour $T + P_{vu} + (u, v)$ are a set of $|A| - |V| + 2$ affinely independent tours of D^* that satisfy $\pi^* x^* \geq \pi_0$ with equality. □

LEMMA 2.8. *Let $\pi^* x^* \geq \pi_0$ be a valid inequality for* GATSP(D^*) *such that $D^* = (V^*, A^*)$ has a directed chordless cycle C with $\pi_a^* = 0$ for $a \in C$. Let D_C^* be the graph*

obtained from D^* by contracting all arcs in C and deleting all resulting parallel arcs. The inequality $\pi^* x^* \geq \pi_0$ is facet defining for GATSP(D^*) *if it satisfies the shortest path property and $\pi x \geq \pi_0$ is facet defining for* GATSP(D^*_C), *where π is the restriction of π^* to the arcs of D^*_C.*

Proof. Denote by V^C the node set of the cycle C. Given two arcs $a_1 = (w, v_1)$ and $a_2 = (w, v_2)$ (or $a_1 = (v_1, w)$ and $a_2 = (v_2, w)$), where $w \notin V^C$, we say that they are *relative* if both v_1 and v_2 belong to V^C. Observe that two relative arcs of A^* correspond to two parallel arcs of the graph obtained from D^* by contracting all arcs of C. Let A' be a maximal subset of A^* that does not contain relative arcs. There is a one-to-one correspondence between the arcs of the graph D^*_C and the arcs in A'.

Consider any valid inequality $\gamma^* x^* \geq \gamma_0$ for GATSP(D^*) such that

$$\{x \in \text{GATSP}(D^*) \mid \pi^* x^* = \pi_0\} \subseteq \{x^* \mid \gamma^* x^* = \gamma_0\}.$$

Let $\overline{a}$ be any arc of C and S be a spanning tree of D^* with arcs in A' which contains all the arcs of C except $\overline{a}$. Declare a node of V^* as the root of S. Traverse S in a breadth-first search (BFS) manner and use the flow equations $F_{D^*} = 0$ for every node u_i reached in the search to set $\gamma^*(u_i, u_j) = \pi^*(u_i, u_j)$, where u_i and u_j are in the parent-child relation in BFS and (u_i, u_j) is an arc of S.

We say that a tour T^* of D^* is obtained by *extending* a tour T of D^*_C if it is obtained by taking the arcs of A' that correspond to the arcs of T and by adding the arcs of C that are necessary to complete a tour. Let T^* be a tour obtained by extending an equality tour for $\pi x \geq \pi_0$. The tour T^* is an equality tour for $\pi^* x^* \geq \pi_0$ and so is the tour $T^* + C$. It follows that $\gamma^*(T^*) = \gamma^*(T^* + C)$ and $\gamma^*_{\overline{a}} = 0$.

Since $\pi x \geq \pi_0$ is facet defining for GATSP(D^*_C), the set of equality tours for $\pi^* x^* \geq \pi_0$ obtained by extending all the equality tours for $\pi x \geq \pi_0$ is sufficient to define the coefficients γ^*_a for $a \in A'$ up to a common positive multiple.

Let (w, v_1) be an arc of $A^* - A'$ and (w, v_2) be its relative arc in A'. For u and v in V^C denote by $C_{u,v}$ the directed path having only arcs in C that connects u to v. From the shortest path property of $\pi^* x^* \geq \pi_0$ it follows that $\pi^*((w, v_1) + C_{v_1 v_2}) \geq \pi^*(w, v_2)$ and $\pi^*((w, v_2) + C_{v_2 v_1}) \geq \pi^*(w, v_1)$, and so $\pi^*(w, v_1) = \pi^*(w, v_2)$. Let T^* be an equality tour for $\pi^* x^* \geq \pi_0$ containing (w, v_2) and only arcs in A'. The tour $T^* - (w, v_2) + (w, v_1) + C_{v_1 v_2}$ satisfies also $\pi^* x^* \geq \pi_0$ with equality, and so $\gamma^*(w, v_1) = \gamma^*(w, v_2)$. One proceeds analogously for an arc (v_1, w) in $A^* - A'$.

Since the vector γ^* is a multiple of π^*, the inequality $\pi^* x^* \geq \pi_0$ defines a facet of GATSP(D^*). □

Note that every facet-defining inequality $\pi x \geq \pi_0$ with $\pi(C) = 0$ can be converted to an equivalent inequality $\pi' x \geq \pi_0$ (by adding to it a suitable linear combination of the flow equations) such that $\pi'_a = 0$ for all a in C.

Let $\pi x \geq \pi_0$ be a facet-defining inequality for GATSP(D), with $D = (V, A)$. Let $D' = (V', A')$ be a strongly connected digraph. Let $D^* = (V^*, A^*)$ be the digraph defined by

$$\begin{aligned} V^* &= V - \{\overline{v}\} + V', \\ A^* &= A - \delta^+_D(\overline{v}) - \delta^-_D(\overline{v}) + A' + \left\{(v, u) \mid (v, \overline{v}) \in A, u \in V^-_v \subseteq V'\right\} \\ &\quad + \left\{(u, v) \mid (\overline{v}, v) \in A, u \in V^+_v \subseteq V'\right\}, \end{aligned}$$

where $\overline{v}$ is a node of V and V^-_v and V^+_v are arbitrary nonempty subsets of V' associated

with v for $v \in V - \{\overline{v}\}$. The inequality $\pi^* x^* \geq \pi_0$ defined by

$$\pi^*(u,v) = \begin{cases} \pi(u,v) & \text{for } (u,v) \in A - \delta_D^+(\overline{v}) - \delta_D^-(\overline{v}), \\ 0 & \text{for } (u,v) \in A', \\ \pi(u,\overline{v}) & \text{for } (u,v) \in \delta_{D^*}^-(V'), \\ \pi(\overline{v},v) & \text{for } (u,v) \in \delta_{D^*}^+(V') \end{cases}$$

is said to be obtained from $\pi x \geq \pi_0$ by *zero lifting* of the node $\overline{v}$.

THEOREM 2.9. *If $\pi^* x^* \geq \pi_0$ is obtained by zero lifting of a facet-defining inequality $\pi x \geq \pi_0$ for* GATSP(D), *then $\pi^* x^* \geq \pi_0$ is facet defining for* GATSP(D^*).

Proof. Every strongly connected digraph can be contracted to a single node by recursively deleting all the chords of a directed cycle and then contracting all edges of the cycle. Consequently, the theorem follows from Lemmas 2.7 and 2.8. □

Due to Theorem 2.9, in the following we will restrict ourselves only to *simple* inequalities, i.e., to inequalities that cannot be obtained by zero lifting of other inequalities. Therefore, we will derive facet-defining inequalities for GATSP from simple facet-defining inequalities for GTSP.

3. Facets of GATSP from those of GTSP. Consider an undirected complete graph with n nodes $K_n = (V_n, E_n)$ and the associated polyhedron GTSP(K_n). Cornuéjols, Fonlupt, and Naddef [1] and Naddef and Rinaldi [6], [7] have studied this polyhedron and given several families of facet-defining inequalities for it.

Assume that the inequality

$$\mu y \geq \mu_0 \tag{3.1}$$

is facet defining for GTSP(K_n). A subgraph $G = (V_n, E)$ of K_n is said to be a *skeleton* for (3.1) if the restriction of (3.1) to $\mathbb{R}^E$ defines a facet of GTSP(G). The inequality (3.1) is said to be *stable for the skeleton* G if for every edge $e \in E_n - E$ there exists an equality tour Θ_e for (3.1) containing e and only edges in E. This is equivalent to saying that the sequential lifting coefficients of the edges in $E_n - E$ do not depend on the lifting sequence. (For a definition of sequential lifting see, e.g., [3].) From this definition it follows that if (3.1) is stable for the skeleton G, then for every graph $G' = (V_n, E')$ with $E \subseteq E'$, the restriction of (3.1) to $\mathbb{R}^{E'}$ defines a facet of GTSP(G'). The definitions of *skeleton* and of *inequality stable for a skeleton* apply, mutatis mutandis, also to the case of a directed graph.

Remark 3.1. All the facet-defining inequalities for GTSP(K_n) described in [1], [6], and [7] have the property that they are stable for a skeleton which is quite sparse. Using this nice property, it is possible to simplify the proofs that these inequalities define facets of the polyhedron. For the inequalities described in this section this property holds as well.

Let $G = (V_n, E)$ be a skeleton of (3.1) and let $D = (V_n, A)$ be the directed graph obtained from G by replacing each edge $e = [u_i, u_j]$ by the pair of arcs (u_i, u_j) and (u_j, u_i). Here we give a set of sufficient conditions under which a facet-defining inequality for GTSP(G) gives a facet-defining inequality with symmetric coefficients for GATSP(D).

Define the inequality

$$\pi x \geq \pi_0, \tag{3.2}$$

with $\pi(u_i, u_j) = \pi(u_j, u_i) = \mu_e$, where $e = [u_i, u_j]$ and $\pi_0 = \mu_0$. For any tour T of D_n one can form an undirected tour Θ of K_n where the multiplicity of any edge

$e = [u_i, u_j]$ in Θ is the sum of the multiplicity of the arcs (u_i, u_j) and (u_j, u_i) in T. If T violates (3.2) then Θ violates (3.1). Thus the inequality (3.2) is valid for GATSP(D_n).

Let S be any spanning tree of G with edges $E(S)$. Consider the set of arcs $S_D = \{(u_i, u_j), (u_j, u_i) \mid [u_i, u_j] \in E(S)\}$.

THEOREM 3.1. *If the equality tours for the inequality* (3.1) *satisfy the condition* (3.3) *stated below, then D is a skeleton for the inequality* (3.2). *Moreover, if* (3.1) *is stable for G so is* (3.2) *for D.*

(3.3) *There exists a spanning tree S of $G = (V_n, E)$ and an ordering of the edges in $E - E(S)$, say $\langle e_1, e_2, \ldots, e_t \rangle$ where $t = |E - E(S)|$, such that for each edge e_i there exists an equality tour $\Theta(e_i)$ for* (3.1) *with edges in E and multiplicity of e_i equal to one and a cycle $C(e_i)$ in $\Theta(e_i)$ that contains e_i with $C(e_i) \subseteq E(S) \cup \{e_1, e_2, \ldots, e_i\}$.*

Proof. Let π_A be the restriction of π to $\mathbb{R}^A$. Consider any valid inequality $\gamma x \geq \gamma_0$ for GATSP(D) such that $\{x \in \text{GATSP}(D) \mid \pi_A x = \pi_0\} \subseteq \{x \mid \gamma x = \gamma_0\}$.

In the same way as in the proof of Lemma 2.8 we can use the flow equations $F_D x = 0$ to assume that

$$\gamma(u_i, u_j) = \gamma(u_j, u_i) \quad \text{for all } [u_i, u_j] \in S, \tag{3.4}$$

where S is a spanning tree of G. Any tour Θ is an Eulerian graph and so can be converted to a directed tour T by traveling the edges of Θ in some direction. If Θ is an equality tour for (3.1) with edges in E, then T must be an equality tour for (3.2). Thus all equality tours for (3.1) with edges in E can be converted to equality tours for (3.2).

First we show that if the coefficients in γ are symmetric, i.e.,

$$\gamma(u_i, u_j) = \gamma(u_j, u_i) \quad \text{for all } [u_i, u_j] \in E, \tag{3.5}$$

then the inequality (3.2) is facet defining for GATSP(D). The restriction of (3.1) to $\mathbb{R}^E$ is facet defining for GTSP(G); therefore we have a sufficient number of independent equality solutions to uniquely define the coefficients of μ. Each of these tours when directed satisfies (3.2) at equality and the resulting incidence vectors are still independent. Let B be the incidence matrix of these equality solutions. We have

$$B\gamma^T = \overline{\gamma}_0, \tag{3.6}$$

where $\overline{\gamma}_0$ is a vector with each entry equaling γ_0. If the condition (3.5) holds, then (3.5) and (3.6) uniquely define γ to be a multiple of π_A, and so (3.2) defines a facet of GATSP(D).

Given the ordering of the edges in $E - E(S)$, assume that $e_r = [u_{i(r)}, u_{j(r)}]$. First consider the case $r = 1$. By (3.3) there exists an equality tour $\Theta(e_1)$ for (3.1) with edges in E and multiplicity of e_1 equal to one. One can direct the edges in $\Theta(e_1)$ to obtain the directed tour $T(e_1)$. Without loss of generality assume that $(u_{i(1)}, u_{j(1)}) \in T(e_1)$. From (3.3) there is a cycle $C(e_1)$ in $\Theta(e_1)$ such that $e_1 \in C(e_1)$ and $C(e_1) \subseteq E(S) + \{e_1\}$. Let $C_D(e_1)$ be the corresponding directed cycle in the directed tour $T(e_1)$. Let $\overline{C}_D(e_1)$ be the directed cycle obtained by reversing every arc in $C_D(e_1)$; i.e., $(u_i, u_j) \in \overline{C}_D(e_1)$ if and only if $(u_j, u_i) \in C_D(e_1)$. Consider the tour

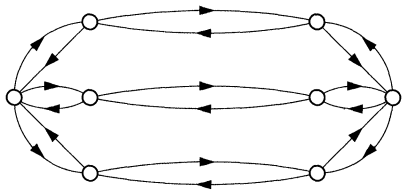


FIG. 3.1.

$T(e_1) = T(e_1) + \overline{C}_D - C_D$. Since incidence vectors of both $T(e_1)$ and $\overline{T}(e_1)$ satisfy (3.2) at equality, we have

$$\sum_{a \in C_D(e_1)} \gamma_a = \sum_{a \in \overline{C}_D(e_1)} \gamma_a.$$

Using (3.4) and the facts that $C_D(e_1) \subseteq S_D \cup \{(u_{i(1)}, u_{j(1)})\}$ and $\overline{C}_D(e_1) \subseteq S_D \cup \{(u_{j(1)}, u_{i(1)})\}$, we have

$$\gamma(u_{i(1)}, u_{j(1)}) = \gamma(u_{j(1)}, u_{i(1)}) = \gamma_{e_1}.$$

Continuing in the same manner for $r = 2, 3, \ldots, t$ we obtain

$$\gamma(u_{i(r)}, u_{j(r)}) = \gamma(u_{j(r)}, u_{i(r)}) = \gamma_{e_r} \quad \text{for } 1 \leq r \leq t.$$

This proves that (3.5) holds and that D is a skeleton for (3.2).

Let (u_i, u_j) be an arc not contained in A. This implies that the edge $[u_i, u_j]$ is not contained in E. If (3.1) is stable for G there exists an equality tour Θ for (3.1) containing $[u_i, u_j]$ and only edges in E. We can direct the edges of Θ to obtain a tour T that contains the arc (u_i, u_j) and only arcs of D. The tour T satisfies (3.2) with equality, and so this inequality is stable for D. □

Now we use the above result to prove that several families of facet-defining inequalities for GTSP give facet-defining inequalities for GATSP.

3.1. The s-path inequalities. We begin with the k-path configurations described in [1] for the undirected case. For any odd $k \geq 3$ and any k-tuple of positive integers $(n_1, \ldots, n_k)$, with $n_i \geq 2$ for $i \in \{1, \ldots, k\}$, let $P(n_1, \ldots, n_k)$ be the directed graph with node set

$$V_P = \{Y, Z\} \cup \left\{u_j^i \mid i \in \{1, \ldots, k\};\ j \in \{1, \ldots, n_i\}\right\}$$

and arc set

$$A_P = \left\{(u_j^i, u_{j+1}^i), (u_{j+1}^i, u_j^i) \mid i \in \{1, \ldots, k\};\ j \in \{0, \ldots, n_i\}\right\}.$$

We call $P(n_1, \ldots, n_k)$ a *directed k-path configuration.* For convenience, we label $u_0^i = Y$ and $u_{n_i+1}^i = Z$. The arc set of a directed 3-path configuration $P(2, 2, 2)$ is shown in Figure 3.1.

Let $D = (V_P, A)$ be any graph that contains $P(n_1, \ldots, n_k)$ as a subgraph. An *s-path inequality* associated with $P(n_1, \ldots, n_k)$ is the inequality

$$cx \geq c_0 \tag{3.1.1}$$

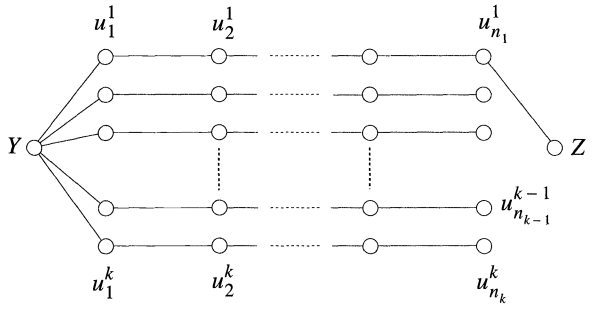


FIG. 3.2.

on $\mathbb{R}^A$, where c and c_0 are defined as follows:

$$c_0 = 1 + \sum_{i=1}^{k} \frac{n_i + 1}{n_i - 1},$$

$$c(Y, Z) = c(Z, Y) = 1;$$

for $i \in \{1, \ldots, k\}$, $j, q \in \{0, \ldots, n_i + 1\}$ with $0 < |j - q| < n_i + 1$,

$$c(u_j^i, u_q^i) = \frac{|j - q|}{n_i - 1};$$

for $i, j \in \{1, \ldots, k\}$, $i \neq r$ $j \in \{1, \ldots, n_i\}$, $q \in \{1, \ldots, n_r\}$,

$$c(u_j^i, u_q^r) = \frac{1}{n_i - 1} + \frac{1}{n_r - 1} + \left| \frac{j - 1}{n_i - 1} - \frac{q - 1}{n_r - 1} \right|.$$

Note that the coefficients of (3.1.1) described above are symmetric and identical to those described in [1] for the path inequality on the corresponding undirected graph $G = (V_P, E)$.

THEOREM 3.1.1. *The s-path inequality is facet defining for* GATSP(D).

Proof. Let $G = (V, E(P))$ be an undirected k-path configuration, i.e., the undirected graph whose edge set is

$$E(P) = \left\{ [u_j^i, u_{j+1}^i] \mid i \in \{1, \ldots, k\};\ j \in \{0, \ldots, n_i\} \right\}.$$

In [1] it is proven that the undirected path inequality is stable for the skeleton G. It is straightforward to apply Theorem 3.1, using the spanning tree S with edge set (see Figure 3.2)

$$\begin{aligned} E(S) = &\left\{ [u_j^1, u_{j+1}^1] \mid j \in \{0, \ldots, n_1\} \right\} \\ &+ \left\{ [u_j^i, u_{j+1}^i] \mid i \in \{2, \ldots, k\};\ j \in \{0, \ldots, n_i - 1\} \right\} \end{aligned}$$

and an arbitrary order for the edges in $E(P) - E(S)$. □

3.2. The s-wheelbarrow inequalities. Next we turn to the k-wheelbarrow configuration also discussed in [1] for the undirected case. For any odd integer $k \geq 3$ and k-tuple of positive integers $(n_1, n_2, \ldots, n_k)$ with $n_i \geq 2$ for $i \in \{1, 2, \ldots, k\}$, we define a *directed k-wheelbarrow configuration* $W(n_1, \ldots, n_k) = (V_W, A_W)$ where the

node set V_W is given by deleting the node Z of the k-path configuration and arc set A_W is given by

$$A_W = A_P - \left\{(u^i_{n_i}, Z), (Z, u^i_{n_i}) \mid i \in \{1, \ldots, k\}\right\} + \left\{(u^i_{n_i}, u^{i+1}_{n_{i+1}}), (u^{i+1}_{n_{i+1}}, u^i_{n_i}) \mid i \in \{1, \ldots, k\}\right\}.$$

All superscript indices are taken modulo k where $0 = k$. The arc set of $W(2, 3, 2)$ is shown in Figure 3.3.

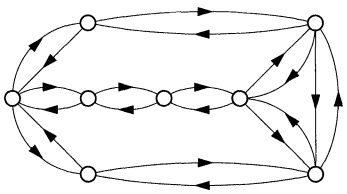


FIG. 3.3.

Let $D = (V_W, A)$ be any directed graph that contains $W(n_1, \ldots, n_k)$ as a subgraph. An *s-wheelbarrow inequality* associated with $W(n_1, \ldots, n_k)$ is the inequality

$$cx \geq c_0 \tag{3.2.1}$$

on $\mathbb{R}^A$, where c and c_0 are defined as for the s-path inequality (3.1.1). The inequality (3.2.1) is a restriction of the s-path inequality (3.1.1). The coefficients of (3.2.1) are symmetric and identical to those described in [1] for the wheelbarrow inequality on the corresponding undirected graph $G = (V_W, E)$.

THEOREM 3.2.1. *The s-wheelbarrow inequality is facet defining for* GATSP(D).

Proof. Let $G = (V, E(W))$ be the undirected k-wheelbarrow configuration, i.e., the undirected graph whose edge set is

$$E(W) = \left\{[u^i_j, u^i_{j+1}] \mid i \in \{1, \ldots, k\};\ j \in \{0, \ldots, n_i\}\right\} + \left\{[u^i_{n_i}, u^{i+1}_{n_{i+1}}] \mid i \in \{1, \ldots, k\}\right\}.$$

In [1] it is proven that the undirected wheelbarrow inequality is stable for the skeleton G. It is a simple exercise to apply Theorem 3.1, using the spanning tree S with edge set (see Figure 3.4)

$$E(S) = \left\{[u^i_j, u^i_{j+1}] \mid i \in \{1, \ldots, k\};\ j \in \{0, \ldots, n_i - 1\}\right\}$$

and an arbitrary order for the edges in $E(W) - E(S)$. □

3.3. The *s*-bicycle inequalities. Next we turn to the k-bicycle configuration also discussed in [1] for the undirected case. For any odd integer $k \geq 3$ and k-tuple of positive integers $(n_1, n_2, \ldots, n_k)$ with $n_i \geq 2$ for $i \in \{1, 2, \ldots, k\}$, we define a *directed k-bicycle configuration* $B(n_1, \ldots, n_k) = (V_B, A_B)$ where the node set

$$V_B = V_W - \{Y\}$$

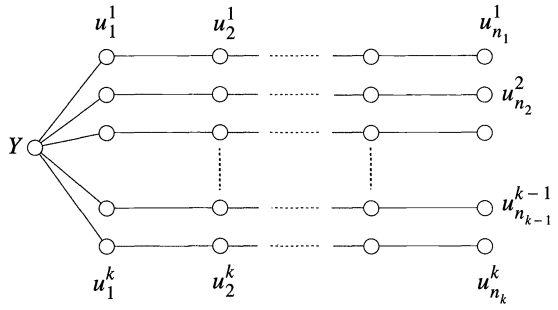


FIG. 3.4.

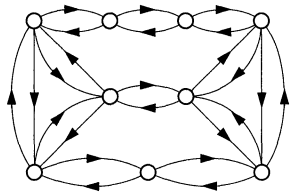


FIG. 3.5.

and arc set

$$\begin{aligned} A_B = A_W &- \{(u_1^i, Y), (Y, u_1^i) \mid i \in \{1, \ldots, k\}\} \\ &+ \{(u_1^i, u_1^{i+1}), (u_1^{i+1}, u_1^i) \mid i \in \{1, \ldots, k\}\} \\ &+ \{(u_1^i, u_1^{i+2}), (u_1^{i+2}, u_1^i), (u_{n_i}^i, u_{n_{i+2}}^{i+2}), (u_{n_{i+2}}^{i+2}, u_{n_i}^i) \mid i \in \{1, \ldots, k\}\}. \end{aligned}$$

All superscript indices are taken modulo k with $0 = k$. The arc set of the configuration $B(3, 2, 4)$ is shown in Figure 3.5.

Let $D = (V_B, A)$ be any graph that contains $B(n_1, \ldots, n_k)$ as a subgraph. An *s-bicycle inequality* associated with $B(n_1, \ldots, n_k)$ is the inequality

$$cx \geq c_0 \tag{3.3.1}$$

on $\mathbb{R}^A$, where c and c_0 are as defined for the s-path inequality (3.1.1). The inequality (3.3.1) is a restriction of (3.1.1). The coefficients of (3.3.1) are symmetric and identical to those described in [1] for the bicycle inequality of the corresponding undirected graph $G = (V_B, E)$.

THEOREM 3.3.1. *The s-bicycle inequality is facet defining for* GATSP(D).

Proof. Let $G = (V, E(B))$ be the undirected k-bicycle configuration. In [1] it is proven that the undirected bicycle inequality is stable for the skeleton G. To apply Theorem 3.1 we need only exhibit a spanning tree S of G, an ordering of the edges in $E(B) - E(S)$, and a suitable equality tour for each of these edges. Consider the spanning tree S with edge set

$$\begin{aligned} E(S) = &\{[u_j^i, u_{j+1}^i] \mid i \in \{1, \ldots, k\};\ j \in \{1, \ldots, n_i - 1\}\} \\ &+ \{[u_{n_i}^i, u_{n_{i+1}}^{i+1}] \mid i \in \{1, 3, \ldots, k-2\}\} \\ &+ \{[u_1^i, u_1^{i+1}] \mid i \in \{2, 4, \ldots, k-1\}\}. \end{aligned}$$

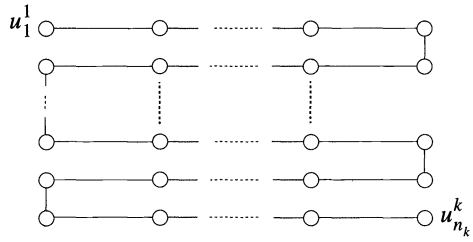


FIG. 3.6.

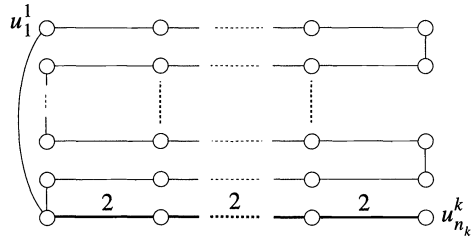


FIG. 3.7.

The tree S is as shown in Figure 3.6.

The edges in $E(B) - E(S)$ are ordered as follows:

$$\Big\langle [u_1^1, u_1^k];\ [u_{n_1}^1, u_{n_k}^k];\ \big\{[u_1^i, u_1^{i+1}], [u_{n_{i+1}}^{i+1}, u_{n_{i+2}}^{i+2}] \mid i \in \{1, 3, \ldots, k-2\}\big\};$$
$$\big\{[u_1^i, u_1^{i+2}], [u_{n_i}^i, u_{n_{i+2}}^{i+2}] \mid i \in \{1, 2, \ldots, k\}\big\}\Big\rangle.$$

For the edge $e = [u_1^1, u_1^k]$ consider the tour

$$\Theta_1 = E(S) + \{[u_j^k, u_{j+1}^k] \mid j \in \{1, \ldots, n_k - 1\}\} + \{[u_1^1, u_1^k]\},$$

which is shown in Figure 3.7. For the edge $[u_{n_1}^1, u_{n_k}^k]$ consider the tour

$$\Theta_2(0) = \Theta_1 + \{[u_{n_1}^1, u_{n_k}^k]\} + \big\{[u_j^1, u_{j+1}^1] \mid j \in \{1, \ldots, n_1 - 1\}\big\}$$
$$- \{[u_1^1, u_1^k]\} - \big\{[u_j^k, u_{j+1}^k] \mid j \in \{1, \ldots, n_k - 1\}\big\}.$$

We now define a collection of tours $\Theta_2(i)$. For the edge $[u_1^i, u_1^{i+1}]$ with $i \in \{1, 3, \ldots, k-2\}$,consider the tour

$$\Theta_2(i) = \Theta_2(i-1) + \{[u_1^i, u_1^{i+1}]\} + \big\{[u_j^{i+1}, u_{j+1}^{i+1}] \mid j \in \{1, \ldots, n_{i+1} - 1\}\big\}$$
$$- \{[u_{n_i}^i, u_{n_{i+1}}^{i+1}]\} - \big\{[u_j^i, u_{j+1}^i] \mid j \in \{1, \ldots, n_i - 1\}\big\}.$$

The tour $\Theta_2(1)$ is shown in Figure 3.8.

For the edge $[u_{n_i}^i, u_{n_{i+1}}^{i+1}]$ with $i \in \{2, 4, \ldots, k-1\}$, consider the tour

$$\Theta_2(i) = \Theta_2(i-1) + \{[u_{n_i}^i, u_{n_{i+1}}^{i+1}]\}$$
$$+ \big\{[u_j^{i+1}, u_{j+1}^{i+1}] \mid j \in \{1, \ldots, n_{i+1} - 1\}\big\}$$
$$- \{[u_1^i, u_1^{i+1}]\} - \big\{[u_j^i, u_{j+1}^i] \mid j \in \{1, \ldots, n_i - 1\}\big\}.$$

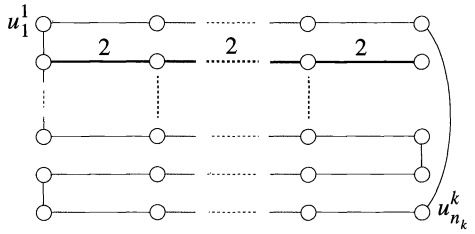


FIG. 3.8.

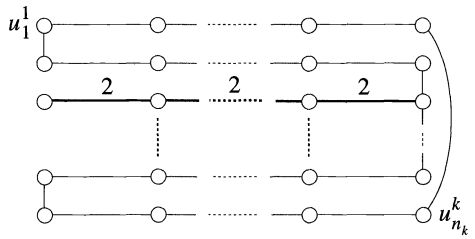


FIG. 3.9.

The tour $\Theta_2(2)$ is shown in Figure 3.9.

For any edge e of the form $[u_1^i, u_1^{i+2}]$ or $[u_{n_i}^i, u_{n_{i+2}}^{i+2}]$, Cornuéjols, Fonlupt, and Naddef [1] have shown equality tours using e with multiplicity one and all other edges from the set E_1 where

$$E_1 = \left\{[u_j^i, u_{j+1}^i] \mid i \in \{1, \ldots, k\};\ j \in \{1, \ldots, n_i - 1\}\right\} + \left\{[u_1^i, u_1^{i+1}], [u_{n_i}^i, u_{n_{i+1}}^{i+1}] \mid i \in \{1, \ldots, k\}\right\}.$$

For example, for the edge $[u_1^1, u_1^3]$, consider the tour

$$\Theta_3 = \Theta_2(2) - [u_1^1, u_1^2] - 2\{[u_1^3, u_2^3]\} + [u_1^2, u_1^3] + [u_1^1, u_1^3].$$

Also the tours constructed so far satisfy the undirected bicycle inequality at equality. Each of them contains a cycle that satisfies the requirements of (3.3). □

3.4. The *s*-crown inequalities. Now we turn to the *crown inequalities* described in [7]. For any integer $k \geq 2$, let $CR(p) = (V_C, A_C)$ be a *directed crown configuration*, where $p = 4k$ and

$$V_C = \{u_i \mid i \in \{1, \ldots, p\}\},$$
$$A_C = \{(u_i, u_{i+1}), (u_{i+1}, u_i), (u_i, u_{2k+i}), (u_{2k+i}, u_i), (u_i, u_{i+2}), (u_{i+2}, u_i) \mid i \in \{1, \ldots, p\}\}.$$

All indices are modulo p with $0 = p$. The arcs entering and leaving u_1 in $CR(8)$ are as shown in Figure 3.10 (a). The edges of the form $[u_i, u_{i+1}]$ are called *cycle edges* and those of the form $[u_i, u_{2k+i}]$ are called *diameters*.

Let $D = (V_C, A)$ be any directed graph that contains $CR(p)$ as a subgraph. An *s-crown inequality* associated with $CR(p)$ is the inequality

$$cx \geq c_0, \tag{3.4.1}$$

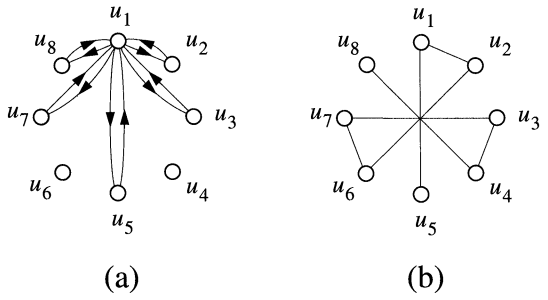


FIG. 3.10.

where $c_0 = 12k(k-1) - 2$, and

$$c(u_i, u_{i+j}) = \begin{cases} 4k - 6 + |j| & \text{for } 1 \le |j| \le 2k-1, \\ 2(k-1) & \text{for } j = 2k. \end{cases}$$

The coefficients of (3.4.1) are symmetric and identical to those described in [7] for the crown inequality on the corresponding undirected graph $G = (V_C, E)$.

THEOREM 3.4.1. *The s-crown inequality is facet defining for* GATSP(D).

Proof. Let $G = (V, E(CR))$ be the undirected crown configuration, i.e., the undirected graph whose edge set is

$$E(CR) = \{[u_i, u_{i+1}], [u_i, u_{2k+i}], [u_i, u_{i+2}] \mid i \in \{1, \dots, p\}\}.$$

In [7] it is proven that the undirected crown inequality is stable for the skeleton G. As usual, to apply Theorem 3.1 we need only exhibit a spanning tree S of G, an ordering of the edges in $E(CR) - E(S)$, and a suitable equality tour for each of these edges. Consider the spanning tree S with edge set

$$\begin{aligned} E(S) = {} & \{[u_i, u_{i+2k}] \mid i \in \{1, \dots, 2k\}\} + \{[u_{2i-1}, u_{2i}] \mid i \in \{1, 2, \dots, k\}\} \\ & + \{[u_{2(k+i)}, u_{2(k+i)+1}] \mid i \in \{1, \dots, k-1\}\}. \end{aligned}$$

For $k = 2$, the tree S is as shown in Figure 3.10 (b).

The edges in $E(CR) - E(S)$ are ordered as follows:

$$\begin{aligned} \Big\langle & [u_{4k}, u_1]; \\ & \{[u_{2k-j+1}, u_{2k-j+2}];\ [u_{4k-j}, u_{4k-j+1}];\ [u_{2k+j}, u_{2k+j+1}]; \\ & \qquad [u_{j+1}, u_{j+2}] \mid j \in \{1, 3, \dots, k \text{ (or } k-1 \text{ if } k \text{ is even})\}\}; \\ & \qquad \{[u_i, u_{i+2}] \mid i \in \{1, 2, \dots, 4k\}\}\Big\rangle. \end{aligned}$$

For $i \in \{1, 2, \dots, 4k\}$, let

$$E^+(i) = \{[u_i, u_{i+1}], [u_i, u_{2k+i}]\} \quad \text{and} \quad E^-(i) = \{[u_i, u_{i-1}], [u_i, u_{2k+i}]\}.$$

For the edge $[u_{4k}, u_1]$, consider the tour

$$\Theta_4(0) = \Theta_4(-2) = E(S) + E^-(1).$$

For $k = 2$, the tour $\Theta_4(0)$ is as shown in Figure 3.11 (a).

For $j = 1, 3, \ldots, k$ (or $k - 1$ if k is even) and for the edges $[u_{2k-j+1}, u_{2k-j+2}]$, $[u_{4k-j}, u_{4k-j+1}]$, $[u_{2k+j}, u_{2k+j+1}]$, $[u_{j+1}, u_{j+2}]$, consider the following tours (see Figure 3.11 (b) for the case $k = 2$):

$$\begin{aligned}
\Theta_4(2j-1) &= \Theta_4(2j-4) + E^+(2k-j+1) - E^-(4k-j+2),\\
\Theta_4(2j) &= \Theta_4(2j-1) + E^+(4k-j) - E^-(2k-j+1),\\
\Theta_4(2j+1) &= \Theta_4(2j-2) + E^-(2k+j+1) - E^+(j),\\
\Theta_4(2j+2) &= \Theta_4(2j+1) + E^-(j+2) - E^+(2k+j+1),
\end{aligned}$$

respectively.

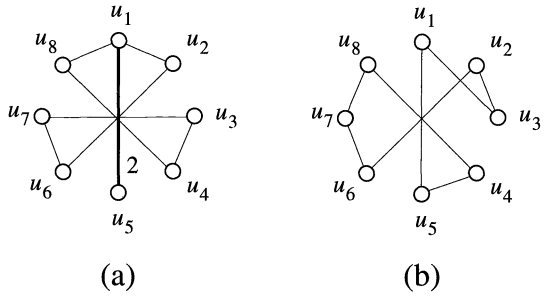


FIG. 3.11.

For any edge $e = [u_i, u_{i+2}]$, Naddef and Rinaldi [7] have exhibited equality Hamiltonian cycles using only e, cycle edges, and diameters. Each of these cycles is easily constructed by taking the cycle edge $[u_{i+1}, u_{i+2}]$, all the diameters except $[u_{i+2+2k}, u_{i+2}]$, and the other cycle edges necessary to complete the cycle (see Figure 3.11 (b) for $k = 2$).

All the tours considered so far satisfy the undirected crown inequality with equality. Each of them contains a cycle that satisfies the requirements of (3.3). □

3.5. Composition of symmetric inequalities. Let $G^1 = (V_{n_1}, E_1)$ and $G^2 = (V_{n_2}, E_2)$ be the skeletons of the inequalities

$$\mu^1 y^1 \geq \mu_0^1, \tag{3.5.1}$$

$$\mu^2 y^2 \geq \mu_0^2, \tag{3.5.2}$$

facet defining for GTSP(K_{n_1}) and GTSP(K_{n_2}), respectively. Let $V^1 = \{v_1^1, v_2^1, \ldots, v_s^1\}$ and $V^2 = \{v_1^2, v_2^2, \ldots, v_s^2\}$, with $s \geq 2$, be subsets of V_{n_1} and V_{n_2}, respectively. For simplicity we assume that the subgraph of G^1 induced by V^1 is isomorphic to the subgraph of G^2 induced by V^2, and that v_i^1 corresponds to v_i^2, for $i = 1, 2, \ldots, s$, in the isomorphism.

The *s-sum* of G^1 and G^2 is the graph $G = (V_n, E)$, obtained by identifying the node v_i^1 with v_i^2 for $i = 1, 2, \ldots, s$ and removing the resulting parallel edges. Here $n = n_1 + n_2 - s$, $V_n = (V_{n_1} - V^1) + (V_{n_2} - V^2) + W$, $W = \{w_1, w_2, \ldots, w_s\}$, w_i results from the identification of v_i^1 and v_i^2 for $i \in \{1, 2, \ldots, s\}$, and, finally, E is such that the subgraph of G induced by $(V_{n_i} - V^i) + W$ is isomorphic to G^i for $i = 1, 2$.

If the condition

$$\mu^1(v_i^1, v_j^1) = \mu^2(v_i^2, v_j^2) \quad \text{for all } i \neq j \in \{1, 2, \ldots, s\} \tag{3.5.3}$$

is satisfied (for $s = 2$ this can always be obtained by multiplying (3.5.2) by a suitable multiplier), we can define an inequality

$$\mu y \geq \mu_0 \tag{3.5.4}$$

on $\mathbb{R}^{E_n}$ *obtained by s-sum of* (3.5.1) *and* (3.5.2) in the following way. To the edges with one endpoint in $V_1 - W$ and the other in $V_2 - W$, which we call the *crossing edges of* (3.5.4), we assign an ordering

$$\langle e_1, e_2, \ldots, e_r \rangle. \tag{3.5.5}$$

Then

$$\mu(v_i, v_j) = \mu^r(v_i, v_j) \quad \text{for } \{v_i, v_j\} \subseteq V^r \text{ and } r \in \{1, 2\}.$$

The right-hand side μ_0 is the length of an optimal solution of *gtsp* in G when the weights are given by μ. Finally, for a crossing edge e the coefficient μ_e is computed by a sequential lifting procedure based on the ordering (3.5.5).

If (3.5.3) and

$$\mu^1(v_i^1, v_{i+j}^1) = \sum_{p=1}^{j-1} \mu^1(v_p^1, v_{p+1}^1) \quad \text{for all } j \in \{1, 2, \ldots, s\}$$

hold (for $s = 2$ this is clearly always the case), the inequality (3.5.4) is said to be obtained by a *linear s-sum* of (3.5.1) and (3.5.2). In [6] it is shown that for the linear s-sum,

$$\mu_0 = \mu_0^1 + \mu_0^2 - 2\sum_{p=1}^{s-1} \mu^1(v_p^1, v_{p+1}^1).$$

A subset $V = \{v_1, v_2, \ldots, v_s\}$ of nodes of the skeleton of an inequality is called *identifiable* if there exists an equality tour for the inequality that contains the family $2\{[v_i, v_{i+1}] \mid i \in \{1, 2, \ldots, s-1\}\}$. The following theorem is proven in [6].

THEOREM 3.5.1. *Let* (3.5.4) *be obtained by a linear s-sum of* (3.5.1) *and* (3.5.2). *If* V^1 *and* V^2 *are identifiable, the graph* G *is a skeleton of* (3.5.4). *The set* W *and all the subsets of* V_{n_1} *and* V_{n_2} *which are identifiable for* (3.5.1) *and* (3.5.2), *respectively, also are identifiable for* (3.5.4).

By Theorem 3.5.1 it is possible to show by induction that the graph obtained by repeatedly applying the s-sum operation is a skeleton for the corresponding inequality that is produced with this procedure.

The following theorem extends Theorem 3.5.1 to GATSP, using basically the same proof given in [6]. As done for Theorem 3.1, let us denote by $D^1 = (V_{n_1}, A^1)$, $D^2 = (V_{n_2}, A^2)$, and $D = (V_n, A)$ the bidirected graphs obtained from the skeletons G^1, G^2, and G, respectively, by replacing each edge $[v_i, v_j]$ of the three graphs by the pair of arcs (v_i, v_j) and (v_j, v_i). Define the inequalities

$$\pi^1 x^1 \geq \pi_0^1, \tag{3.5.6}$$

$$\pi^2 x^2 \geq \pi_0^2, \tag{3.5.7}$$

$$\pi x \geq \pi_0, \tag{3.5.8}$$

with $\pi^1(v_i, v_j) = \pi^1(v_j, v_i) = \mu_f^1$, $\pi^2(v_i, v_j) = \pi^2(v_j, v_i) = \mu_f^2$, and $\pi(v_i, v_j) = \pi(v_j, v_i) = \mu(v_i, v_j)$, where $\pi_0^1 = \mu_0^1$, $\pi_0^2 = \mu_0^2$, and $\pi_0 = \mu_0$.

THEOREM 3.5.2. *Let* (3.5.4) *be obtained by linear s-sum of* (3.5.1) *and* (3.5.2). *If* V^1 *and* V^2 *are identifiable and if* D^1 *and* D^2 *are skeletons for the inequalities* (3.5.6) *and* (3.5.7), *respectively, then* D *is a skeleton for the inequality* (3.5.8).

Proof. The inequality (3.5.8) is clearly valid for GATSP(D). For $i = 1, 2$, let Θ^i be an equality tour of $\mu^i y^i \geq \mu_0^i$ containing the family $2\left\{[v_p^i, v_{p+1}^i] \mid p \in \{1, 2, \ldots, s-1\}\right\}$; such a tour exists since V^i is assumed to be identifiable. Let T^i be the directed tour of D^i obtained by traveling the edges of the tour $\Theta^i - 2\left\{[v_p^i, v_{p+1}^i] \mid p \in \{1, 2, \ldots, s-1\}\right\}$ in some direction. Let $\gamma x \geq \gamma_0$ be any valid inequality for GATSP(D) for which

$$\{x \in \text{GATSP}(D) \mid \pi_A x = \pi_0\} \subseteq \{x \mid \gamma x = \gamma_0\}.$$

To prove the theorem we have to compute all the coefficients of the vector γ by using equality tours for (3.5.8). Consider a set $\mathcal{T}^1$ of $(|A^1| - n_1 + 1)$ linearly independent directed tours satisfying (3.5.6) at equality (this set exists since (3.5.6) is facet defining for GATSP(D^1)). The directed tours obtained by adding T^2 to each tour of $\mathcal{T}^1$ are also linearly independent and satisfy (3.5.8) at equality. From these tours and from the flow equations $F_D x = 0$, it follows that there exists $\rho^1 > 0$ such that $\gamma_a = \rho^1 \pi_a^1$ for all a in A isomorphic to an arc of A^1. By the same argument, interchanging the rôles of D^1 and D^2, it follows that there exists $\rho^2 > 0$ such that $\gamma_a = \rho^2 \pi_a^2$ for all a in A isomorphic to an arc of A^2. Since the arc a of D which is isomorphic to (v_1^1, v_2^1) is also isomorphic to (v_1^2, v_2^2), it follows that $\mu_a = \rho^1 \pi^1(v_1^1, v_2^1) = \rho^2 \pi^2(v_1^2, v_2^2)$, and since $\pi^1(v_1^1, v_2^1) = \pi^2(v_1^2, v_2^2)$ we have $\rho^1 = \rho^2$ and the theorem follows. □

For simplicity, let us call in this subsection any of the configurations of §§3.1, 3.2, and 3.3 a *path configuration* and let us call the corresponding inequality a *path inequality.* A path configuration (or a path inequality) is called *regular* if $n_i = n$ for $i = 1, 2, \ldots, k$. The nodes Y and Z are called the *odd* nodes of the configuration; all the other nodes are called *even.*

We consider now the 2-sum of path inequalities. In this case the only sets that qualify to be identifiable are the pairs $\{u_j^i, u_{j+1}^i\}$ for some $i \in \{1, 2, \ldots, k\}$ and $j \in \{0, \ldots, n_i\}$. When two nodes u' and u'' are identified in the 2-sum, the resulting node is even only if both u' and u'' are even and odd in all the other cases.

Next we consider the *regular parity path tree inequality*, defined recursively as

(i) a regular path, wheelbarrow, or bicycle inequality,

(ii) the 2-sum of a regular parity path tree inequality and a regular path, wheelbarrow, or bicycle inequality, with the condition that an odd node be identified only with an odd node.

A *regular parity path tree configuration* and a *directed regular parity path tree configuration* are defined analogously. In [6] it is shown that a regular parity path tree inequality is stable for the corresponding regular parity path tree configuration; i.e., the coefficients of the crossing edges do not depend on the sequence (3.5.5) and thus can be given in closed form. If (3.1) is a regular parity path tree inequality, the corresponding inequality (3.2) for the directed case is called a *regular parity s-path tree inequality.* By applying Theorem 3.5.2 to the regular parity path tree configurations, we have the following.

THEOREM 3.5.3. *Let* D *be any directed graph that contains a directed regular parity path tree configuration as a subgraph. Then the corresponding regular parity s-path tree inequality is facet defining for* GATSP(D).

3.6. Extensions of symmetric inequalities. Let

$$cx \geq c_0 \tag{3.6.1}$$

be a facet-defining inequality for GTSP(K_n) and let e be an edge in E_n. We say that the inequality

$$c^*x^* \geq c_0^* \tag{3.6.2}$$

defined in $\mathbb{R}^{E_{n+2h}}$, with $h \geq 1$, is obtained from (3.6.1) by *cloning the edge e (h times)* if

(a) the restriction of (3.6.2) to $\mathbb{R}^{E_n}$ coincides with (3.6.1),

(b) the remaining coefficients of (3.6.2) are defined as follows (where we assume, without loss of generality, that $e = [v_{n-1}, v_n]$; see Figure 3.12 for $h = 1$):

$$c_0^* = c_0 + 2hc_e,$$

$$\begin{aligned}
&c^*[v_i, v_{n+j}] = c[v_i, v_{n-1}] && \text{for } 1 \leq i \leq n-2,\ 1 \leq j \leq 2h-1 \text{ and } j \text{ odd},\\
&c^*[v_i, v_{n+j}] = c[v_i, v_n] && \text{for } 1 \leq i \leq n-2,\ 2 \leq j \leq 2h \text{ and } j \text{ even},\\
&c^*[v_{n+i}, v_{n+j}] = 2c_e && \text{for } -1 \leq i < j \leq 2h \text{ and } j-i \text{ even},\\
&c^*[v_{n+i}, v_{n+j}] = c_e && \text{for } -1 \leq i < j \leq 2h \text{ and } j-i \text{ odd}.
\end{aligned}$$

Roughly speaking, cloning the edge $e = [v_{n-1}, v_n]$ (h times) amounts to replacing e and the edge sets $\delta(v_{n-1})$ and $\delta(v_n)$ with h their replicas.

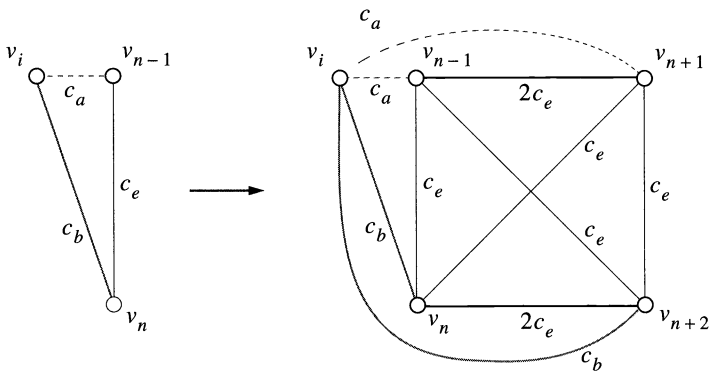


FIG. 3.12.

An edge $e = [u, v] \in E_n$ is called *c-clonable* if $c(\Theta) \geq c_0 + (t-2)c_e$ for every tour Θ of K_n, where t is the minimum of the degrees of u and v in Θ. We call a node, $v \in V_n$, *q-critical* for (3.6.1) if for an optimal solution Θ^* of *gtsp* in $K_n - \{v\}$, when the weights are given by the restriction of (3.6.1), $c(\Theta^*) = c_0 - q$.

THEOREM 3.6.1. *If $e = [v_{n-1}, v_n]$ is a c-clonable edge such that v_{n-1} and v_n are $2c_e$-critical for* (3.6.1), *then the following hold:*

(a) (3.6.2) *is facet defining for* GTSP(K_{n+2h});

(b) *if $f = [z_1, z_2] \neq e$ is an edge in E_n such that z_1 and z_2 are $2c_f$-critical for* (3.6.1), *then z_1 and z_2 are $2c_f^*$-critical for* (3.6.2);

(c) *if $G = (V_n, E)$ is a skeleton for* (3.6.1), *then the graph $G' = (V_{n+2h}, E')$ is a skeleton for* (3.6.2), *where*

$$\begin{aligned}
E' = E &\cup \{[v_{n+i}, v_{n+i+1}], [v_{n+i}, v_{n+i-1}], [v_{n+i+1}, v_{n+i-2}] \mid i = 1, 3, \ldots, 2h-1\}\\
&\cup \{[w, v_{n+i}] \mid [w, v_{n-1}] \in E,\ i = 1, 3, \ldots, 2h-1\}\\
&\cup \{[w, v_{n+i+1}] \mid [w, v_n] \in E,\ i = 1, 3, \ldots, 2h-1\}.
\end{aligned}$$

Proof. In [6], (a) and (b) are explicitly proven. One can easily show that (c) holds, following the same technique used in [6]. □

The following theorem gives sufficient conditions under which cloning preserves the properties required in (3.3).

THEOREM 3.6.2. *If* (3.6.1) *satisfies the conditions of Theorem* 3.6.1, (3.3) *holds for* (3.6.1) *and if the spanning tree* S *and all the tours* $\Theta(e_i)$ *of* (3.3) *contain the edge* e, *then* (3.3) *holds also for* (3.6.2).

Proof. We show that if the theorem is true for $h = 1$, then the proof can be completed by induction on h. Let $G = (V_n, E)$ and $G' = (V_{n+2}, E)$ be the skeletons of (3.6.1) and (3.6.2), respectively. Let S, $\langle e_1, e_2, \ldots, e_t \rangle$, and $\Theta(e_i)$ be the spanning tree, the edge ordering, and the equality tours, respectively, that satisfy (3.3) for (3.6.1). Consider the spanning tree S' of G' whose edge set is the union of the edge set of S and the edges $\{[v_n, v_{n+1}], [v_{n+1}, v_{n+2}]\}$. Consider the following ordering of the edges of G':

$$\big\langle [v_{n-1}, v_{n+2}];\ \langle e_1, e_2, \ldots, e_t \rangle; \{[w, v_{n+1}] \mid [w, v_{n-1}] \in E\}\, \{[w, v_{n+2}] \mid [w, v_n] \in E\} \big\rangle.$$

Let Φ be a solution of *gtsp* with weights given by (3.6.1) in $G - \{v_n\}$. Then $\Phi' = \Phi + 2\{e\}$ is an equality tour for (3.6.1). Let $E_1 = \{[v_{n-1}, v_{n+2}], [v_{n+2}, v_{n+1}], [v_{n+1}, v_n]\}$. For the edge $[v_{n-1}, v_{n+2}]$ consider the tour $\Phi' - [v_{n-1}, v_n] + E_1$. This tour satisfies (3.3) for (3.6.2). For the edge e_i the tour $\Theta(e_i) - \{e\} + E_1$ satisfies (3.3) for (3.6.2), for $i = 1, 2, \ldots, t$. For an edge $[w, v_{n+1}]$ consider the following equality tour for (3.6.2):

$$\Phi - [w, v_{n-1}] + [w, v_{n+1}] + [v_{n+1}, v_{n+2}] + [v_{n+2}, v_{n-1}],$$

where Φ is an equality tour for (3.6.1) that contains the edge $[w, v_{n-1}]$. Clearly this tour satisfies (3.3). Similarly, for an edge $[w, v_{n+2}]$ consider the following equality tour for (3.6.2)

$$\Phi - [w, v_n] + [w, v_{n+2}] + [v_{n+2}, v_{n+1}] + [v_{n+1}, v_n],$$

where Φ is an equality tour for (3.6.1) that contains the edge $[w, v_n]$. This tour satisfies (3.3) as well. □

Let (3.1) be an inequality obtained by cloning some of the edges $e = [u_1^i, u_2^i]$, with $n_i = 2$, of a path (or a wheelbarrow or a bicycle) inequality. We call (3.1) an *extended path* (or *wheelbarrow* or *bicycle*) *inequality* and (3.2) an *extended s-path* (or *s-wheelbarrow* or *s-bicycle*) *inequality.* Observe that if $n_i > 2$, then the edge e is not μ-clonable.

THEOREM 3.6.3. *The extended s-path and the extended s-wheelbarrow inequalities are facet defining for* GATSP(D_n). *The extended s-bicycle inequality is facet defining for* GATSP(D_n) *if none of the edges* $[u_1^r, u_1^{r+2}]$, $[u_{n_r}^r, u_{n_{r+2}}^{r+2}]$ *for* $r = 1, 2, \ldots, k$, *is adjacent with more than one cloned edge.*

Proof. In [5] it is proven that any edge $f = [u_1^i, u_2^i]$, for $i \in \{1, \ldots, k\}$ with $n_i = 2$, of an undirected k-path (or k-wheelbarrow or k-bicycle) configuration that defines an inequality $cx \geq c_0$ is c-clonable and that u_1^i and u_2^i are $2c_f$-critical. Consequently the extended path, wheelbarrow, and bicycle inequalities are facet defining for GTSP(K_n). Moreover, for the path, the wheelbarrow, and the bicycle inequalities, (3.3) holds and the spanning trees given in the proofs of the Theorems 3.1.1, 3.2.1, and 3.3.1, as well as the tours $\Theta(e_i)$, contain all the edges $[u_1^i, u_2^i]$ for $i \in \{1, \ldots, k\}$. The only exceptions occur for the bicycle inequalities and concern the tours $\Theta(e_i)$ when $e_i = [u_1^r, u_1^{r+2}]$ and $e_i = [u_{n_r}^r, u_{n_{r+2}}^{r+2}]$ for $r = 1, 2, \ldots, k$. Consider, for example, the edge $[u_1^1, u_1^3]$. The corresponding tour, denoted by Θ_3 in the proof of Theorem 3.3.1, does not contain

the edge $[u_1^3, u_2^3]$. If this is a cloned edge, then the edge $[u_1^1, u_2^1]$ is not cloned, by the assumption of the theorem. It is now easy to construct another equality tour Θ_3' satisfying (3.3) that does not contain $[u_1^1, u_2^1]$ but contains $[u_1^3, u_2^3]$, $[u_1^1, u_1^3]$, and all the edges in the set $[u_1^r, u_2^r]$ for $r \in \{2, 4, \ldots, k\}$.

$$\Theta_3' = \Theta_2(0) - [u_1^2, u_1^3] - 2\{[u_1^1, u_2^1]\} + [u_1^1, u_1^2] + [u_1^1, u_1^3],$$

where $\Theta_2(0)$ is defined in the proof of Theorem 3.3.1. By repeating this argument for all the edges $[u_1^r, u_1^{r+2}]$, $[u_{n_r}^r, u_{n_{r+2}}^{r+2}]$ for $r = 1, 2, \ldots, k$, we construct a set of tours $\Theta(e_i)$ satisfying the conditions of Theorem 3.6.2. □

Let (3.1) be an inequality obtained by cloning a subset of the diameters of a crown inequality. We call (3.1) an *extended crown inequality* and (3.2) an *extended s-crown inequality.*

THEOREM 3.6.4. *The extended s-crown inequality defines a facet of* GATSP (D_n) *if none of the edges* $[u_r, u_{r+2}]$, *for* $r = 1, 2, \ldots, 4k$, *is adjacent with more than one cloned diameter.*

Proof. In [7] it is proven that any diameter $f = [u_i, u_{2k+i}]$ for $i \in \{1, \ldots, 2k\}$ of an undirected crown configuration is c-clonable and that u_i and u_{2k+i} are $2c_f$-critical for the corresponding inequality $cx \geq c_0$. By repeating the argument for each cloning, one has by induction that the extended crown inequalities are facet defining for GTSP(K_n). Moreover, for these inequalities (3.3) holds and all the diameters are contained in the spanning tree given in the proof of Theorem 3.4.1 and in all tours $\Theta(e_i)$, with the exception of the tours $\Theta(e_i)$ with $e_i = [u_r, u_{r+2}]$ for $r = 1, 2, \ldots, 4k$ (see Figure 3.11 (b)). If the tour $\Theta([u_r, u_{r+2}])$ does not contain the diameter $[u_r, r_{2k+r}]$ and this diameter is cloned, then by the assumption of the theorem the diameter $[u_{r+2}, u_{2k+r+2}]$ is not cloned. It is easy to construct an equality tour that satisfies (3.3), does not contain $[u_{r+2}, u_{2k+r+2}]$, and contains $[u_r, r_{2k+r}]$, $[u_r, u_{r+2}]$, and all the remaining diameters (see [7]). Thus the conditions of Theorem 3.6.2 are satisfied. □

4. Conclusions. We have defined several inequalities and shown that they define facets of GATSP. These inequalities are derived from facet-defining inequalities for GTSP. These facet-defining inequalities for GTSP are the *path* inequalities [1] and their extensions [5] (the classical *comb* (see, e.g., [3]) and *chain* [9] inequalities are a small subset of these inequalities), the *path-tree* inequalities [6] (the classical *clique-tree* inequalities (see, e.g., [3]) are a small subset of these inequalities), and the *crown* inequalities [7].

Which of these inequalities are also facet defining for ATSP? A possible way to answer this question could be to use the technique introduced in [8] and applied in [5] to several facet-defining inequalities in the undirected case. We have checked, for all the inequalities described in §3, whether they define facets of ATSP(n), with $6 \leq n \leq 12$, and we found that all of them do with only two exceptions. The two exceptions are the bicycle inequality for ATSP(6) and the crown inequality for ATSP(8).

By adding $a - 2$ to the coefficients of a bicycle inequality associated to the arcs shown in Figure 4.1 (a) and by adding $2 - a$ to the coefficients of the arcs shown in Figure 4.1 (b), we obtain a valid inequality for ATSP(6) for all values of a in the open interval $(0, 4)$. The inequality defines the same face of ATSP(6) defined by the bicycle inequality. For $a = 0$ and $a = 4$ the inequality is facet defining for ATSP(6).

By adding $a-2$ to the coefficients of a crown inequality associated to the arcs that are shown in Figure 4.2 (a) and by adding $2 - a$ to the coefficients of the arcs shown in Figure 4.2 (b), we produce an inequality that is valid for ATSP(8) for all values of

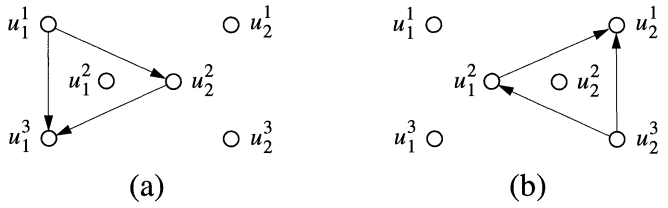


FIG. 4.1.

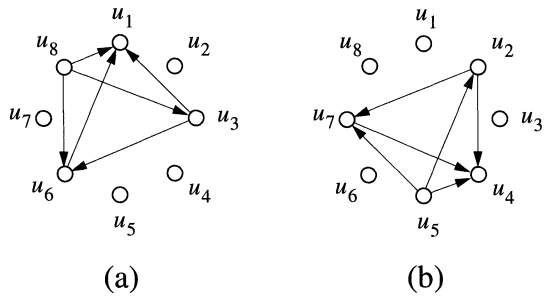


FIG. 4.2.

a in the open interval $(0, 4)$ and that defines the same face of ATSP(8) defined by the crown inequality. For $a = 0$ and $a = 4$ the inequality is facet defining for ATSP(8).

Acknowledgments. The authors wish to thank two anonymous referees for comments that led to improvements in the presentation.

REFERENCES

[1] G. CORNUÉJOLS, J. FONLUPT, AND D. NADDEF, *The traveling salesman problem on a graph and some related integer polyhedra*, Math. Programming, 33 (1985), pp. 1–27.

[2] B. FLEISCHMANN, *A new class of cutting planes for the symmetric travelling salesman problem*, Math. Programming, 40 (1988), pp. 225–246.

[3] M. GRÖTSCHEL AND M. PADBERG, *Polyhedral theory*, in The Traveling Salesman Problem, E. L. Lawler et al., eds., Wiley & Sons, Chichester, 1985, pp. 251–305.

[4] M. JÜNGER, G. REINELT, AND G. RINALDI, *The traveling salesman problem*, in Network Models, M. O. Ball et al., eds., Handbooks of Operations Research and Management Science, Vol. 7, North-Holland, Amsterdam, 1995, pp. 225–330.

[5] D. NADDEF AND G. RINALDI, *The Symmetric Traveling Salesman Polytope: New Facets from the Graphical Relaxation*, Report R. 248, Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche, Rome, 1988.

[6] ———, *The symmetric traveling salesman polytope and its graphical relaxation: Composition of valid inequalities*, Math. Programming A, 51 (1991), pp. 359–400.

[7] ———, *The crown inequalities for the symmetric traveling salesman polytope*, Math. Oper. Res., 17 (1992), pp. 308–326.

[8] ———, *The graphical relaxation: A new framework for the symmetric traveling salesman polytope*, Math. Programming A, 58 (1993), pp. 53–88.

[9] M. PADBERG AND S. HONG, *On the symmetric travelling salesman problem: A computational study*, Math. Programming Study, 12 (1980), pp. 78–107.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 625–642, November 1996

008

CAYLEY GRAPHS WITH NEIGHBOR CONNECTIVITY ONE*

L. L. DOTY†, R. J. GOLDSTONE‡, AND C. L. SUFFEL§

Abstract. We give an algebraic characterization of the generating sets of Cayley graphs with neighbor connectivity equal to one for a class of Cayley graphs that includes the Cayley graphs of all abelian groups. We also show that the determination of the neighbor connectivity of a graph is NP-hard.

Key words. Cayley graphs, neighbor connectivity

AMS subject classifications. 05C25, 05C40

1. Introduction.

In a series of papers, [1], [3], and [4], Gunther and Hartnell proposed modelling an underground resistance movement as a graph in which the vertices represent members of the resistance and the edges represent lines of communication. Unlike the conventional situation where the vertices of a graph represent components of a communication network that fail as individual elements, in the resistance movement scenario if a member of the underground is arrested, all of his or her neighbors are betrayed and so are rendered ineffective. Hence in the graph model, whenever a vertex is "subverted," the entire closed neighborhood of the vertex is deleted from the graph. The minimum number of vertices whose subversion results in an empty, complete, or disconnected subgraph is called the neighbor connectivity of the graph.

In order to describe these ideas more precisely, the following definitions are used: suppose Γ is a graph with vertex set V. For any subset $A \subset V$, $N[A] = A \cup \{v \in V \mid v \text{ is adjacent to } a \text{ for some } a \in A\}$ is called the *closed neighborhood* of A, while $N(A) = N[A] \setminus A$ is called the *open neighborhood* of A. If $A = \{a\}$, then we write $N[a]$ and $N(a)$, respectively. The remaining definitions essentially follow Gunther [4] and Gunther, Hartnell, and Nowakowski [6]. To *subvert a vertex* $v \in \Gamma$ means to remove all elements of $N[v]$ from Γ. The resulting induced subgraph, called the *survival subgraph*, is exactly the subgraph of Γ induced by $V \setminus N[v]$. A set of vertices B is called a *subversion strategy* whenever all of the vertices in $N[B]$ are deleted from the graph. The *survival subgraph* for B is the subgraph of Γ induced by $V \setminus N[B]$. If the survival subgraph for B is empty, complete, or disconnected, then B is called an *effective subversion strategy*. We say that a graph Γ has *neighbor connectivity* k, and we write $\mathrm{NC}(\Gamma) = k$ if k is the minimum size of an effective subversion strategy.

Neighbor connectivity is a variation on connectivity and in some ways is related directly to connectivity. Gunther, Hartnell, and Nowakowski [6] have shown, for example, that $\mathrm{NC}(\Gamma) \leq \kappa(\Gamma)$. Other properties of this new invariant, however, are decidedly unlike the corresponding connectivity properties. For example, whenever $m \geq 6$, $\mathrm{NC}(C_m) = 2$. If an edge is added to C_m in such a way as to avoid forming a

* Received by the editors April 4, 1994; accepted for publication (in revised form) December 12, 1995.

† Department of Computer Science and Mathematics, Marist College, Poughkeepsie, NY 12601 (jzca@maristb.marist.edu).

‡ Department of Mathematics, Bard College, Annandale-on-Hudson, NY 12504 (goldstone@bard.edu).

§ Department of Pure and Applied Mathematics, Stevens Institute of Technology, Hoboken, NJ 07030 (csuffel@sitvxa.stevens-tech.edu).

triangle, then the neighbor connectivity of the graph with the additional edge is less than the neighbor connectivity of the original one. The computation of connectivity is polynomial, while (as we show in the appendix) the decision problem for neighbor connectivity is NP-complete. This essential difference provides striking evidence that neighbor connectivity is fundamentally a more complex idea than connectivity.

Most previous work on neighbor connectivity has centered around the synthesis problem, that is, the design of graphs that have maximum neighbor connectivity or that are in some sense secure against subversion. For example, see [1]–[6]. Critically k-neighbor-connected graphs are investigated in [7].

In this paper, we begin to study the analysis problem for neighbor connectivity by narrowing the focus to Cayley graphs of neighbor connectivity one. In this context, we pursue the idea that algebraic properties of the generating set determine whether or not a finite Cayley graph has the neighbor connectivity analog of a cutvertex. We find algebraic conditions on the generating set that are sufficient to force neighbor connectivity one, and we prove that these conditions are also necessary for a large class of finite Cayley graphs that includes the Cayley graphs of all finite abelian groups.

In the appendix, we show that the decision problem for neighbor connectivity of an arbitrary graph is NP-complete.

1.1. The three main results. Our first main result is that the survival subgraph for the subversion strategy $\{1\}$ of a Cayley graph of a group G has isolated vertices if and only if the generating set S is a union of cosets for a nontrivial subgroup disjoint from S (Theorem 2). Whenever a subset of elements of G is a union of cosets of a nontrivial subgroup of G, we refer to the set as being *periodic*.

If the survival subgraph has no isolated vertices, we pass from the original Cayley graph to a certain type of minimal Cayley subgraph that we call *irreducible* (unless, of course, the original Cayley graph is already irreducible). The irreducible Cayley subgraph has the same survival subgraph for $\{1\}$ as the parent Cayley graph but is more tightly related to their common survival subgraph because the reduction process pares away vertices and edges that are irrelevant to the structure of the survival subgraph. In particular, the reduction process provides a mechanism for focusing on the portion of the generating set whose algebraic properties are decisive for neighbor connectivity.

The irreducible Cayley subgraph can be disconnected, in which case there is a simple description of the disconnectedness of the survival subgraph of the original graph. If the irreducible Cayley subgraph is connected (and its survival subgraph has no isolated vertices), our second main result is that the survival subgraph for $\{1\}$ is disconnected if the generating set is a union of cosets of a nontrivial subgroup H of G together with some—but not all—of the nonidentity elements of H (Theorem 7). Whenever a subset of elements of G has the property just described, we refer to the subset as being *nearly periodic*.

Our third main result concerns the necessity of the condition of near periodicity. For this we restrict our attention to finite Cayley graphs whose generating sets are a union of conjugacy classes—a condition that is trivially true for abelian groups but which may obtain in nonabelian groups as well. We call such generating sets *normal*. For groups with normal generating sets, we conclude that an irreducible connected Cayley graph whose survival subgraph has no isolated vertices is disconnected if and only if the generating set is nearly periodic (Theorem 14). The relation between the irreducible Cayley subgraph and the parent Cayley graph is controlled enough to allow us to rephrase these results entirely in terms of the structure of the generating set of

the original graph (Theorem 16).

1.2. Outline of the proof. The isolated vertex equivalence is easily settled with direct algebraic arguments (Theorem 2), and it is also straightforward to show that an irreducible Cayley graph with a nearly periodic generating set has a disconnected survival subgraph for $\{1\}$ (Theorem 7). We need a much more involved argument to establish a converse for Theorem 7. We consider Cayley graphs arising from a group with a normal generating set, and we show that whenever the survival subgraph of such a Cayley graph is disconnected but without isolated vertices, the Cayley graph has a nearly periodic generating set (Theorem 14). In the course of that argument, the following two critical concepts related to components emerge:

(1) A component C is said to be *nongenerating* if the subgroup generated by the vertices of C is a proper subgroup of G. Nongenerating components are important because we can show that S is nearly periodic if and only if the survival subgraph for $\{1\}$ has a nongenerating component (Proposition 3). In light of this equivalence, the form of our argument is that if the survival subgraph is disconnected without isolated vertices, then it must have a nongenerating component.

(2) In order to find a nongenerating component, we employ the concept of generators that are invisible from a component. For any component C of the survival subgraph, we say that S has elements *invisible* from C if $S \not\subset N(C)$. We show that if the Cayley graph is irreducible and if its survival subgraph for $\{1\}$ is disconnected without isolated vertices, then S must contain elements that are invisible from a component C of maximum vertex cardinality (Proposition 9).

Once we have a set of generators $I_C \subset S$ that are invisible from the component C, we inspect the subgroup generated by I_C. If this subgroup is not contained in the generating set, the portion that lies outside the generating set will contain at least one nongenerating component. If the subgroup generated by I_C lies inside the generating set, then a coset of the normal subgroup generated by I_C will be a nongenerating component. Thus nongenerating components are present in all cases, and we conclude that S is nearly periodic (Theorem 14).

2. Preliminary definitions and results. Let $\mathrm{CAY}(G, S)$ denote a Cayley graph whose vertices are the elements of the finite group G (written multiplicatively with identity 1) and whose edges connect $g, h \in G$ whenever $g^{-1}h \in S$. Since $\mathrm{CAY}(G, S)$ is a graph, we always require $S = S^{-1}$ and $1 \notin S$. The results presented in this paper characterize the set of generators of a large class of Cayley graphs (including all abelian Cayley graphs) with neighbor connectivity equal to one. Cayley graphs are vertex-transitive; thus we assume without loss of generality that the subversion strategy is $\{1\}$. Since $N[1] = S \cup \{1\}$, we use S_1, instead of the usual $N[1]$, to denote the closed neighborhood of $\{1\}$. Further, we use $\mathrm{SCAY}(G, S)$ to denote the survival subgraph induced by the vertex set $G \setminus S_1$. If H is a nontrivial subgroup of G, then a subset $Y \subset G$ is called *left H-periodic* (or *left periodic* if we do not need to refer to H), provided that $HY = Y$. This condition is equivalent to Y being a union of right cosets of H. For any subset $A \subset G$, we use $\langle A \rangle$ to denote the subgroup of G generated by A. Since G is finite, an arbitrary element of $\langle A \rangle$ can be viewed as a product of elements of A without recourse to inverses.

Lemma 1 collects some easily derivable algebraic properties that are useful when describing either the behavior of components in $\mathrm{SCAY}(G, S)$ or the periodic nature of subsets of G.

LEMMA 1. *Let* $\mathrm{CAY}(G, S)$ *be a Cayley graph.*

(a) *If X is a subset of G, then $N(X) = (XS) \setminus X$.*

(b) *If X is a left H-periodic subset of G, then either $H \subset X$ or $H \cap X = \emptyset$. In particular, if X is left H-periodic and $1 \notin X$, then $H \cap X = \emptyset$.*

(c) *For any nonempty subsets X, Y of G, $X^{-1}Y \subset Y$ if and only if $XY \subset Y$ if and only if $\langle X \rangle Y = Y$.*

(d) *If X is a left H-periodic subset of G, then $N(X)$ is left H-periodic.*

(e) *For any subset U of $G \setminus S_1$, U is a union of components of* $\mathrm{SCAY}(G, S)$ *if and only if $US \subset U \cup S$.*

Proof. The conclusion in part (a) follows directly from the definition of $N(X)$.

The conclusion in part (b) is a direct consequence of the fact that cosets form a partition of G.

For part (c), note that it is always true that $Y \subset \langle X \rangle Y$. The reverse containment follows from $XY \subset Y$ and the obvious induction on the length of an element of $\langle X \rangle$. The proof for the hypothesis $X^{-1}Y \subset Y$ is similar.

In part (d), the conclusion that $N(X)$ is left H-periodic follows immediately from part (a) and the observation that if A and B are left H-periodic and Y is any subset of G, then AY and $A \setminus B$ are left H-periodic.

To establish the equivalence in part (e), first note that if C is a component, then $CS \subset C \cup S$, by the definition of component. Since $(A \cup B)S = AS \cup BS$, the inclusion with C replaced by U follows directly. Conversely, suppose U is a set of vertices of $\mathrm{SCAY}(G, S)$ with the property that $US \subset U \cup S$. Let $u \in U$. We claim that if x is adjacent to u in $\mathrm{SCAY}(G, S)$, then $x \in U$. If x is adjacent to u, then $x = us$ for some $s \in S$, from which we conclude that $x \in US \subset U \cup S$. But $x \in G \setminus S$, and so $x \in (U \cup S) \setminus S \subset U$. Using an obvious induction on path length, we conclude that whenever x is joined by a path to u in $\mathrm{SCAY}(G, S)$, then $x \in U$. Hence U is a union of components. □

Recall that $\mathrm{NC}(\mathrm{CAY}(G, S)) = 1$ if $\mathrm{SCAY}(G, S)$ is empty, complete, or disconnected. It is clear that the survival subgraph $\mathrm{SCAY}(G, S)$ is empty if and only if $S = G \setminus \{1\}$. Furthermore, it is easy to see that $(G \setminus S_1)(G \setminus S_1) \subset S_1$ is a necessary and sufficient condition for the survival subgraph to be complete. In the remainder of the paper, therefore, we investigate when $\mathrm{SCAY}(G, S)$ is disconnected.

3. Necessary and sufficient conditions for the survival subgraph to have isolated vertices. In this section we deal with the case in which $\mathrm{SCAY}(G, S)$ has an isolated vertex. We first characterize the generating set of a Cayley graph whose survival subgraph has an isolated vertex.

THEOREM 2. *Let* $\mathrm{CAY}(G, S)$ *be a Cayley graph with $v \in G \setminus S_1$. Then the following statements are equivalent:*

(a) *the element v is an isolated vertex in* $\mathrm{SCAY}(G, S)$*;*

(b) *S is left $\langle v \rangle$-periodic;*

(c) *the nonidentity elements of $\langle v \rangle$ are isolated vertices of* $\mathrm{SCAY}(G, S)$.

Proof. For (a) ⇒ (b), let v be an isolated vertex of $\mathrm{SCAY}(G, S)$. Then $vS = N(v) \subset S$. Using Lemma 1(c), we conclude that $\langle v \rangle S = S$. Hence S is left $\langle v \rangle$-periodic.

For (b) ⇒ (c), let S be left $\langle v \rangle$-periodic. Since $1 \notin S$, Lemma 1(b) implies $\langle v \rangle \cap S = \emptyset$, and so each nonidentity element of $\langle v \rangle$ is a vertex of $\mathrm{SCAY}(G, S)$. The left $\langle v \rangle$-periodicity of S gives $\langle v \rangle S = S$. From these two facts we conclude that each nonidentity element of $\langle v \rangle$ is an isolated vertex in $\mathrm{SCAY}(G, S)$.

The implication (c) ⇒ (a) is immediate. □

COROLLARY 2.1. *Let D be the set of all isolated vertices in* $\mathrm{SCAY}(G, S)$. *The*

set $D \cup \{1\}$ is a subgroup. Furthermore, S is left H-periodic if and only if H is a subgroup of $D \cup \{1\}$.

Proof. To show that $D \cup \{1\}$ is a subgroup, it is enough to verify that it is closed under multiplication. To see this, let v and w be elements of D. Since $S^{-1} = S$, we cannot have $v^{-1} \in S$. If $vw \in S$, the equation $v^{-1}(vw) = w$ indicates that an edge in $\mathrm{SCAY}(G, S)$ links v^{-1} and w, contradicting the assumption that w is isolated. Consequently, $vw \in \mathrm{SCAY}(G, S)$. Since both v and w are isolated vertices in $\mathrm{SCAY}(G, S)$, we have $vS = S$ and $wS = S$. But then $vwS = v(wS) = vS = S$, which shows that vw is an isolated vertex of $\mathrm{SCAY}(G, S)$. Thus $D \cup \{1\}$ is closed under multiplication and is a subgroup as claimed.

Let S be left H-periodic. Then $HS = S$, so each vertex of H is isolated in $\mathrm{SCAY}(G, S)$, establishing that $H \subset D \cup \{1\}$. To establish the converse, let $H \subset D \cup \{1\}$. Then by part (b) of Theorem 2, S is left $\langle h \rangle$-periodic for each $h \in H \setminus \{1\}$. It immediately follows that S is left H-periodic. □

COROLLARY 2.2. *If S is left periodic, then* $\mathrm{NC}(\mathrm{CAY}(G, S)) = 1$.

Proof. Since S is left periodic, Corollary 2.1 guarantees that $\mathrm{SCAY}(G, S)$ has isolated vertices. Thus $D \cup \{1\}$ is a nontrivial subgroup of G. If the cardinality of $D \cup \{1\}$ is two, then $\mathrm{SCAY}(G, S)$ has only one isolated vertex, and $\mathrm{NC}(\mathrm{CAY}(G, S)) = 1$ whether or not $\mathrm{SCAY}(G, S)$ has another component. If the cardinality of $D \cup \{1\}$ is at least three, then $\mathrm{SCAY}(G, S)$ has at least two isolated vertices and is disconnected. Therefore, the neighbor connectivity of $\mathrm{CAY}(G, S)$ is one in all cases. □

4. Sufficient conditions for disconnectedness when the survival subgraph has no isolated vertices. In the previous section, we established that the presence of isolated vertices in the survival subgraph of $\mathrm{CAY}(G, S)$ is equivalent to the periodicity of S for a subgroup contained in $G \setminus S$. In the rest of the paper, we investigate disconnectedness in $\mathrm{SCAY}(G, S)$ when there are no isolated vertices or, equivalently, when S is not periodic for any subgroup contained in $G \setminus S$. In this section, we introduce the concept of near periodicity for generating sets and show it is equivalent to the existence of components whose vertex set does not generate G. We then describe the concept of an irreducible Cayley graph and show that in such graphs, either of the above two equivalent conditions is sufficient for the survival subgraph to be disconnected.

4.1. Nongenerating components and near periodicity. The following example illustrates a generic situation that motivates our next pair of definitions: let $G = (\mathbf{Z}_{20}, +)$ and, shifting to additive notation, let $H = \langle 5 \rangle$ and $S = (H+1) \cup (H+5)$. Then S is H-periodic and the isolated vertices of $\mathrm{SCAY}(G, S)$ are the set $H \setminus \{0\}$. In addition to the isolated vertices, $\mathrm{SCAY}(G, S)$ has a single connected component whose vertex set is $(H + 2) \cup (H + 3)$.

Suppose now that we enlarge S by adding the isolated vertex 10 to S, forming $S^* = (H + 1) \cup (H + 5) \cup \{10\}$. (Since $10 \equiv -10 \pmod{20}$, we have $S^* = -S^*$ as required.) What happens is that $\mathrm{SCAY}(G, S^*)$ becomes disconnected with no isolated vertices. The formerly isolated vertices 5 and 15 are now joined by an edge corresponding to the new generator 10 and form a connected component of $\mathrm{SCAY}(G, S^*)$. The component of $\mathrm{SCAY}(G, S)$ whose vertices are $(H + 2) \cup (H + 3)$ is still a component of $\mathrm{SCAY}(G, S^*)$, although it now has more edges. S^* is no longer periodic—it is now a union of cosets of H together with part, but not all, of H. The new component with vertex set $\{5, 15\}$ is contained in H and so is a nongenerating component in the sense that its vertex set does not generate G.

These observations provide some motivation for making the following definitions. A component C of a subgraph of $\mathrm{CAY}(G,S)$ is called a *nongenerating component* if and only if $\langle C\rangle \neq G$. A subset $Y \subset G$ is called *nearly H-periodic* (or just *nearly periodic* if we do not need to refer to H) if H is a nontrivial subgroup of G, $Y \setminus H$ is left H-periodic, $H \setminus \{1\} \not\subset Y$, and $Y \cap H \setminus \{1\} \neq \emptyset$. The next proposition establishes that, in the absence of isolated vertices in $\mathrm{SCAY}(G,S)$, the two concepts defined above are equivalent. This equivalence will be fundamental to our analysis in §5 of the near periodicity of S when $\mathrm{SCAY}(G,S)$ is disconnected with no isolated vertices.

PROPOSITION 3. *Let* $\mathrm{CAY}(G,S)$ *be a Cayley graph for which* $\mathrm{SCAY}(G,S)$ *has no isolated vertices. The survival subgraph* $\mathrm{SCAY}(G,S)$ *has a nongenerating component if and only if S is nearly periodic.*

Proof. Let C be a nongenerating component of $\mathrm{SCAY}(G,S)$; this means $\langle C\rangle \neq G$, and $\langle C\rangle$ contains at least one nonidentity vertex outside S. Since C is not an isolated vertex, $\langle C\rangle$ contains at least one generator in S. Thus in order to conclude that S is nearly $\langle C\rangle$-periodic, it remains to show that $S \setminus \langle C\rangle$ is left $\langle C\rangle$-periodic. Since S generates G and C does not, $S \setminus \langle C\rangle \neq \emptyset$. In order to establish that $\langle C\rangle\,(S \setminus \langle C\rangle) \subset S \setminus \langle C\rangle$, let $s \in S \setminus \langle C\rangle$. Then $Cs \cap \langle C\rangle = \emptyset$. Since C is a component, we have $CS \subset C \cup S$. We may conclude, therefore, that $Cs \subset (C \cup S) \setminus \langle C\rangle = S \setminus \langle C\rangle$. Thus $C(S \setminus \langle C\rangle) \subset S \setminus \langle C\rangle$. Therefore, by Lemma 1(c), $S \setminus \langle C\rangle$ is left $\langle C\rangle$-periodic.

Conversely, suppose S is nearly periodic. Then there is a nontrivial subgroup $H < G$ such that $H \setminus \{1\} \not\subset S$, $S \cap H \setminus \{1\} \neq \emptyset$, and $S \setminus H$ is left H-periodic. We claim that $H \setminus S_1$, which is nonempty by assumption, is a union of components of $\mathrm{SCAY}(G,S)$. Since the vertices of $H \setminus S_1$ can generate no more than H, the claim implies that $H \setminus S_1$ is a union of nongenerating components. To establish this claim we decompose $(H \setminus S_1)S$ as the union of two terms:

$$(H \setminus S_1)S = (H \setminus S_1)(S \cap H) \cup (H \setminus S_1)(S \setminus H).$$

Since H is a subgroup, the first term of the union is contained in H. Since $S \cap H = (S \cap H)^{-1}$, 1 does not belong to the first term of the union, thus that term is contained in $H \setminus \{1\}$. Because $S \setminus H$ is left H-periodic, $(H \setminus S_1)(S \setminus H) \subset S \setminus H$. Since $S \setminus H$ is finite, a simple cardinality argument implies that $(H \setminus S_1)(S \setminus H) = S \setminus H$. Hence the second term of the union is $S \setminus H$. Thus

$$(H \setminus S_1)S \subset (H \setminus \{1\}) \cup (S \setminus H) = (H \cup S) \setminus \{1\} = (H \setminus S_1) \cup S,$$

and we conclude by Lemma 1(e) that $H \setminus S_1$ is a union of necessarily nongenerating components. □

Remark. Proposition 3 is phrased to coordinate with our plan of attack, which excludes the isolated vertex case already dealt with in §3. However, the proof of Proposition 3 does not require the assumption that $\mathrm{SCAY}(G,S)$ has no isolated vertices. What we actually proved is, first, that if $\mathrm{SCAY}(G,S)$ has a nongenerating component C that is not an isolated vertex, then S is left $\langle C\rangle$-periodic and, second, that if S is left H-periodic, then $H \setminus S_1$ is a union of nongenerating components (some of which could be isolated vertices).

4.2. Irreducible Cayley graphs. We have seen that the periodicity of the generating set S is sufficient (and necessary) for the presence of isolated vertices in $\mathrm{SCAY}(G,S)$ and that the near periodicity of S is sufficient for the existence of a nongenerating component in $\mathrm{SCAY}(G,S)$. One might hope that the existence of a nongenerating component would imply that $\mathrm{SCAY}(G,S)$ is disconnected, in which case the near periodicity of S would imply that $\mathrm{SCAY}(G,S)$ is disconnected. It is

possible, however, for $\mathrm{SCAY}(G, S)$ to consist of a single component which is, nonetheless, nongenerating.

For example, let $G = (\mathbf{Z}_{25}, +)$, $H = \langle 5 \rangle$, and $S = (G \setminus H) \cup \{10, 15\}$. Then $\mathrm{SCAY}(G, S)$ consists of a single component with vertices 5 and 20 joined by the edge corresponding to ± 15. This is a nongenerating component, since its vertex set is contained in H. Hence we have $\mathrm{SCAY}(G, S)$ connected even though S is nearly H-periodic.

To analyze the situation described above, we must consider subgraphs of $\mathrm{CAY}(G, S)$ of the form $\mathrm{CAY}(H, H \cap S)$, where H is a subgroup of G. Since $H \cap S$ may not generate H, we must use a broader definition of Cayley graphs in order to include these new examples. The new more inclusive definition is exactly the same as the definition given in §2, except that we do not automatically require, when we write $\mathrm{CAY}(G, T)$, that T generates G. The nature of $\mathrm{CAY}(G, T)$ when T does not necessarily generate G is well known:

(1) $\mathrm{CAY}(G, T)$ is connected if and only if T generates G. In this case, we refer to $\mathrm{CAY}(G, T)$ as a *connected Cayley graph.*

(2) If T does not generate G, then $\mathrm{CAY}(G, T)$ is a disjoint union of $[G : H]$ connected components, each of which is isomorphic to $\mathrm{CAY}(H, T)$, where $H = \langle T \rangle$. We refer to these new examples as *disconnected Cayley graphs.*

Although there is some risk of confusion, it is conventional, when dealing with a disconnected Cayley graph, to refer to the set T as a set of generators and to the elements of T as generators, even though T generates a proper subgroup of G. Having now broadened our definition, we stress that throughout this paper, $\mathrm{CAY}(G, S)$ always denotes a connected Cayley graph. The only possibly disconnected Cayley graphs we encounter are subgraphs of connected Cayley graphs $\mathrm{CAY}(G, S)$ that have the form $\mathrm{CAY}(H, H \cap S)$ for some subgroup $H < G$.

In the situation described at the beginning of this section, $\mathrm{SCAY}(G, S)$ has a nearly periodic generating set and consists of a single nongenerating component. This condition can occur whenever G has a proper subgroup H for which $G \setminus S_1 = H \setminus S_1$. ($H = \langle 5 \rangle$ in the example given above.) If such a subgroup exists, then as we observe in Lemma 4(b), $\mathrm{SCAY}(G, S) = \mathrm{SCAY}(H, H \cap S)$ and so the connectivity of $\mathrm{SCAY}(G, S)$ is determined only by the relationship of $H \cap S$ to H. Regardless of what this relationship dictates, $S \setminus H$ is always a union of all right cosets of H in G except for H itself. In this case, then, S will be nearly H-periodic whether or not $\mathrm{SCAY}(G, S)$ is connected.

We can ensure that the presence of a nongenerating component implies that $\mathrm{SCAY}(G, S)$ is disconnected if we outlaw the situation described above by restricting our attention to Cayley graphs in which $G \setminus S_1 \neq H \setminus S_1$ for any proper subgroup H. We call a Cayley graph with this property *irreducible.* Otherwise, if $G \setminus S_1 = H \setminus S_1$ for some proper subgroup H, we call the Cayley graph *reducible.* If $\mathrm{CAY}(G, S)$ is reducible, we can replace it with the subgraph $\mathrm{CAY}(H, H \cap S)$ which has the identical survival subgraph. We observe in Lemma 4(b) that whenever $\mathrm{CAY}(G, S)$ is reducible, H can be chosen so that $\mathrm{CAY}(H, H \cap S)$ is irreducible. Consequently, everything that contributes to the component structure of $\mathrm{SCAY}(G, S)$ is encoded in the periodicity structure of $H \cap S$ and the relationship between $H \cap S$ and H; the parts of S and G outside of H are irrelevant.

These observations determine the following strategy: we study irreducible Cayley graphs because their generating sets contain no elements irrelevant to the component structure of the survival subgraph. When confronted with an arbitrary Cayley graph

$\mathrm{CAY}(G, S)$, we replace it, if it is reducible, with the irreducible subgraph $\mathrm{CAY}(H, H \cap S)$, which has the identical survival subgraph. We ultimately will show (Theorem 16) that we can use our results on all connected Cayley graphs with normal generating sets whether or not the graph is reducible.

LEMMA 4.

(a) *The Cayley graph* $\mathrm{CAY}(G, S)$ *is irreducible if and only if* $\langle G \setminus S \rangle = G$.

(b) *If* $\mathrm{CAY}(G, S)$ *is reducible, set* $H = \langle G \setminus S \rangle$. *Then* $\mathrm{CAY}(H, H \cap S)$ *is irreducible and* $\mathrm{SCAY}(G, S) = \mathrm{SCAY}(H, H \cap S)$.

Proof. For (a), we prove the contrapositive equivalence. Let $H = \langle G \setminus S \rangle$ and suppose that $H \neq G$. It is easy to check that

$$G \setminus S_1 = \langle G \setminus S \rangle \setminus S_1 = H \setminus S_1, \tag{1}$$

and so $\mathrm{CAY}(G, S)$ is reducible. Conversely, if $\mathrm{CAY}(G, S)$ is reducible, then $\langle G \setminus S \rangle = \langle H \setminus S \rangle \subset H$ for a proper subgroup $H < G$. Thus $\langle G \setminus S \rangle \subset H \neq G$.

For (b), we first show that $\mathrm{CAY}(H, H \cap S)$ is irreducible. According to part (a), it suffices to check that $\langle H \setminus S \rangle = H$. Recalling what H is, we must verify that $\langle \langle G \setminus S \rangle \setminus S \rangle = \langle G \setminus S \rangle$, and this is elementary. To establish the equality of the survival subgraphs, note first that (1) above shows that the two subgraphs have the same vertex set. Moreover, the generating set $H \cap S$ for $\mathrm{CAY}(H, H \cap S)$ is precisely the set of generators that correspond to edges having both their vertices in the common vertex set. □

Although $\mathrm{SCAY}(G, S)$ is always equal to $\mathrm{SCAY}(H, H \cap S)$ for $H = \langle G \setminus S \rangle$, the graph $\mathrm{CAY}(H, H \cap S)$ may be a disconnected Cayley graph and so will be disconnected *before* subversion. The next lemma indicates that in this case, $\mathrm{NC}(\mathrm{CAY}(G, S)) = 1$. This is not immediately obvious, because we have to address the possibility that $\mathrm{CAY}(H, H \cap S)$ might consist of two components and subversion could eliminate the component containing 1, leaving a single component. Indeed, this can happen, but when it does we observe that the single remaining component is a clique, so that we still have neighbor connectivity equal to one.

LEMMA 5. *Let* $\mathrm{CAY}(G, S)$ *be a connected Cayley graph and set* $H = \langle G \setminus S \rangle$. *If the irreducible subgraph* $\mathrm{CAY}(H, H \cap S)$ *is a disconnected Cayley graph, then* $\mathrm{SCAY}(G, S)$ *is disconnected or complete, and so* $\mathrm{NC}(\mathrm{CAY}(G, S)) = 1$.

Proof. $\mathrm{CAY}(H, H \cap S)$ is a union of components, each of which has a vertex set equal to a right coset of $\langle H \cap S \rangle$ in H and each of which is isomorphic to $\mathrm{CAY}(\langle H \cap S \rangle, H \cap S)$. Since subversion of $\{1\}$ affects only the component of $\mathrm{CAY}(H, H \cap S)$ containing 1, we have a disjoint union

$$\mathrm{SCAY}(H, H \cap S) \cong \mathrm{SCAY}(\langle H \cap S \rangle, H \cap S) \cup \bigcup_{\tau} \mathrm{CAY}(\langle H \cap S \rangle, H \cap S), \tag{2}$$

where τ is the set of all nonidentity right cosets of $\langle H \cap S \rangle$ in H. According to Lemma 4, $\mathrm{SCAY}(G, S) = \mathrm{SCAY}(H, H \cap S)$. If $\mathrm{SCAY}(H, H \cap S)$ is disconnected, we are finished. If $\mathrm{SCAY}(H, H \cap S)$ is connected, the union in (2) contains exactly one coset. Since subversion of $\{1\}$ affects only the component of $\mathrm{CAY}(H, H \cap S)$ containing 1, it follows that $\mathrm{CAY}(H, H \cap S)$ consists of exactly two $\langle H \cap S \rangle$-cosets: $\langle H \cap S \rangle$ itself and one nonidentity right coset. Moreover, every element of $\langle H \cap S \rangle$ must be deleted upon subversion of $\{1\}$; hence, $\langle H \cap S \rangle \subset S_1$. Consequently, all the vertices of $\mathrm{SCAY}(G, S) = \mathrm{SCAY}(H, H \cap S)$ are in the only nonidentity right $\langle H \cap S \rangle$-coset in H, and these vertices induce a complete graph. □

In view of Lemma 5, we are left to consider the case in which the irreducible

subgraph $\mathrm{CAY}(H, H \cap S)$ is connected. Specifically, the question of the connectedness of $\mathrm{SCAY}(G, S)$ has been reduced to the question of the connectedness of $\mathrm{SCAY}(H, H \cap S)$ where $H = \langle G \setminus S \rangle$ and $\mathrm{CAY}(H, H \cap S)$ is an irreducible connected Cayley subgraph of $\mathrm{CAY}(G, S)$. In the rest of the paper, we address the question of the connectivity of $\mathrm{SCAY}(G, S)$ when $\mathrm{CAY}(G, S)$ is an irreducible Cayley graph. However, some of the results that we develop to elucidate this question do not require the assumption of irreducibility, and so we do not impose it uniformly.

4.3. Sufficiency results for irreducible Cayley graphs.

LEMMA 6. *If* $\mathrm{CAY}(G, S)$ *is irreducible and* $\mathrm{SCAY}(G, S)$ *has a nongenerating component, then* $\mathrm{SCAY}(G, S)$ *is disconnected.*

Proof. The proof is by contradiction. Suppose $\mathrm{SCAY}(G, S)$ is connected. Then the nongenerating component C is the only component of $\mathrm{SCAY}(G, S)$. Hence $\langle C \rangle \setminus S = G \setminus S$. Since $\langle C \rangle \neq G$, G is reducible. □

THEOREM 7. *Let* $\mathrm{CAY}(G, S)$ *be an irreducible Cayley graph. If* S *is nearly periodic, then* $\mathrm{SCAY}(G, S)$ *is disconnected.*

Proof. Since S is nearly periodic, we conclude from Proposition 3 that $\mathrm{SCAY}(G, S)$ has a nongenerating component. According to Lemma 6, this means that $\mathrm{SCAY}(G, S)$ is disconnected, since $\mathrm{CAY}(G, S)$ is irreducible. □

5. Necessary conditions for disconnectedness when the survival subgraph has no isolated vertices. Beginning with a disconnected survival subgraph with no isolated vertices, our goal is to show that we must have a nearly periodic generating set. This means we need a special subset of generators that will generate a proper subgroup for which the generating set is nearly periodic. Equivalently, Proposition 3 allows us to search for a nongenerating component of the survival subgraph. We use both these equivalent approaches below.

5.1. Invisible generators. In this section we show how to use components of the disconnected survival subgraph to pick out special subsets of generators, called invisible generators, that will generate proper subgroups for which the generating set is nearly periodic. Define the set of *generators invisible from* C as $I_C = S \setminus N(C)$. In Proposition 9 we show, under our standing hypotheses of no isolated vertices, that $\mathrm{SCAY}(G, S)$ always has components with invisible generators. In Proposition 13 we show that the existence of invisible generators leads to nongenerating components in $\mathrm{SCAY}(G, S)$ for a large class of Cayley graphs including all those with abelian G, thus establishing the converse of Theorem 7 for that class of graphs.

We begin with a technical lemma.

LEMMA 8. *Let* $X \subset G$ *have the property that* $N(X) = S$ *in* $\mathrm{CAY}(G, S)$. *If there is a* $g \in G$ *such that* $gX = X$, *then* g *is an isolated vertex of* $\mathrm{SCAY}(G, S)$.

Proof. Let g and X be as stated in the hypothesis.

$$
\begin{aligned}
gX = X &\Longrightarrow \langle g \rangle X = X \quad \text{by Lemma 1(c)},\\
&\Longrightarrow \langle g \rangle N(X) = N(X) \quad \text{by Lemma 1(d)},\\
&\Longrightarrow \langle g \rangle S = S \quad \text{since } N(X) = S,\\
&\Longrightarrow \langle g \rangle \cap S = \emptyset \quad \text{by Lemma 1(b)},\\
&\Longrightarrow g \text{ is an isolated vertex of } \mathrm{SCAY}(G, S). \qquad \square
\end{aligned}
$$

We shall call a component C of $\mathrm{SCAY}(G, S)$ a *maximum component* if, in the set of all components of $\mathrm{SCAY}(G, S)$, C has maximum vertex cardinality. The relevance

of maximum components is indicated by the following proposition.

PROPOSITION 9. *Let* $\mathrm{CAY}(G,S)$ *be a Cayley graph for which* $\mathrm{SCAY}(G,S)$ *is disconnected and has no isolated vertices. Then any maximum component has invisible generators in* S.

Proof. Let C be a maximum component and suppose that C has no invisible generators in S. We shall obtain a contradiction. Since $\mathrm{SCAY}(G,S)$ is disconnected, we may let g be a vertex of $\mathrm{SCAY}(G,S)$ that lies outside of C. (To refer to such vertices, we use the shorthand $g \notin C \cup S_1$.) Then $g^{-1}C \cap S_1 = \emptyset$, because if $g^{-1}c = s$ or $g^{-1}c = 1$ with $c \in C$, then $c = gs$ or $c = g$, contradicting $g \notin C$. Thus each element of $g^{-1}C$ is a vertex of $\mathrm{SCAY}(G,S)$. Whenever c_1 and c_2 are adjacent vertices of C, $g^{-1}c_1$ and $g^{-1}c_2$ are adjacent vertices of $g^{-1}C$. Consequently, $g^{-1}C$ is a connected subset of $\mathrm{SCAY}(G,S)$ of maximum vertex cardinality and so must be a maximum component of $\mathrm{SCAY}(G,S)$.

Consider the set of all components of the form $g^{-1}C$ as g ranges over $G \setminus (C \cup S_1)$. In light of Lemma 8, $g^{-1}C \neq C$, so the vertices of the components $g^{-1}C$ are contained in $G \setminus (C \cup S_1)$. We claim it cannot be that $g^{-1}C \cap h^{-1}C = \emptyset$ for all $g, h \notin C \cup S_1$. Suppose this were so. Then, on the one hand, we would have

$$\left| \bigcup_{g \notin C \cup S_1} g^{-1}C \right| = |G \setminus (C \cup S_1)| \cdot |C|. \tag{3}$$

On the other hand, since each $g^{-1}C$ is a subgraph of $\mathrm{CAY}(G,S)$ disjoint from $C \cup S_1$, we have

$$\bigcup_{g \notin C \cup S_1} g^{-1}C \subset G \setminus (C \cup S_1),$$

which means that

$$\left| \bigcup_{g \notin C \cup S_1} g^{-1}C \right| \leq |G \setminus (C \cup S_1)|. \tag{4}$$

Since $|C| \geq 2$, (3) and (4) are incompatible. We conclude that $g^{-1}C \cap h^{-1}C = \emptyset$ cannot occur for every distinct $g, h \notin C \cup S_1$. Hence, there must be distinct $g, h \notin C \cup S_1$ such that $g^{-1}C \cap h^{-1}C \neq \emptyset$. Now $g^{-1}C$ and $h^{-1}C$ are components, so the only way they can intersect is if $g^{-1}C = h^{-1}C$. This gives us $g, h \notin C \cup S_1$ with $gh^{-1} \neq 1$ and $gh^{-1}C = C$. By Lemma 8, gh^{-1} is an isolated vertex of $\mathrm{SCAY}(G,S)$, contradicting the hypothesis and so proving that C must have invisible generators. □

5.2. Normal generating sets. The results of the preceding section guarantee that if $\mathrm{SCAY}(G,S)$ is disconnected with no isolated vertices, then $\mathrm{SCAY}(G,S)$ has a component with invisible generators. We want to show that, in most cases, S is nearly periodic with respect to the subgroup generated by these invisible generators. In order to do this, and hence to derive a partial converse of Theorem 7, we restrict our attention to a class of Cayley graphs that includes all those for which G is abelian.

For a finite group G and subset $X \subset G$, we use $\langle\langle X \rangle\rangle$ to denote the normal subgroup generated by X, i.e., the subgroup of G generated by all conjugates of elements of X. We call X a *normal subset* provided that $gXg^{-1} = X$ for all $g \in G$. In the remainder of the paper, we show that for the class of Cayley graphs $\mathrm{CAY}(G,S)$ for which S is a normal subset, $\mathrm{SCAY}(G,S)$ is disconnected with no isolated vertices only if S is nearly periodic.

For an example of a nonabelian group G with normal generating set S, take $G = A_n$, the alternating group on n elements, let $n \geq 3$, and take S to be the set of all 3-cycles.

In order to understand the importance of requiring S to be a normal subset of G, first recall that a group is abelian if and only if the inversion of elements is a group automorphism. For nonabelian groups, inversion is not a group automorphism, but Cayley graphs with normal generating sets are "almost abelian" in the sense specified by the following lemma.

LEMMA 10. *The inversion of elements in G induces a graph automorphism of* $\mathrm{CAY}(G,S)$ *if and only if S is a normal subset of G. Analogously, the inversion of elements in G induces a graph automorphism of* $\mathrm{SCAY}(G,S)$ *if and only if S is a normal subset of G.*

Proof. For $g \in G$, the generator $s \in S$ joins g to gs if and only if gsg^{-1} joins $(gs)^{-1}$ to g^{-1}, and so the statement for $\mathrm{CAY}(G,S)$ is true. Since $S = S^{-1}$, inversion preserves the generating set. Thus S_1 is closed under both inversion and conjugation; consequently, so is $G \setminus S_1$. □

COROLLARY 10.1. *If S is a normal generating set, then C is a component of* $\mathrm{SCAY}(G,S)$ *if and only if C^{-1} is also.*

The next lemma lists three algebraic properties of normal subsets.

LEMMA 11. *Let S be a normal subset of G in Cay(G,S) and let $H = \langle G \setminus S \rangle$. Then*

(a) *$H \cap S$ is a normal subset in the irreducible Cayley graph* $\mathrm{CAY}(H, H \cap S)$;

(b) *for any $T \subset G$, $TS_1 = S_1$ implies $\langle\langle T \rangle\rangle S_1 = S_1$;*

(c) *the group consisting of all isolated vertices of* $\mathrm{SCAY}(G,S)$ *together with* 1 *is a normal subgroup of G.*

Proof. Part (a) follows immediately from Lemma 4(b) and the observation that the intersection of a normal subset of G with a subgroup of G is a normal subset of G. Parts (b) and (c) are established by straightforward calculations using the definition of normal subset and the fact that $\langle\langle T \rangle\rangle$ is the subgroup of G generated by all conjugates of elements of T. □

5.3. Quotient Cayley graphs. In the course of the argument that follows, we find it useful to pass from a connected Cayley graph $\mathrm{CAY}(G,S)$ to a Cayley graph of G/H for a particular subgroup H that is normal in G. In the situation that arises, the generating set S is very well adapted to the factorization—S_1 is H-periodic with $H \subset S_1$. For the rest of this section, we assume that S_1 has this property.

Before considering the correspondence between the Cayley graph of G and the Cayley graph of G/H, it is useful to describe how the condition that S_1 is H-periodic with $H \subset S_1$ affects the structure of $\mathrm{CAY}(G,S)$ and $\mathrm{SCAY}(G,S)$:

(1) The H-periodicity of S_1 implies that whenever a vertex in Hx is adjacent to a vertex in Hy, every vertex of Hx is adjacent to every vertex of Hy. To see why, we can assume without loss of generality that x is adjacent to y. Then $x^{-1}y \in S$, which means that $Hx^{-1}y \subset S$ since S is H-periodic. The normality of H then yields $x^{-1}Hy \subset S$, so that for all h_1, $h_2 \in H$, h_1x is adjacent to h_2y using the edge corresponding to $x^{-1}h_1^{-1}h_2y \in S$.

(2) The fact that $H \subset S_1$ implies that, in both $\mathrm{CAY}(G,S)$ and $\mathrm{SCAY}(G,S)$, the vertices belonging to each coset of H induce a clique.

The two observations above imply that when S_1 is H-periodic with $H \subset S$, both $\mathrm{CAY}(G,S)$ and $\mathrm{SCAY}(G,S)$ are graphs whose vertex sets are partitioned (by H-cosets) into cliques with the property that between any two cliques there are either

no edges or a complete join. We may thus refer to two such cliques as being adjacent or nonadjacent without fear of ambiguity.

We now turn to the associated Cayley graph on G/H. Let $\phi\colon G \to G/H$ be the quotient homomorphism and let $T = \phi S \setminus \{1\}$. Then $T = T^{-1}$, $\phi S_1 = T_1$, and since S generates G, T must generate $\phi G = G/H$. Hence we may speak of the Cayley graph $\mathrm{CAY}(\phi G, T)$. Its vertices correspond to the cosets of H in G and it is connected if and only if $\mathrm{CAY}(G, S)$ is connected. Since S_1 is H-periodic, we also have $\phi^{-1}T_1 = S_1$ and so $\phi(G \setminus S_1) = \phi G \setminus T_1$. Consequently, ϕ induces maps of graphs $\mathrm{CAY}(\phi)\colon \mathrm{CAY}(G, S) \to \mathrm{CAY}(\phi G, T)$ and $\mathrm{SCAY}(\phi)\colon \mathrm{SCAY}(G, S) \to \mathrm{SCAY}(\phi G, T)$. (The effect of $\mathrm{CAY}(\phi)$ and $\mathrm{SCAY}(\phi)$ on both vertices and edges is precisely the effect of ϕ.)

$\mathrm{CAY}(\phi)$ collapses each clique in $\mathrm{CAY}(G, S)$ induced by a coset of H to a vertex in $\mathrm{CAY}(\phi G, T)$. Whenever two H-cliques in $\mathrm{CAY}(G, S)$ are adjacent, the vertices they collapse to in $\mathrm{CAY}(\phi G, T)$ are adjacent, and whenever two H-cliques in $\mathrm{CAY}(G, S)$ are nonadjacent, the vertices they collapse to in $\mathrm{CAY}(\phi G, T)$ are nonadjacent. This description indicates that we can recover $\mathrm{CAY}(G, S)$ from $\mathrm{CAY}(\phi G, T)$ by the following procedure: replace each vertex of $\mathrm{CAY}(\phi G, T)$ by a clique on $|H|$ vertices and link two such cliques by a complete join whenever the vertices they replaced are adjacent. In other words, $\mathrm{CAY}(G, S) = \mathrm{CAY}(\phi G, T)[K_{|H|}]$, the wreath product (or composition or lexicographic product) of graphs $\mathrm{CAY}(\phi G, T)$, and $K_{|H|}$, where K_n denotes the clique on n vertices. Exactly the same reasoning shows that $\mathrm{SCAY}(G, S) = \mathrm{SCAY}(\phi G, T)[K_{|H|}]$.

The wreath product structure of $\mathrm{SCAY}(G, S)$ makes it evident that each component C of $\mathrm{SCAY}(G, S)$ is of the form $C = D[K_{|H|}] = \mathrm{SCAY}(\phi)^{-1}D$, where D is a component of $\mathrm{SCAY}(\phi G, T)$. Thus $\mathrm{SCAY}(\phi)$ induces a one-to-one correspondence of the components of $\mathrm{SCAY}(G, S)$ and $\mathrm{SCAY}(\phi G, T)$.

It is easy to see that nongenerating components of $\mathrm{SCAY}(\phi G, T)$ are preserved by $\mathrm{SCAY}(\phi)^{-1}$: since ϕ is a homomorphism, we have $\phi\langle X\rangle = \langle \phi X\rangle$ for any subset $X \subset G$. Thus if the vertices of a component C generate G, then the vertices of the component ϕC will generate ϕG. Consequently, if D is a nongenerating component of $\mathrm{SCAY}(\phi G, T)$, its inverse image $\mathrm{SCAY}(\phi)^{-1}D$ is a nongenerating component of $\mathrm{CAY}(G, S)$.

Finally, suppose C is a component of $\mathrm{SCAY}(G, S)$ with invisible generators I_C such that $\langle\langle I_C\rangle\rangle \subset S_1$ and S_1 is $\langle\langle I_C\rangle\rangle$-periodic. The normal subgroup H in the above discussion is now $\langle\langle I_C\rangle\rangle$. Then the component $\mathrm{SCAY}(\phi)C$ of $\mathrm{SCAY}(\phi G, T)$ has no invisible generators in T, because such an invisible generator would give rise to an entire coset of generators invisible from C in $\mathrm{CAY}(G, S)$.

These results are summarized in the following lemma.

LEMMA 12. *Let* $\mathrm{CAY}(G, S)$ *be a connected Cayley graph for which* H *is a normal subgroup of* G, $H \subset S_1$, *and* S_1 *is* H*-periodic. Further, let* $\phi\colon G \to G/H$ *be the quotient homomorphism and let* $T = \phi S \setminus \{1\}$.

(a) *Then* T *is a generating set for* ϕG, *so that* $\mathrm{CAY}(\phi G, T)$ *is a connected Cayley graph.*

(b) *The homomorphism* ϕ *induces a one-to-one correspondence between components of* $\mathrm{SCAY}(G, S)$ *and components of* $\mathrm{SCAY}(\phi G, T)$.

(c) *If* D *is a nongenerating component of* $\mathrm{SCAY}(\phi G, T)$, *then* $\mathrm{SCAY}(\phi)^{-1}D$ *is a nongenerating component of* $\mathrm{SCAY}(G, S)$.

(d) *If* $H = \langle\langle I_C\rangle\rangle$, *where* I_C *is a set of generators invisible from a component* C *of* $\mathrm{SCAY}(G, S)$, *then the component* $\mathrm{SCAY}(\phi)C$ *of* $\mathrm{SCAY}(\phi G, T)$ *has no invisible*

generators in T.

5.4. Necessity results for Cayley graphs with normal generating sets. Proposition 9 guarantees that when the survival subgraph is disconnected without isolated vertices, there will be components from which some generators are invisible. We now have all the tools we need to show that when the generating set is normal, the presence of invisible generators ensures that the generating set will be nearly periodic. In Proposition 13, we obtain near periodicity in some cases. The remaining cases are handled in Theorem 14, which summarizes all the cases in the language of nongenerating components.

PROPOSITION 13. *Let* $\mathrm{CAY}(G,S)$ *have a normal generating set* S*, and let* $\mathrm{SCAY}(G,S)$ *be disconnected with no isolated vertices. Suppose* C *is a component of* $\mathrm{SCAY}(G,S)$ *that has invisible generators* $I_C \subset S$*. Then*

(a) $N(C)$ *is left* $\langle I_C\rangle$*-periodic with* $\langle I_C\rangle \cap N(C) = \emptyset$*;*

(b) *if* $\langle I_C\rangle \not\subset S_1$*, then* S *is nearly* $\langle I_C\rangle$*-periodic;*

(c) *if* $\langle I_C\rangle \subset S_1$*, then* S_1 *is left* $\langle\langle I_C\rangle\rangle$*-periodic with* $\langle\langle I_C\rangle\rangle \subset S_1$*.*

Proof. We begin by proving (a). Note that ${I_C}^{-1}C \cap S = \emptyset$; otherwise, there would be $c \in C$, $s \in I_C$, and $s_0 \in S$ such that $s^{-1}c = s_0$. But then $cs_0{}^{-1} = s$, which contradicts the fact that $s \in I_C$.

Using the fact that $S^{-1} = S$, it follows immediately that ${I_C}^{-1}C \cap S = \emptyset$ if and only if $C^{-1}I_C \cap S = \emptyset$. Since S is a normal generating set, we know from Corollary 10.1 that C^{-1} is a component of $\mathrm{SCAY}(G,S)$ and so

$$\begin{aligned} C^{-1}I_C &\subset C^{-1}S \\ &\subset C^{-1} \cup S \quad \text{by Lemma 1(e).} \end{aligned}$$

Combining this with $C^{-1}I_C \cap S = \emptyset$, we get $C^{-1}I_C \subset C^{-1}$, and

$$\begin{aligned} C^{-1}I_C \subset C^{-1} &\Longrightarrow C^{-1}I_C = C^{-1}, \\ &\Longrightarrow {I_C}^{-1}C = C, \\ &\Longrightarrow C \text{ is left } \langle I_C\rangle\text{-periodic by Lemma 1(c)}, \\ &\Longrightarrow N(C) \text{ is left } \langle I_C\rangle\text{-periodic by Lemma 1(d).} \end{aligned}$$

Since the generators I_C are disjoint from $N(C)$ by definition, Lemma 1(b) indicates that $\langle I_C\rangle \cap N(C) = \emptyset$, and this completes part (a).

For part (b), the definition of near periodicity requires that we show three things: $S \setminus \langle I_C\rangle$ is $\langle I_C\rangle$-periodic, $\langle I_C\rangle \setminus \{1\} \not\subset S$, and $S \cap \langle I_C\rangle \setminus \{1\} \neq \emptyset$. The second condition has been imposed by the hypothesis for part (b), and the third condition is true because $I_C \subset S \cap \langle I_C\rangle \setminus \{1\}$.

As for the first condition, begin by recalling that S is a disjoint union $S = N(C) \cup I_C$ by definition of I_C. We have $\langle I_C\rangle \cap N(C) = \emptyset$ from part (a). Consequently, $S \setminus \langle I_C\rangle = S \setminus I_C = N(C)$, so $S \setminus \langle I_C\rangle$ is $\langle I_C\rangle$-periodic by part (a). This completes the verification for part (b).

For part (c), the conditions $S_1 = N(C) \cup I_C \cup \{1\}$ and $\langle I_C\rangle \cap N(C) = \emptyset$, when combined with the assumption that $\langle I_C\rangle \subset S_1$, imply that $\langle I_C\rangle = I_C \cup \{1\}$. Hence, we have the disjoint union $S_1 = N(C) \cup \langle I_C\rangle$. In light of part (a), S_1 is $\langle I_C\rangle$-periodic, and Lemma 11(b) implies that S_1 must be $\langle\langle I_C\rangle\rangle$-periodic. Since $\langle\langle I_C\rangle\rangle \cap S_1 \neq \emptyset$, Lemma 1(b) indicates that $\langle\langle I_C\rangle\rangle \subset S_1$. □

THEOREM 14. *Let* $\mathrm{CAY}(G,S)$ *be a connected Cayley graph for which* S *is a normal subset,* $\mathrm{SCAY}(G,S)$ *is disconnected with no isolated vertices, and* C *is a max-*

imum component of $\mathrm{SCAY}(G,S)$. *Then* $\mathrm{SCAY}(G,S)$ *has a nongenerating component* C', *and* S *is nearly* $\langle C'\rangle$*-periodic. In fact,* C' *can be specified further:*

(a) *if* $\langle I_C\rangle \not\subset S_1$, $C' \subset \langle I_C\rangle \setminus S_1$;

(b) *if* $\langle I_C\rangle \subset S_1$, *then* C' *is a clique.*

Proof. Since C is a maximum component of $\mathrm{SCAY}(G,S)$, Proposition 9 indicates that C has invisible generators $I_C \subset S$.

In case (a), we have $\langle I_C\rangle \not\subset S_1$. According to Proposition 13(b), S must be nearly $\langle I_C\rangle$-periodic. By the proof of Proposition 3, $\langle I_C\rangle \setminus S_1$ is a union of nongenerating components, and we can take C' to be any one of them.

In case (b), we have $\langle I_C\rangle \subset S_1$. Using Proposition 13(c) and Lemma 12(a), we factor G by $\langle\langle I_C\rangle\rangle$ and let $\phi\colon G \to G/\langle\langle I_C\rangle\rangle$ be the quotient map. We set $T = \phi S \setminus \{1\}$ and so obtain a Cayley graph $\mathrm{CAY}(\phi G, T)$ which, when subverted, has the same number of components as the original graph by Lemma 12(b). Applying Lemma 12(d), we conclude that the component $\mathrm{SCAY}(\phi)C$ has no invisible generators. Then $\mathrm{SCAY}(\phi)C$ is a maximum component of $\mathrm{SCAY}(\phi G, T)$ with no invisible generators, and this would contradict Proposition 9 if $\mathrm{SCAY}(\phi G, T)$ had no isolated vertices. Hence there must be an isolated vertex in $\mathrm{SCAY}(\phi G, T)$.

An isolated vertex v in $\mathrm{SCAY}(\phi G, T)$ cannot generate ϕG, because $\langle v\rangle$ must be contained in $\phi G \setminus T$ by Theorem 2, and we now check that $\phi G \setminus T$ is a proper subset of ϕG by verifying that $T = \emptyset$ is impossible: since T must generate ϕG by Lemma 12(a), the only way $T = \emptyset$ can occur is if $\phi G = 1$, and this is impossible because Proposition 13(c) assures us that $\langle\langle I_C\rangle\rangle$ cannot be all of G.

Thus the isolated vertex v in $\mathrm{SCAY}(\phi G, T)$ is a nongenerating component. Invoking Lemma 12(c), we conclude that $C' = \mathrm{SCAY}(\phi)^{-1}v$ is a nongenerating component of $\mathrm{SCAY}(G,S)$. Finally, since $\mathrm{SCAY}(\phi)^{-1}v$ is a coset of $\langle\langle I_C\rangle\rangle$ and since $\langle\langle I_C\rangle\rangle \subset S_1$, it follows that C' is a clique.

Thus, regardless of whether $\langle I_C\rangle \not\subset S_1$ or $\langle I_C\rangle \subset S_1$, $\mathrm{SCAY}(G,S)$ has a nongenerating component C' and therefore, according to Proposition 3, S is nearly $\langle C'\rangle$-periodic. □

6. Periodic structure of the generating set for disconnected survival subgraphs. Let $\mathrm{CAY}(G,S)$ be a connected Cayley graph. Here is the current state of our knowledge about $\mathrm{SCAY}(G,S)$ when it is disconnected:

(1) Theorem 2 gives a necessary and sufficient condition on S for $\mathrm{SCAY}(G,S)$ to have an isolated vertex, namely, that S be left periodic.

(2) Suppose that $\mathrm{SCAY}(G,S)$ has no isolated vertices.

(a) If $\mathrm{CAY}(G,S)$ is irreducible, Theorem 7 gives a sufficient condition on S for $\mathrm{SCAY}(G,S)$ to be disconnected, namely, that S be nearly periodic.

(b) If S is normal, Theorem 14 gives a necessary condition on S for $\mathrm{SCAY}(G,S)$ to be disconnected, again, that S be nearly periodic.

By combining all these hypotheses, we obtain in Theorem 15 an algebraic characterization of the generating sets of a large class of Cayley graphs with disconnected survival subgraphs. Corollary 15.1 establishes that a Cayley graph of prime order cannot be disconnected by the subversion of a single vertex.

THEOREM 15. *Let* $\mathrm{CAY}(G,S)$ *be an irreducible connected Cayley graph with normal generating set* S. *Then* $\mathrm{SCAY}(G,S)$ *is disconnected if and only if* S *is left periodic or nearly periodic.*

COROLLARY 15.1. *For an irreducible connected Cayley graph* $\mathrm{CAY}(G,S)$ *with normal generating set* S *and* $S_1 \neq G$, *let* p *be the smallest prime divisor of* $|G|$. *If* $|S| < p$, *then* $\mathrm{SCAY}(G,S)$ *is connected. In particular, if* $G = \mathbf{Z}_p$ *and* $S_1 \neq \mathbf{Z}_p$, *then*

SCAY($\mathbf{Z}_p, S$) *is connected.*

Proof. We assume that SCAY(G, S) is disconnected and derive a contradiction. By Theorem 15, S is either left H-periodic or nearly H-periodic for some nontrivial subgroup $H < G$. In either case, S must contain at least one coset of H. Hence $p \leq |H| \leq |S|$, and this contradicts the hypothesis $|S| < p$. □

In case G is abelian, there is a slightly improved version of Corollary 15.1.

COROLLARY 15.2. *For an irreducible connected Cayley graph* CAY(G, S) *with* G *abelian and* $S_1 \neq G$, *let* p *be the smallest prime divisor of* $|G|$. *If* $|S| \leq p$, *then* SCAY(G, S) *is connected.*

Proof. If $|S| < p$ then SCAY(G, S) is connected by Corollary 15.1, so all we must show here is that $|S| = p$ implies SCAY(G, S) connected. Again, we assume SCAY(G, S) is disconnected and obtain a contradiction. The same argument as we used above indicates that S must contain at least one coset of a nontrivial subgroup $H < G$, but now the fact that $|S| = p$ implies that S must be a single nonidentity coset of H. Since $S = S^{-1}$, this coset has order 2 in G/H, and so 2 divides $|G|$. This means, according to the definition of p, that $p = 2$, and so $|S| = 2$. Thus CAY(G, S) is a cycle and so SCAY(G, S) is connected—the desired contradiction. □

The final theorem is an extension of Theorem 15 to all connected Cayley graphs, irreducible or not, with normal generating sets. This extension relies on the fact, from Lemma 4, that in CAY(G, S), CAY($H, H \cap S$) is irreducible when $H = \langle G \setminus S \rangle$.

THEOREM 16. *Let* CAY(G, S) *be a connected Cayley graph with normal generating set for which* SCAY(G, S) *is neither empty nor complete, and set* $H = \langle G \setminus S \rangle$. *The following are equivalent:*

(a) NC(CAY(G, S)) = 1;

(b) SCAY(G, S) *is disconnected;*

(c) $H \cap S$ *is left periodic,* $H \cap S$ *is nearly periodic, or* $H \cap S_1$ *is a proper subgroup of* G.

Proof. (a)⇔(b): this equivalence follows immediately from the definition of neighbor connectivity and the hypothesis that SCAY(G, S) is neither empty nor complete.

(c)⇒(b): we begin with the following assumptions:

(i) $H \cap S$ is left periodic or nearly periodic, or

(ii) $H \cap S_1$ is a proper subgroup of G.

We must show that SCAY(G, S) is disconnected. Since SCAY(G, S) = SCAY($H, H \cap S$) by Lemma 4(b), it suffices to prove that SCAY($H, H \cap S$) is disconnected.

First observe that, regardless of whether either of conditions (i) or (ii) holds, if we ever find that CAY($H, H \cap S$) is disconnected, then we can conclude that SCAY($H, H \cap S$) is disconnected by Lemma 5 and the hypothesis that SCAY(G, S) is not complete.

Suppose condition (i) holds. There are two cases to consider. Case 1: CAY($H, H \cap S$) is disconnected. By the remark above, SCAY($H, H \cap S$) is disconnected. Case 2: CAY($H, H \cap S$) is connected. By Lemma 4(b), CAY($H, H \cap S$) is irreducible so SCAY($H, H \cap S$) is disconnected by Theorem 15.

Suppose condition (ii) holds. We claim that $\langle H \cap S \rangle \neq H$ and so CAY($H, H \cap S$) is disconnected. By the remark above, SCAY($H, H \cap S$) would then be disconnected. To see why $\langle H \cap S \rangle \neq H$, note that $\langle H \cap S \rangle = H \cap S_1$ since $H \cap S_1$ is a subgroup, so we are claiming that $H \cap S_1 \neq H$. The contrary assumption that $H \cap S_1 = H$ implies that $H \subset S_1$, and

$$\begin{aligned} H \subset S_1 &\Longrightarrow \langle G \setminus S \rangle \subset S_1 \\ &\Longrightarrow G \setminus S \subset S_1 \end{aligned}$$

$$\Longrightarrow G \setminus S = 1$$
$$\Longrightarrow \mathrm{CAY}(G, S) \text{ is complete}$$
$$\Longrightarrow \mathrm{SCAY}(G, S) = \emptyset,$$

and this contradicts the hypotheses. We have now established (c)⇒(b).

To show (b)⇒(c), let $\mathrm{SCAY}(G, S)$ be disconnected. In the case that $\mathrm{CAY}(G, S)$ is irreducible, Theorem 15 applies since $\mathrm{CAY}(G, S)$ is connected by hypothesis. Furthermore, the irreducibility of $\mathrm{CAY}(G, S)$ implies $H \cap S = G \cap S = S$, and so Theorem 15 allows us to conclude that $H \cap S$ is left periodic or periodic. In the case that $\mathrm{CAY}(G, S)$ is reducible, then Lemma 4 states $\mathrm{CAY}(H, H \cap S)$ is irreducible and $\mathrm{SCAY}(G, S) = \mathrm{SCAY}(H, H \cap S)$, so $\mathrm{SCAY}(H, H \cap S)$ is disconnected. If $\mathrm{CAY}(H, H \cap S)$ is connected, Theorem 15 insures that $H \cap S$ is left periodic or periodic. Hence we may assume (i) $\mathrm{CAY}(H, H \cap S)$ is disconnected; i.e., $\langle H \cap S \rangle \neq H$. We know that $H \neq G$, so if $H \cap S_1$ is a proper subgroup of G, we are done. Hence we also may assume (ii) $H \cap S_1$ is not a subgroup of G.

Combining assumptions (i) and (ii), we are left to consider the case in which $H \cap S_1$ is a proper subset of $\langle H \cap S \rangle \neq H$. Thus, $\langle H \cap S \rangle$ contains a component C of $\mathrm{SCAY}(H, H \cap S)$. Since $C \subset \langle H \cap S \rangle \neq H$, C is nongenerating. If $\mathrm{SCAY}(H, H \cap S)$ has an isolated vertex, then by Theorem 2, $H \cap S$ is left periodic. If $\mathrm{SCAY}(H, H \cap S)$ has no isolated vertices, then by Proposition 3, the existence of a nongenerating component means that $H \cap S$ is nearly periodic. □

7. Concluding remarks. It is interesting to notice that when we move to a subversion strategy with two vertices, the situation becomes decidedly more complex. To see the substance of this observation, recall first that we have shown all prime order Cayley graphs remain connected upon subversion of a single vertex. In contrast, the variety of structure exhibited by the survival subgraphs resulting from subversion of two vertices of a prime order Cayley graph hints at the difficulty entailed in an analysis of this problem. It is easy to produce examples of prime order Cayley graphs that become disconnected upon subversion of two vertices. For $G = (\mathbf{Z}_{19}, +)$ and $S = \{\pm 1, \pm 3, \pm 5, \pm 6\}$, the subversion strategy $\{0, 9\}$ produces a disconnected survival subgraph of $\mathrm{CAY}(\mathbf{Z}_{19}, S)$ with two nontrivial components. For $T = \{\pm 1, \pm 3, \pm 6\}$, the subversion strategy $\{0, 8\}$ produces a disconnected survival subgraph of $\mathrm{CAY}(\mathbf{Z}_{19}, T)$ with an isolated vertex. For $U = \{\pm 1, \pm 3, \pm 9\}$, $\mathrm{CAY}(\mathbf{Z}_{19}, U)$ is connected upon subversion of any two vertices. So even in the case where G has prime order, anything can happen to $\mathrm{CAY}(G, S)$ when two vertices are subverted. It appears, therefore, that our methods, which rely heavily on the notion of periodicity, will not extend easily to the case in which two vertices are subverted.

Appendix: Computational complexity. We show that the decision problem for neighbor connectivity of an arbitrary graph is NP-complete. As membership in NP is clear, it suffices to show that the decision problem for neighbor connectivity is NP-hard.

Problem. The neighbor connectivity of an arbitrary graph.

Instance. Given any graph X and an integer $k < |V|$, where $V = V(X)$.

Question. Is $\mathrm{NC}(X) \leq k$?

THEOREM 17. *Neighbor connectivity is NP-hard.*

Proof. We show a polynomial reduction of the dominating set problem to this problem. Given graph X, construct $\hat{X}$ as follows (see Fig. 1).

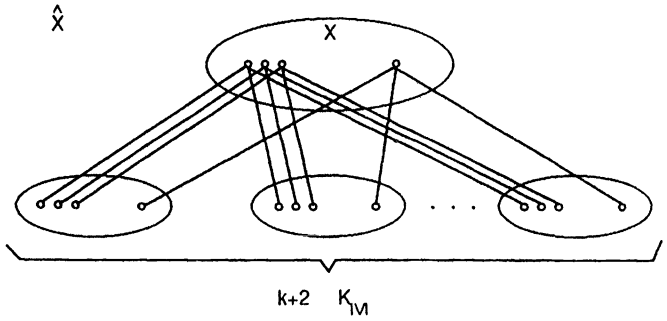


FIG. 1.

Set

$$V_j = \{v_{ij} \mid 1 \le i \le |V|\} \quad \text{for } j = 0, 1, \ldots, k+2, \text{ and}$$
$$\hat{V} = \bigcup_{j=0}^{k+2} V_j.$$

Then the subgraph induced by V_0, $[V_0]$, is X, and $[V_j] = K_{|V|}$ for $j = 0, 1, \ldots, k+2$. Finally, the remaining edges are of the form $\{v_{i0}, v_{ij}\}$ for $1 \le i \le |V|$, $1 \le j \le k+2$.

We show that X has a dominating set of size at most k if and only if $\hat{X}$ has an effective subversion strategy of size at most k. First suppose D is a dominating set of X of size at most k. Then subversion of D removes all the vertices of X and leaves $k+2$ cliques of size $|V| - |D|$. Thus $\mathrm{NC}(\hat{X}) \le k$.

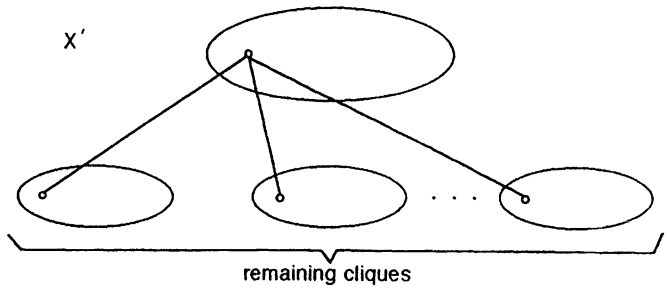


FIG. 2.

Next suppose that S is an effective subversion strategy of $\hat{X}$ with at most k vertices. Consider $S \cap V_0 = D$, and suppose that D is not a dominating set of X. If subversion of S removes all the vertices of X, then there are vertices of $S \setminus D$ that are adjacent to the vertices of V not dominated by D. Thus the number of vertices of V not dominated by D is less than or equal to the number of vertices in $S \setminus D$. Consequently, there is a dominating set in X of size at most k. Now suppose that subversion of S does not remove all the vertices of X. We claim that this assumption leads to a contradiction. Indeed, if the survival subgraph contains a nonempty subset $V_0' \subset V_0$, then the survival subgraph consists of $[V_0'] = X'$ and at least two complete subgraphs each of size at least $|V_0'|$ connected as shown in Fig. 2. Now nodes in separate complete subgraphs are not adjacent, so the survival subgraph is connected and not complete. In other words, S is not an effective subversion strategy. This is a contradiction. Thus there must be a dominating set in X of size at most k. □

REFERENCES

[1] G. GUNTHER, *Neighbour-connectivity in regular graphs*, Discrete Appl. Math., 11 (1985), pp. 233–243.

[2] ———, *On the existence of neighbour-connected graphs*, Congr. Numer., 54 (1986), pp. 105–110.

[3] G. GUNTHER AND B. L. HARTNELL, *On minimizing the effects of betrayals in a resistance movement*, in Proc. 8th Manitoba Conference on Numerical Mathematics and Computing, Winnipeg, Manitoba, Canada, 1978, pp. 285–306.

[4] ———, *Optimal k-secure graphs*, Discrete Appl. Math., 2 (1980), pp. 225–231.

[5] ———, *On m-connected and k-neighbour-connected graphs*, in Proc. of the 6th Quadrennial International Conference on the Theory and Applications of Graphs, Western Michigan University, Kalamazoo, MI, 1991, pp. 585–596.

[6] G. GUNTHER, B. L. HARTNELL, AND R. NOWAKOWSKI, *Neighbor-connected graphs and projective planes*, Networks, 17 (1987), pp. 241–247.

[7] S. Y. WU AND M. B. COZZENS, *The minimum size of critically m-neighbour-connected graphs*, Ars Combin., 29 (1990), pp. 149–160.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 643–653, November 1996

009

IIS-HYPERGRAPHS*

JENNIFER RYAN†

Abstract. Given an inconsistent set of inequalities $Ax \leq b$, the *irreducibly inconsistent subsystems* (IIS's) designate subsets of the inequalities such that at least one member of each subset must be deleted in order to achieve a feasible system. Each IIS can be considered the edge of a hypergraph. The purpose of this paper is to present several properties of this special class of hypergraphs (IIS-hypergraphs). IIS-hypergraphs are bicolourable, and their placement in Berge's hierarchy of "hypergraphs generalizing bipartite graphs" is discussed. The greedy algorithm finds the minimum transversal for 2-uniform IIS-hypergraphs. It is shown that the greedy algorithm does *not* work for general IIS-hypergraphs. However, if the IIS-hypergraph is "nondegenerate" (implying uniform), the transversal number always can be found in time polynomial in the size of the hypergraph. An interesting intermediate result arises regarding blocking pairs of polyhedra arising from subspaces in $\Re^n$.

Key words. hypergraphs, linear programming, infeasibility

AMS subject classifications. 05C65, 05C70, 90C05, 90C27

1. Introduction.

Let A be an $m \times n$ real matrix and let b be a real m-vector. Suppose that the system $Ax \leq b$ is inconsistent. A subsystem $A'x \leq b'$ is an *irreducibly inconsistent subsystem* (IIS) if it is inconsistent and if it has no inconsistent proper subsystem. In general, an inconsistent system will have many overlapping IIS's. In order to achieve a consistent system, at least one inequality must be dropped from every IIS. Sankaran [12] has shown that the problem of finding a minimum cardinality set of constraints to drop to achieve feasibility is NP-hard. (See also [1] for related complexity results.)

Given an inconsistent system $Ax \leq b$, let the set V index the inequalities. The corresponding *IIS-hypergraph* H on V is obtained by letting edge set E of H be the sets of indices of the IIS's. A *transversal* T of a hypergraph H is a subset of V covering the edge set E; i.e., $e \cap T \neq \emptyset$ for all $e \in E$. Thus a minimal transversal of H will be a minimal set T such that the inequalities indexed by $V \setminus T$ are a consistent set. Using notation of Berge [2] we will let TrH denote the *transversal hypergraph* of H. The edges of TrH are the setwise minimal transversals of H. $\tau(H)$ will denote the minimum cardinality of a transversal of H. Thus if H is an IIS-hypergraph, $\tau(H)$ is the minimum number of constraints that must be deleted or altered in order to achieve a feasible system. Below, the terms transversal and cover will both be used, but transversal will be always be used when referring to hypergraphs in order to be consistent with terminology of [2].

In [11] it is shown that 2-uniform IIS-hypergraphs are bipartite. A hypergraph is *bicolourable* if its nodes can be partitioned into two subsets so that neither of the subsets contains an edge. The following result from [11] follows immediately from an observation of Pulleyblank [9] that an inconsistent system can always be partitioned into two consistent systems.

THEOREM 1 (see [11]). *Any IIS-hypergraph H is bicolourable.*

Note that bicolourability of a hypergraph H does not imply that H is an IIS-hypergraph. In fact any bipartite graph is a bicolourable hypergraph, and as shown

* Received by the editors July 1, 1994; accepted for publication (in revised form) December 21, 1995.

† US WEST Advanced Technologies, 4001 Discovery Drive, Boulder, CO 80303 (jryan@advtech.uswest.com).

in [11], 2-uniform IIS-hypergraphs are bipartite graphs with very special structure. Section 2 discusses the relationship between IIS-hypergraphs and other classes of hypergraphs generalizing bipartite graphs.

2. Relation of IIS-hypergraphs to other hypergraphs generalizing bipartite graphs. Let M denote the edge-node incidence matrix of an IIS-hypergraph H. If H is 2-uniform (and so a bipartite graph from above), then it is well known that $\tau(H)$ is the value of the linear programming problem

$$\begin{array}{lll} \min & 1^T x & \\ & Mx & \geq 1, \\ & x & \geq 0. \end{array} \tag{1}$$

Note that if integrality conditions are added, the value gives the transversal number of *any* hypergraph.

There are many classes of hypergraphs generalizing bipartite graphs (see, e.g., Berge [2]). Many of these (for example balanced hypergraphs, normal hypergraphs, and Mengerian hypergraphs) have the property that the value of (1) is the transversal number. If the value of (1) is the transversal number, the hypergraph is called *paranormal* [2]. IIS-hypergraphs are *not* paranormal, as the following example shows.

Consider the drawing (Figure 1) of an inconsistent system lying in 2-dimensional space.

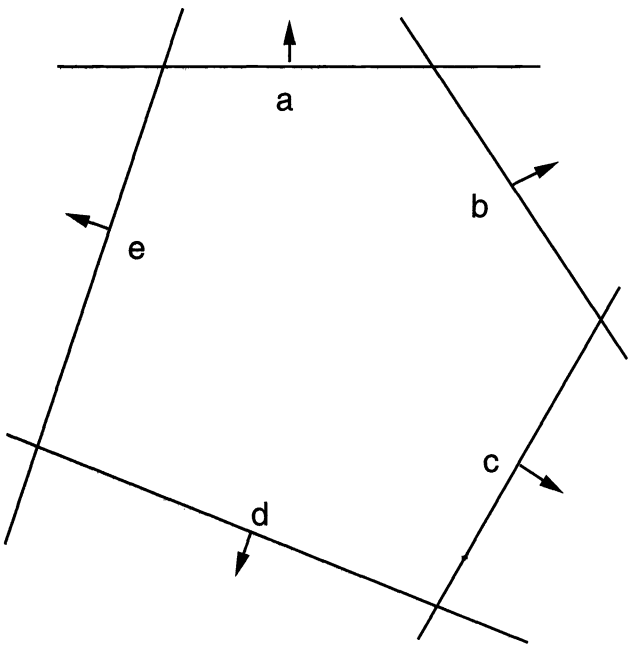


FIG. 1. *An inconsistent system in 2-dimensional space where the value of* (1) *is not an integer.*

The IIS's are $\{\{a,b,d\},\{b,c,e\},\{c,d,a\},\{d,e,b\},\{e,a,c\}\}$. The optimal solution

to (1) is achieved by setting $x_i = 1/3$ for each i which gives a value of 5/3. The transversal number is clearly 2.

Berge [2] has detailed the relationships between many classes of hypergraphs generalizing bipartite graphs. The diagram in Figure 2 is from [2] with the exception of the box in dotted lines representing IIS-hypergraphs. Definitions of properties in Figure 2 are given in the proof of Theorem 2 only if they are needed for the proof. For definitions of the other properties, the interested reader is referred to [2]. Theorem 2 shows that IIS-hypergraphs do not share many properties with other known classes of hypergraphs generalizing bipartite graphs.

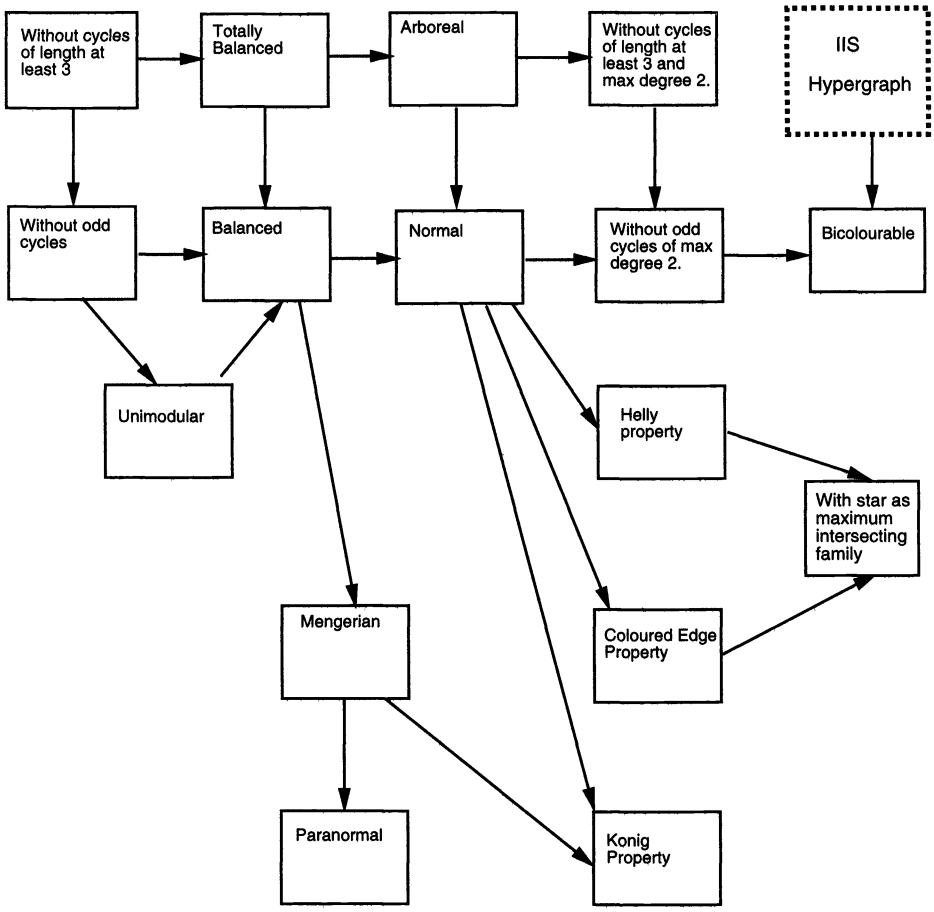


FIG. 2. *Hypergraphs generalizing bipartite graphs.*

THEOREM 2. *Figure 2 is complete. There are no other implications that could be added regarding IIS-hypergraphs.*

Proof. It will first be shown that there can be no additional arrows leading out of the IIS-hypergraph box. A hypergraph has the Konig property if the transversal number is equal to the cardinality of the maximum cardinality matching. In the example of Figure 1, the transversal number is 2, but since every pair of edges intersects, the maximum matching has cardinality 1, so IIS-hypergraphs do not in general have

the Konig property. By studying the implications of Figure 2, it is clear that in order to show that there are no additional outward arrows from the IIS-hypergraph box it suffices to show the following:

1. IIS-hypergraphs may have odd cycles with maximum degree 2.
2. IIS-hypergraphs may have a maximum intersecting family which is not a star.
3. IIS-hypergraphs are not always paranormal.

(The maximum degree of a cycle is the maximum degree of any node in the hypergraph whose edge set is the edges of the cycle.) We have already seen that statement 3 is true. A *cycle* in a hypergraph is a set of edges $e_1, e_2, \ldots, e_k$ such that there are nodes $x_1, x_2, \ldots, x_k$ satisfying $x_i, x_{i+1} \in e_i$ for $i = 1, \ldots, k-1$ and $x_k, x_1 \in e_k$. In the example of Figure 1, the three edges $e_1 = abd$, $e_2 = bce$, and $e_3 = cda$ form an odd cycle with maximum degree 2, so statement 1 is true. Finally, an *intersecting family* is a set of pairwise intersecting edges. A *star* is a set of edges with a common element. The five edges in Figure 1 form a maximum intersecting family which is not a star. Hence, statement 2 is true, so there can be no additional outward arrows from the IIS-hypergraph box.

To show that there can be no additional arrows leading into the IIS-hypergraph box, note that the property of having no cycles of length at least 3 implies all other boxes. So if there hypergraphs with no cycle of length at least 3 that are *not* IIS-hypergraphs, there can be no arrows leading into the IIS-hypergraph. Consider the hypergraph whose edges are $\{ab, bc, cd, de\}$. Clearly this has no cycle of length 3 or greater (it has no cycle at all). Any IIS of cardinality 2 must consist of nonintersecting parallel half-spaces, and there is no way to arrange five parallel half spaces to obtain the edge set $\{ab, bc, cd, de\}$ as the set of all IIS's. So this hypergraph is not an IIS-hypergraph. □

3. Finding the transversal number of an IIS-hypergraph. In order to further study transversals of IIS-hypergraphs we will need to make use of the following theorem which is easily derived from Farkas' lemma and elementary polyhedral theory. If $x \in \Re^n$, let the *support* of x be the set of indices $S(x) = \{j | x_j \neq 0\}$. Define $\text{pos}(x)$ to be

$$\text{pos}(x)_j = \begin{cases} x_j & \text{if } x_j > 0, \\ 0 & \text{otherwise.} \end{cases}$$

THEOREM 3 (see [8]). *Let $Ax \leq b$ denote an inconsistent set of inequalities. Then the IIS's are in one-to-one correspondence with the extreme points of the polyhedron $P = \{y \in \Re^n | yA = 0,\ yb = -1, y \geq 0\}$. In particular, the map $y \to S(y)$ is a bijection between the extreme points of P and the IIS's of the inconsistent system.*

Following Berge [2], we denote the *greedy transversal number* by $\tau_G(H)$. The greedy transversal number is the cardinality of the transversal obtained by successively picking the node having maximum degree among the currently uncovered edges. For general hypergraphs (even bipartite graphs), $\tau(H)$ can be much smaller than $\tau_G(H)$. In fact the ratio between τ_G and τ can be as much as $1 + \log(\Delta(H))$, where $\Delta(H)$ is the maximum degree of a node in H. (See, e.g., Chvátal [4].) For 2-uniform IIS-hypergraphs, the greedy algorithm finds the optimal transversal.

THEOREM 4 (see [11]). *Let H be a 2-uniform IIS-hypergraph. Then $\tau(H) = \tau_G(H)$.*

Theorem 4 shows that 2-uniform IIS-hypergraphs have a very special structure. In particular, not all bipartite graphs are IIS-hypergraphs, since the greedy algorithm

fails on general bipartite graphs. Unfortunately, Theorem 4 does not hold for general IIS-hypergraphs, as is shown by the example in Figure 3.

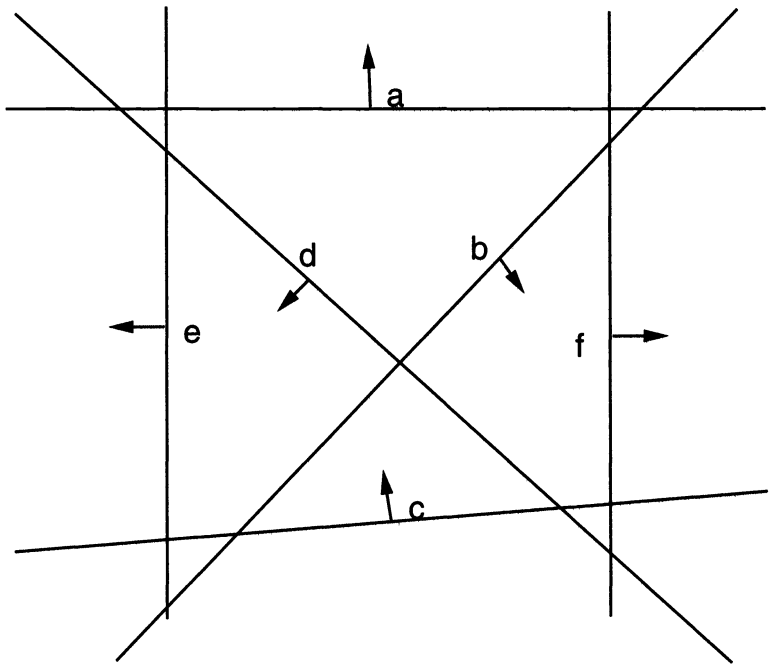


FIG. 3. *Example of an IIS-hypergraph where* $\tau(H) \neq \tau_G(H)$.

In Figure 3, several elements tie for the maximum degree (3). If a is chosen first, the minimum transversal will not be found using the greedy algorithm.

The remainder of this section is devoted to showing that although the greedy algorithm does not find the minimum transversal for all IIS-hypergraphs, a minimum transversal *can* be found in polynomial time if the hypergraph is *nondegenerate*. We say that an IIS-hypergraph is nondegenerate if the corresponding polyhedron (as in Theorem 3) is nondegenerate. Note that a nondegenerate IIS-hypergraph will always be uniform, i.e., all edges will have the same cardinality. In general the problem of finding a minimum transversal is NP-hard even for 2-uniform hypergraphs (i.e., graphs); see [7].

Recall that the IIS's are in one-to-one correspondence with the supports of the extreme points of some polyhedron P as described in Theorem 3. Thus we are interested in covering the extreme points of P or, equivalently, the *basic feasible solutions* of the system

$$\begin{aligned} A^T y &= 0, \\ \mathbf{1}^T y &= -1, \\ y &\geq 0. \end{aligned} \tag{2}$$

Bland [3] has an important result relating to the problem of covering the extreme points of a polyhedron. In order to present Bland's theorem, we need to establish some notation. For background on blocking polyhedra the reader is referred to [5] and [6].

If $\mathcal{R}$ is a vector subspace of $\Re^{n+1}$, let $\mathcal{R}^{\perp}$ denote the orthogonal complement of $\mathcal{R}$ and let $\mathcal{F}(\mathcal{R})$ denote the set of *elementary* vectors of $\mathcal{R}$ (the nonzero vectors of $\mathcal{R}$ having setwise minimal support). Bland considers the following sets:

$$\mathcal{C} = \{\text{pos}(x_1, x_2, \ldots, x_n) | (x_1, x_2, \ldots, x_n, 1) \in \mathcal{F}(\mathcal{R})\}$$

and

$$\mathcal{D} = \{x \in \Re^n_+ | (x, -1) \in \mathcal{F}(\mathcal{R}^{\perp})\}.$$

We now can state Bland's result.

THEOREM 5 (see [3]). *The blocker of the polyhedron* $\mathcal{B} = \{x \in \Re^n_+ | c \cdot x \geq 1, \forall c \in \mathcal{C}\}$ *is* $\hat{\mathcal{B}} = \{x \in \Re^n_+ | d \cdot x \geq 1, \forall d \in \mathcal{D}\}$.

Thus, the support of any vector in $\mathcal{C}$ is a transversal of the hypergraph whose edges are the supports of the vectors in $\mathcal{D}$. Let $\mathcal{R}$ be the row space of the $m \times (n+1)$ matrix $M = [C - d]$. Then $\mathcal{D}$ is the set of elementary vectors among those satisfying the system $Cx = d, x \geq \mathbf{0}$. In particular, the supports of all extreme points of the polyhedron $Q = \{x \in R^n | Cx = d, x \geq 0\}$ are covered by the support of any vector in C. It is not hard to see the following.

LEMMA 6 (see [10]). $\mathcal{F}(\mathcal{R})$ *is contained among those rows of tableaus of the form*

$$\left[B^{-1}C | B^{-1}d\right],$$

where B *is a column basis of* M.

A tableau $T = [B^{-1}C | B^{-1}d]$ will be called *feasible* if the right-hand side of T, $B^{-1}d$, is nonnegative. The *elements of a row* of T will mean those elements not including the right-hand side. Let T be any tableau. The positive elements of any row having a positive right-hand side will index an element of $\mathcal{C}$. Theorem 5 and Lemma 6 imply that *all* the elements of $\mathcal{C}$ can be found by enumerating the tableaus (T) over all column bases B of M.

In general, tableaus T may have both positive and negative elements in the last component. Theorem 7 says that all setwise minimal elements of $\mathcal{C}$ can be found by inspecting only feasible tableaus.

THEOREM 7. *Let* $\mathcal{R}$ *be the row space of* M *above. Then any minimal set covering the extreme points of the polyhedron* $Q = \{x \in R^n | Cx = d, x \geq 0\}$ *is the support of a vector in* $\mathcal{C}$. *Moreover, every setwise minimal element of* $\mathcal{C}$ *is the set of positive elements of some row with a positive right-hand side in a feasible tableau.*

Proof. From the discussion above, we already have that any minimal set covering the extreme points of the polyhedron $Q = \{x \in R^n | Cx = d, x \geq 0\}$ is the support of a vector in $\mathcal{C}$ and that all vectors of C will be the positive elements of a row with a positive right-hand side in some tableau T. What remains to be shown is that the search for *minimal* elements of $\mathcal{C}$ can be restricted to *feasible* tableaus.

Let K denote a minimal cover of the extreme points of Q. Then K is indexed by the set of positive elements of row k in some tableau T, where the right-hand side element of row k is positive. If the right-hand side of T is all nonnegative, we are done. So suppose there is some row ℓ of T with a negative right-hand side. It will be shown that there is always a dual simplex pivot preserving the set of positive entries of row k.

Since the dual simplex algorithm is finite (invoking anticyling rules if necessary), by considering row k as the objective row, we can always terminate with an all positive right-hand side. Since the set of positive elements in row k is unchanged, row k will still index the minimal cover K.

Since T is a tableau, there is some column basis B of C such that $T = [B^{-1}C|B^{-1}d]$. For ease of notation, assume that B is the first m column of C. Then column k of T will be the kth unit vector e_k. Let N_1 be the indices of nonbasic columns having a positive element in row k. Let N_2 be the index set of nonbasic columns having a nonpositive element in row k. Then T has the following sign pattern.

	k	ℓ	N_1	N_2	RHS
	$\begin{matrix} 1 & 0 & \cdots & 0 & \cdots \\ & \ddots & & \vdots & \\ & & 1 & 0 & \end{matrix}$	$\begin{matrix} 0 & \cdots \\ \vdots & \\ & \end{matrix}$			
k	$\begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & & 0 \\ \vdots & & 1 & 0 \end{matrix}$	$\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}$	$+ \cdots +$	$\leq 0 \cdots \leq 0$	$+$
ℓ	$\begin{matrix} 0 & 0 \\ \vdots & \vdots \end{matrix}$	$\begin{matrix} 1 \\ 0 \\ \vdots \end{matrix}$			$-$

Consider the N_2 columns and the basic columns of row k, the objective row in execution of the dual simplex algorithm. If a pivot is available in the N_2 columns, we can eliminate all negative right-hand side values, while maintaining the entries of row k nonpositive in all columns but k and the N_1 columns. By the minimality of the cover represented by row k, the entries in the N_1 columns will remain positive. So it remains to be shown that a pivot is always available in any row (such as row ℓ) with a negative right-hand side. That is, we must show that there is a negative element in row ℓ among the N_2 columns. Let $\bar{a}_{ij}$ denote the element of T in the ith row and jth column. Then, since any $x \in Q$ must satisfy row ℓ of T we have that for any $x \in Q$,

$$x_\ell + \sum_{j \in N_1} \bar{a}_{\ell j} x_j + \sum_{j \in N_1} \bar{a}_{\ell j} x_j < 0.$$

If there is no pivot available in the N_2 columns, $\bar{a}_{\ell j} \geq 0$ for all $j \in N_2$. Thus, since any $x \in Q$ is nonnegative for all $x \in Q$,

$$\sum_{j \in N_1} \bar{a}_{\ell j} x_j < 0.$$

Again, since each $x \in Q$ is nonnegative, we must have that for all $x \in Q$, $\sum_{j \in N_1} x_j > 0$. But then N_1 covers all extreme points of Q, contradicting the minimality of K. So the pivot is available, and the entries of row k in column k and the N_1 columns will remain positive (the set N_2 will change). The argument can be repeated (using appropriate dual simplex anticycling rules to pick a new ℓ) until no negative right-hand side element remains. □

The significance of Theorem 7 is computational. If one wants to enumerate all minimal "covers" of the extreme points of a polyhedron P, one must consider only

the rows of feasible tableaus, rather than all tableaus. In the case where the matrix $[C - d]$ is

$$\begin{bmatrix} A^T & \mathbf{0} \\ \mathbf{1}^T & -1 \end{bmatrix},$$

the supports of the minimal elements of $\mathcal{C}$ are the minimal transversals of the IIS-hypergraph on the system $Ax \leq b$.

THEOREM 8. *Let H be a nondegenerate IIS-hypergraph. Then the size of TrH is polynomial in the size of H.*

Proof. Let P be the polyhedron corresponding to H as in Theorem 3. If H is nondegenerate, then P is nondegenerate and there is only one feasible tableau corresponding to each extreme point of P, i.e., each edge of H. Let r be the number of elements in each edge of H. Since there will be at most r rows corresponding to distinct minimal transversals in each feasible tableau, there can be at most $|E(H)| * r$ minimal transversals of H. □

We now know that the number of minimal transversals of a nondegenerate IIS hypergraph H is polynomal in the size of H. If we can find an algorithm to enumerate TrH in time polynomial with respect to the size of TrH, we can find the minimum cardinality transversal of H simply by enumerating all minimal transversals. For simplicity, let a hypergraph H be defined by its edges. Let $H = (e_1, \ldots, e_q)$ and $H' = (f_1, \ldots, f_p)$ be hypergraphs and, following [2], let

$$H \cup H' = (e_1, \ldots, e_q, f_1, \ldots, f_p),$$
$$H \vee H' = \{e \cup f | e \in H, f \in H'\},$$

and

$$\text{Min } H = \{e | e \in H \text{ such that } e \text{ is setwise minimal in } H\}.$$

Note that $Tr(H \cup H') = \text{Min}(TrH \vee TrH')$.

Although we know that TrH has size bounded polynomially in the size of H, it is not trivial to find a polynomial enumeration scheme for TrH. In general, the number of intermediate steps can get too large. For example, consider the following algorithm given in Berge's book [2]. Suppose the edges of H have been indexed $e_1, e_2, \ldots, e_q$ and let H_i denote the hypergraph containing edges $e_1, e_2, \ldots, e_i$. Then the procedure below enumerates the edges of TrH, i.e., all minimal transversals of H.

Step 1. $TrH_1 = (\{k\} | k \in e_1)$.
Step 2. $TrH_2 = Tr(H_1 \cup \{e_2\}) = \text{Min}(TrH_1 \vee (\{k\} | k \in e_2))$.
Step i. $TrH_i = Tr(H_{i-1} \cup \{e_i\}) = \text{Min}(TrH_{i-1} \vee (\{k\} | k \in e_i))$ etc..
At the end of the procedure $TrH = TrH_q$.

Although we know that TrH is polynomial in size, the intermediate transversal hypergraphs may not be. This is because the intermediate hypergraphs will not in general be IIS-hypergraphs. It is easy to construct examples where these intermediate hypergraphs get large. Thus an alternate enumeration scheme is needed. Let G_i be the hypergraph obtained from H by considering only those edges using nodes $1, 2, \ldots, i$. In other words, G_i is the IIS-hypergraph of the system obtained by deleting constraints $i+1, \ldots, m$. Thus TrG_i is polynomial in the size of G_i which is smaller than the size of H.

If H is nondegenerate, we can order its nodes $1, 2, \ldots, m$ so that G_i contains at most one node more than G_{i-1}. We can also assume that the first nonempty G_i contains exactly one edge. This is because for every edge e there is another edge f differing in exactly one component. (In the event of degeneracy, it may be that there is no way to admit one node without admitting several others simultaneously.) Assume that the nodes of H are so ordered. Let E_i denote those edges containing node i, but not node j for any $j > i$. Then the following algorithm will create TrH in polynomial time.

MINIMAL TRANSVERSAL ENUMERATION ALGORITHM (MTEA)

Step 1. Let ℓ be the first index such that G_ℓ is nonempty. Then G_ℓ contains exactly one edge e and $TrG_\ell = (\{k\} | k \in e)$.

Step i $(i \geq \ell + 1)$. Let $\overline{TrG_{i-1}}$ denote those elements of TrG_{i-1} that cover E_i. Let $\underline{TrG_{i-1}}$ denote $TrG_{i-1} \setminus \overline{TrG_{i-1}}$. Let $T = \overline{TrG_{i-1}}$. For each edge $e \in \overline{TrG_{i-1}}$:

Step ia:

Let $F_{i_e} = \{f \in E_i | f \cap e = \emptyset\}$. Find TrF_{i_e} using MTEA.

Step ib:

Let $T = T \cup (e \vee TrF_{i_e})$.

Set $TrG_i = \text{Min}(T)$, etc.

At the end of the procedure $TrH = TrG_m$.

Theorem 9 establishes the correctness and complexity of MTEA.

THEOREM 9. *Let H be a nondegenerate IIS-hypergraph. Then MTEA enumerates all minimal transversals of H in polynomial time with respect to the size of H.*

Proof. Let r be the cardinality of each edge in H. The proof is by induction on the number of nodes p in H. Clearly if $p \leq r$, MTEA will terminate after Step 1 with TrH. Suppose that the nodes are ordered so that G_i contains at most one more node than G_{i-1} and that MTEA correctly finds TrH' for any nondegenerate IIS hypergraph with fewer than i nodes.

It will first be argued that Step i will be executed in time polynomial in the size of G_i and then that TrG_i is correctly identified. Since G_{i-1} is a nondegenerate IIS-hypergraph, TrG_{i-1} can be found in polynomial time (with respect to the size of G_{i-1}) by the induction hypothesis. Since G_{i-1} is contained in G_i, TrG_{i-1} (and also $\underline{TrG_{i-1}}$ and $\overline{TrG_{i-1}}$) can be formed in time polynomial with respect to the size of G_i.

Note that F_{i_e} is also a nondegenerate IIS-hypergraph corresponding to the system with constraints $i+1, \ldots, m$ and with the constraints indexed by e deleted. Since G_i contains at most one more node than G_{i-1} and since e contains at least one node, F_{i_e} contains at most $i - 1$ nodes. Thus by induction TrF_{i_e} can be found in polynomial time with respect to the size of F_{i_e}. Since F_{i_e} is contained in G_i, TrF_{i_e} can be found in time polynomial with respect to the size of G_i. Since TrF_{i_e} has cardinality that is polynomial with respect to the size of G_i, Step ib can also be executed in time polynomial in the size of G_i. Since Steps ia and ib are executed at most $|TrG_{i-1}|$ times, the size of T is bounded polynomially with respect to the size of G_i. Since the Min operation can be executed in polynomial time, the entire Step i can be executed in time that is polynomial in the size of G_i.

It now must be shown that Step i correctly identifies TrG_i. Let E_i be defined as in Step i of the MTEA.

$$TrG_i = Tr(G_{i-1} \cup E_i) = \text{Min}(TrG_{i-1} \vee TrE_i).$$

Consider $TrG_{i-1} \vee TrE_i$.

$$TrG_{i-1} \vee TrE_i = \cup_{e \in TrG_{i-1}} (e \vee TrE_i)$$

$$= \left\{ \cup_{e \in \overline{TrG_{i-1}}} (e \vee TrE_i) \right\} \bigcup \left\{ \cup_{e \in \underline{TrG_{i-1}}} (e \vee Tr(F_{i_e} \cup (E_i \setminus F_{i_e}))) \right\}$$

$$= \left\{ \cup_{e \in \overline{TrG_{i-1}}} (e \vee TrE_i) \right\} \bigcup \left\{ \cup_{e \in \underline{TrG_{i-1}}} (e \vee \mathrm{Min}(TrF_{i_e} \vee Tr(E_i \setminus F_{i_e}))) \right\}.$$

If $e \in \overline{TrG_{i-1}}$ then either $e \in TrE_i$ or some subset of e is in TrE_i. Thus $\mathrm{Min}\,(e \vee TrE_i) = e$ in this case. If $e \in \underline{TrG_{i-1}}$, then by definition of F_{i_e}, $e \in Tr(E_i \setminus F_{i_e})$, so $e \vee TrF_{i_e} \subseteq e \vee TrF_{i_e} \vee Tr(E_i \setminus F_{i_e})$. So in this case,

$$\mathrm{Min}\,(e \vee TrF_{i_e} \vee Tr(E_i \setminus F_{i_e})) = \mathrm{Min}\,(e \vee TrF_{i_e}).$$

We have that

$$TrG_i = \mathrm{Min}(TrG_{i-1} \vee TrE_i)$$

$$= \mathrm{Min} \left\{ \cup_{e \in \overline{TrG_{i-1}}} (e \vee TrE_i) \bigcup \cup_{e \in \underline{TrG_{i-1}}} (e \vee \mathrm{Min}(TrF_{i_e} \vee Tr(E_i \setminus F_{i_e}))) \right\}$$

$$= \mathrm{Min} \Big\{ \mathrm{Min} \left(\cup_{e \in \overline{TrG_{i-1}}} (e \vee TrE_i) \right)$$

$$\bigcup \mathrm{Min} \left(\cup_{e \in \underline{TrG_{i-1}}} (e \vee \mathrm{Min}(TrF_{i_e} \vee Tr(E_i \setminus F_{i_e}))) \right) \Big\}$$

$$= \mathrm{Min} \left\{ \overline{TrG_{i-1}} \bigcup \mathrm{Min} \left(\cup_{e \in \underline{TrG_{i-1}}} (e \vee TrF_{i_e}) \right) \right\} = \mathrm{Min}(T).$$

Thus Step i correctly identifies G_i in time polynomial with respect to the size of G.

Since each G_i is contained in H, algorithm MTEA correctly identifies TrH in time polynomial with respect to the size of H. □

Thus we have the following.

THEOREM 10. *The minimum cardinality transversal of a nondegenerate IIS-hypergraph can be found in time polynomial in the size of H.*

It is important to note that Theorem 10 does not conflict with Sankaran's NP-hard result, since the size of H may be exponential in the size of the original system $Ax \leq b$.

4. Discussion. Nondegenerate IIS-hypergraphs are a class of bicolourable hypergraphs where the minimum transversal problem can be solved in polynomial time. Several interesting questions remain.

STRUCTURE OF IIS-HYPERGRAPHS. A simple characterization of 2-uniform IIS-hypergraphs is given in [11]. The characterization is independent of the linear system of inequalities from which the hypergraph arose. No such characterization is known for general IIS-hypergraphs.

MORE EFFICIENT ALGORITHM FOR FINDING TRANSVERSAL. It has been shown here that the minimum cardinality transversal of a nondegenerate IIS-hypergraph can

be found in polynomial time. However, the proof depends on the fact that all minimal transversals can be enumerated in polynomial time. It would be nice to have an algorithm that finds the minimum cardinality transversal without such enumeration. In fact I do not know of a nondegenerate IIS-hypergraph where the greedy algorithm fails. There probably is one, but the example is likely nontrivial.

DEGENERATE IIS-HYPERGRAPHS. It is not known whether there is a polynomial time algorithm to find the minimum transversal of a degenerate IIS hypergraph. The arguments given here would fail in two places. First, there could be an exponential number of tableaus associated with each IIS, so the proof of Theorem 8 would fail. Second, the nodes of H could not be ordered as required by the enumeration algorithm if H was degenerate.

Note that any polyhedron $P = \{x \in \Re^n | Cx = d, x \geq 0\}$, where C is a real matrix and d is a vector, can be transformed to the form of (2). Thus any hypergraph whose edges are indexed by the supports of the extreme points of a polyhedron is an IIS-hypergraph. Perhaps they should be called *polyhedral hypergraphs*, but I chose to keep the name IIS-hypergraphs in reference to the original motivation for their study.

Acknowledgment. The author would like to thank Jon Lee for suggesting that Bland's result (Theorem 5) might be used to simplify the proof of Theorem 7 given in an earlier version of this paper.

REFERENCES

[1] E. AMALDI AND V. KANN, *The Complexity and Approximability of Finding Maximum Feasible Subsystems of Linear Relations*, Technical report ORWP 93/11, Ecole Polytechnique Federale do Lausanne, Lausanne, Switzerland, 1993.

[2] C. BERGE, *Hypergraphs*, North-Holland, Amsterdam, 1989.

[3] R. G. BLAND, *Elementary vectors and two polyhedral relaxations*, Math. Programming Study, 8 (1978), pp. 159–166.

[4] V. CHVÁTAL, *A greedy heuristic for the set covering problem*, Math. Oper. Res., 4 (1979), pp. 233–235.

[5] D. R. FULKERSON, *Blocking polyhedra*, in Graph Theory and its Applications, B. Harris, ed., Academic Press, New York, 1970, pp. 93–111.

[6] ———, *Blocking and antiblocking pairs of polyhedra*, Math. Programming, 1 (1971), pp. 168–194.

[7] M. GARY AND D. JOHNSON, *Computers and Intractibility*, W. H. Freeman, New York, 1979.

[8] J. GLEESON AND J. RYAN, *Identifying minimally inconsistent subsystems*, ORSA J. Comput., 2 (1990), pp. 61–67.

[9] W. PULLEYBLANK, Personal communication, 1988.

[10] R. T. ROCKAFELLAR, *The elementary vectors of a subspace of* $\Re^n$, in Combinatorial Mathematics and its Applications, R.C. Bose and T.A. Dowling, eds., University of North Carolina Press, Chapel Hill, NC, 1969, pp. 104–127.

[11] J. RYAN, *Transverals of IIS-hypergraphs*, Congr. Numer., 81 (1991), pp. 17–22.

[12] J. K. SANKARAN, *A note on resolving infeasibility in linear programs by constraint relaxation*, OR Letters, 13 (1993), pp. 19–20.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 654–673, November 1996

010

A GRAY CODE FOR NECKLACES OF FIXED DENSITY*

TERRY MIN YIH WANG† AND CARLA D. SAVAGE†

Abstract. A necklace is an equivalence class of binary strings under rotation. In this paper, we present a Gray code listing of all n-bit necklaces with d ones so that (i) each necklace is listed exactly once by a representative from its equivalence class and (ii) successive representatives, including the last and the first in the list, differ only by the transposition of two bits. The total time required is $O(nN(n,d))$, where $N(n,d)$ denotes the number of n-bit binary necklaces with d ones. This is the first algorithm for generating necklaces of fixed density which is known to achieve this time bound.

Key words. Gray codes, necklaces, combinatorial generation, Hamilton cycles

AMS subject classifications. 05A05, 05C45, 68R05, 68R10

1. Introduction.

In a combinatorial family, a Gray code is an exhaustive listing of the objects in the family so that successive objects differ only in a small way [Wil]. The classic example is the binary reflected Gray code [Gra], which is a list of all n-bit binary strings in which each string differs from its successor in exactly one bit. By applying the binary Gray code, a variety of problems have been solved and the complexities of the solutions to other problems have been improved [Gar, ChLeDu, ChChCh, Los, Ric]. There are many examples of combinatorial families for which Gray codes are known, including permutations [Joh, Tro], combinations [BuWi, NiWi, Rus1], compositions [Kli], set partitions [Kay], integer partitions [Sav, RaSaWe], binary trees [RuPr, Luc, LuRoRu], and linear extensions [PrRu1, PrRu2, Rus2, Sta, Wes].

When an application requires an exhaustive examination of all objects in a combinatorial family, Gray codes can be used to speed up the task. With a Gray code scheme, it is often possible to list a family of N objects, each of size $O(n)$, in time $O(n+N)$ rather than time $O(n*N)$, by listing the first object and thereafter listing only the (constant size) change between successive objects. There is an additional advantage if each object has an associated cost, for it is likely that the cost of an object can be computed in constant time from the cost of its predecessor on the Gray code list.

In this paper we consider Gray codes for binary *necklaces*. A *necklace* is an equivalence class of binary strings under rotation. To be precise, let $\Sigma = \{0,1\}$ and for $n \geq 0$ let Σ^n denote the set of all strings of length n over Σ. Define the rotation operation $\sigma : \Sigma^n \to \Sigma^n$ by

$$\sigma(x_1 x_2 \cdots x_n) = x_2 \cdots x_n x_1.$$

Then, strings x and y are in the same *necklace* if and only if $\sigma^i(x) = y$ for some integer i. A necklace can be identified by specifying any one of its elements and frequently the lexicographically smallest string is chosen as the representative. The *density* of a necklace is the number of ones in a representative of the necklace. We use $N(n)$ and $N(n,d)$ to denote the number of n-bit necklaces and the number of n-bit necklaces of density d, respectively.

* Received by the editors February 13, 1995; accepted for publication (in revised form) December 21, 1995. This research was supported by National Science Foundation grants CCR8906500 and DMS9302505.

† Department of Computer Science, North Carolina State University, Box 8206, Raleigh, NC 27695-8206 (cds@adm.csc.ncsu.edu).

Note that distinct n-bit strings of density d must differ in at least two positions. For this paper, by *a Gray code for necklaces of fixed density* we mean a listing of n-bit binary strings with d ones, exactly one from each necklace, in which successive strings differ in exactly two bit positions. We will show that such a Gray code is always possible and that it gives rise to the most efficient algorithm known for generating necklaces of fixed density.

A simple and elegant algorithm for listing the lexicographically smallest representatives of all n-bit necklaces was given in [FrMa, FrKe], and we refer to this as the FKM algorithm. It was shown in [RuSaWa] that the time required by the FKM algorithm is $O(N(n))$, that is, constant average time per necklace, which is the best possible time. The efficiency here is achieved by amortization rather than a Gray code, since successive representatives listed by the FKM algorithm can differ in $\Omega(n)$ bits. In fact, it can be shown that, in general, there is no listing for n-bit necklaces in which successive representatives differ in just one bit. In such a listing, the density of successive representatives would alternate between even and odd. However, this is impossible for even n, since the numbers of even-density and odd-density necklaces differ by more than one when $n > 0$.

In contrast to the situation with all necklaces, there is no parity problem which precludes, for any n and d, a Gray code for n-bit necklaces of density d. The main result of this paper is to show the existence of a Gray code for any n and d. The proof is constructive and the resulting algorithm, which has been implemented in the programming language **C**, takes time $O(nN(n,d))$. We note here that the necklace representative used in the Gray code is *not* the lexicographically smallest one, x. Instead, it is the representative obtained from x by applying σ to x until the leftmost bit is a one.

No previous algorithm for the problem of listing all n-bit necklaces of fixed density d is known to be as efficient as $O(nN(n,d))$ for general d. However, in the special case when $n = 2d + 1$, a Gray code for n-bit necklaces of density d follows from the Gray code of Ruskey and Proskurowski [RuPr] for balanced parentheses. This is determined by a straightforward bijection between these two families, and this algorithm achieves constant average time per object.

In §2 we present background and technical lemmas used for the main result. The Gray code construction is presented in §3 and its implementation is described and analyzed in §4.

2. Background and technical lemmas. For $\alpha,\ \beta \in \Sigma^*$, we use $\alpha \leq \beta$ ($\alpha < \beta$) to denote that α precedes (strictly precedes) β in lexicographic order. Let $L(n)$ be the set of lexicographically smallest representatives of the n-bit necklaces. That is,

$$L(n) \ = \ \{x \in \Sigma^n \mid x \leq \sigma^i(x) \ \text{ for } 1 \leq i < n\}.$$

Let $L(n,d)$ be the subset of $L(n)$ of strings of density d. As the backbone of our Gray code construction, we will use a tree of elements of $L(n)$, which was introduced in [WaSa]. For $n \geq 1$, let $\tau \ : \ \Sigma^n \to \Sigma^n$ be the function

$$\tau(x_1 x_2 \cdots x_n) \ = \ x_1 x_2 \cdots \overline{x_n}.$$

Then the tree, $TREE(n)$, is defined recursively by the following:

(i) 0^n is the root of $TREE(n)$ and
(ii) if x is a node of $TREE(n)$, then for $1 \leq i < n$, $\tau\sigma^i(x)$ is a child of x if and only if $\tau\sigma^i(x) \in L(n)$.

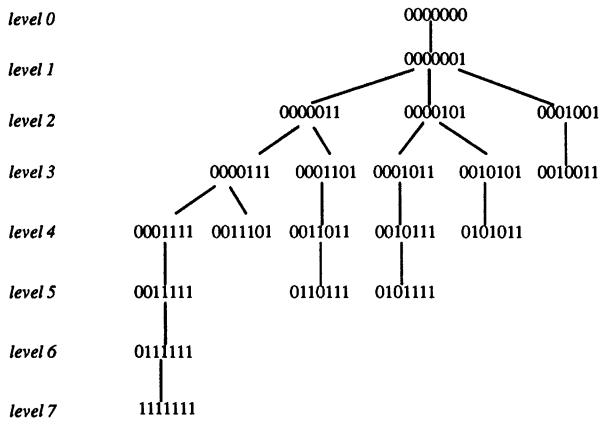


FIG. 1. *The tree of 7-bit necklaces, $TREE(7)$.*

As an example, $TREE(7)$ is shown in Figure 1. Note that no element of $L(n)$ appears more than once in $TREE(n)$, because if $x, y \in L(n)$ and $\tau\sigma^i(x) = \tau\sigma^j(y)$, then $\sigma^i(x) = \sigma^j(y)$, which means x and y are representatives of the same necklace and therefore must be identical. Thus, $TREE(n)$ has no cycles and, since by definition it is connected, it is in fact a tree. It is straightforward to verify that the nodes of $TREE(n)$ are exactly the elements of $L(n)$ and that $L(n,d)$ is the set of nodes on level d. First note that 0^n appears in the tree at level 0 and $0^{n-1}1$ appears at level 1. For $d \geq 2$, assume inductively that all elements in $L(n, d-1)$ appear in $TREE(n)$ at level $d-1$. Then $y \in L(n,d)$ can be written as $y = 0^k\alpha 10^i 1$, where $k, i \geq 0$. But then $y = \tau\sigma^{i+1}(0^{k+i+1}\alpha 1)$ and $x = 0^{k+i+1}\alpha 1 \in L(n, d-1)$. By induction, x is in $TREE(n)$ at level $d-1$ and therefore, by definition of $TREE(n)$, y is a child of x at level d.

The following result, crucial to our construction, was proved in [WaSa].

THEOREM 2.1. *For node $x = 0^k1\alpha$ in $TREE(n)$ with $k \geq 0$ and $\alpha \in \Sigma^*$ and for i satisfying $1 \leq i \leq k$, if $\tau\sigma^i(x) \notin L(n)$, then $\tau\sigma^{i+1} \notin L(n)$.* □

As a consequence of Theorem 2.1, if a node $x = 0^k1\alpha$ in $TREE(n)$ has exactly $c > 0$ children, then those children are $\tau\sigma^1(x), \tau\sigma^2(x), \ldots, \tau\sigma^c(x)$, and none of $\tau\sigma^{c+1}(x), \ldots, \tau\sigma^k(x)$ is in $L(n)$. Thus, if y is a child of $x = 0^k1\alpha$ in $TREE(n)$, there is a unique i, $1 \leq i \leq k$, such that $y = \tau\sigma^i(x)$.

For node x in $TREE(n)$, let $lev(x)$ denote the level of x in $TREE(n)$, where the root is at level 0. For $d \geq 0$, let $DESC(x,d)$ be the set of descendants of x in $TREE(n)$ on level $d + lev(x)$. A node x is called *d-rich* if $DESC(x,d) \neq \emptyset$. For example, in Figure 1, $x = 0000001$ is 2-*rich*, since

$$DESC(0000001, 2) = \{0000111,\ 0001101,\ 0001011,\ 0010101,\ 0010011\} \neq \emptyset.$$

Although we are interested in a Gray code for the set $DESC(0^n, d)$ of all n-bit necklaces of density d, we consider only $d \leq \lfloor n/2 \rfloor$, since, otherwise, a Gray code for $DESC(0^n, d)$ can be obtained from one for $DESC(0^n, n-d)$ by interchanging the roles of 0 and 1. Our construction will in fact give a Gray code for any set $DESC(x,d)$ with $d + lev(x) \leq \lfloor n/2 \rfloor$.

By repeated application of Theorem 2.1, if $x = 0^k1\alpha$, any $z \in DESC(x,d)$ can be written uniquely in the form

$$z = \tau\sigma^{a_d} \cdots \tau\sigma^{a_2}\tau\sigma^{a_1}(x),$$

where $0 < a_i < n$, $\tau\sigma^{a_i} \cdots \tau\sigma^{a_2}\tau\sigma^{a_1}(x) \in L(n)$ and $a_1 + \cdots + a_i \leq k$ for $1 \leq i \leq d$.

For a string $\alpha \in \Sigma^*$, let $\max(\alpha)$ be the length of the longest consecutive block of zeros in α. We repeatedly use the facts that since $x \in L(n)$ if and only if it is the lexicographically smallest of all of its rotations, (i) if $0^k1\alpha \in L(n)$ then $k \geq \max(\alpha)$ and (ii) if $k > \max(\alpha)$ then $0^k1\alpha \in L(n)$.

Our Gray code construction depends on the restrictive structure of the necklace tree and requires the technical lemmas below. For a string $\alpha \in \Sigma^*$, define $\#ones(\alpha)$ and $\#zeros(\alpha)$ to be the total number of ones and zeros, respectively, in α. (The density of α is $\#ones(\alpha)$.) We denote the *empty string* by λ.

LEMMA 2.2. *Assume that $\alpha = \lambda$ or that $\alpha \in \Sigma^+$ and α ends in 1. For $z, z' \in \Sigma^n$, if $z = 0^a1\alpha0^b\beta$ is in $L(n)$ where $a \geq 1$ and $b \geq 1$, then any n-bit string of the form*

$$z' = 0^{a+c}1\alpha0^{b-1}1\beta'$$

must be in $L(n)$ for any $c \geq 0$ and $\beta' \in \Sigma^*$ with $a + c > \max(\beta')$. Furthermore, if $z \in L(n, j)$ and $j \leq \lfloor n/2 \rfloor$, then either $a > 1$ or $a \geq \#ones(\beta) - \#zeros(\beta)$.

Proof. Since $z \in L(n)$,

(i) $a \geq b$ and

(ii) either (a) $a > \max(\alpha)$

or (b) $a = \max(\alpha)$ and $z \leq w$ for any rotation w of z.

If (i) and (ii)(a) hold, then $z' \in L(n)$. Otherwise, (i) and (ii)(b) hold so $a = \max(\alpha)$ and $\alpha \neq \lambda$. If $z' \notin L(n)$ then there exist $\alpha_1, \alpha_2 \in \Sigma^*$, with $|\alpha_1| \geq 0$ and $|\alpha_2| \geq 1$, such that α_2 starts with "1" and $\alpha = \alpha_1 0^a \alpha_2$ and z' is greater than its rotation $r(z')$.

$$z' > 0^a\alpha_2 0^{b-1}1\beta'0^{a+c}1\alpha_1 = r(z'). \tag{1}$$

It must follow that $c = 0$. Let $r(z)$ be the rotation of z:

$$r(z) = 0^a\alpha_2 0^b\beta 0^a 1\alpha_1.$$

Note that $0^{a+c}1\alpha \leq z$, and by (ii)(b), $z \leq r(z)$. Also, $r(z) < r(z')$, so $0^{a+c}1\alpha < r(z')$. Combining this with (1), since $0^{a+c}1\alpha$ is a prefix of z', $0^{a+c}1\alpha$ is also a prefix of $r(z')$. However, note that the prefix $0^a\alpha_2 0^{b-1}1$ of $r(z')$ is shorter than $0^{a+c}1\alpha$.

$$\begin{aligned} |0^a\alpha_2 0^{b-1}1| &= a + |\alpha_2| + b \\ &\leq a + |\alpha_2| + a && \text{(by (i))}, \\ &< a + c + 1 + |\alpha_1| + a + |\alpha_2| && \text{(since } c \geq 0 \text{ and } |\alpha_1| \geq 0\text{)}, \\ &= |0^{a+c}1\alpha|. \end{aligned}$$

Therefore, $0^a\alpha_2 0^{b-1}1$ is a prefix of $0^{a+c}1\alpha$. Thus,

$$0^a\alpha_2 0^{b-1}1 \leq 0^{a+c}1\alpha \leq z \leq r(z) = 0^a\alpha_2 0^b\beta 0^a 1\alpha_1,$$

a contradiction.

If $z \in L(n, j)$ and $j \leq \lfloor n/2 \rfloor$, then $\#zeros(z) \geq \#ones(z)$, so

$$a + b + \#zeros(\alpha) + \#zeros(\beta) \geq 1 + \#ones(\alpha) + \#ones(\beta).$$

Thus,

$$a \geq (\#ones(\alpha) - \#zeros(\alpha)) + (\#ones(\beta) - \#zeros(\beta)) + 1 - b. \tag{2}$$

If $a \leq 1$, then since $a \geq b \geq 1$, it follows that $a = b = 1$, so $\max(\alpha) \leq 1$. Therefore α cannot have more zeros than ones, since α ends in 1. Thus, from (2), $a \geq \#ones(\beta) - \#zeros(\beta)$. □

We introduce a notation to label key nodes of $TREE(n)$ used in the Gray code construction. For a binary string, z, define u, u', u'', v, v', and w as follows:

$$\begin{array}{lll}
\text{when } d \geq 0, & & \\
 & u(z,d) = (\tau\sigma)^d(z); & \\
\text{when } d = 0, & & \\
 & v(z,d) = z; & \\
\text{when } d \geq 1, & & \\
 & v(z,d) = \tau\sigma^2(\tau\sigma)^{d-1}(z), & \\
 & w(z,d) = \tau\sigma^3(\tau\sigma)^{d-1}(z), & \\
 & u'(z,d) = (\tau\sigma)^{d-1}\tau\sigma^2(z), & \\
 & u''(z,d) = (\tau\sigma)^{d-1}\tau\sigma^3(z); & \\
\text{when } d \geq 2, & & \\
 & v'(z,d) = \tau\sigma^2(\tau\sigma)^{d-2}\tau\sigma^2(z). &
\end{array}$$

For example, in Figure 1, if $z = 0000001$, then $u(z,2) = 0000111$, $v(z,2) = 0001101$, $u'(z,2) = 0001011$, $v'(z,2) = 0010101$, and $u''(z,2) = 0010011$. Note that for general z, $u(z,0) = z$, $u'(z,1) = v(z,1)$, and $u''(z,1) = w(z,1)$.

Let $x \in L(n)$ and let $y_i = \tau\sigma^i(x)$ denote the ith child of x. Further use of the vertex labels is illustrated in Figure 2. The lemma below shows that existence of a node at certain locations in $TREE(n)$ forces the existence of nodes at certain other locations in $TREE(n)$.

LEMMA 2.3. *Let $x \neq 0^n$ be a node in $TREE(n)$. Let $y_i = \tau\sigma^i(x)$ and assume that $j \leq \lfloor n/2 \rfloor$. For*

$$\begin{array}{llllll}
1 \leq i < n-1,\ d \geq 1, & \text{(i)} & \textit{if } u(y_{i+1},d) \in L(n,j), & \textit{then } u(y_i,d) \in L(n,j), \\
 & \text{(ii)} & \textit{if } u(y_{i+1},d) \in L(n,j), & \textit{then } v(y_i,d) \in L(n,j), \\
 & \text{(iii)} & \textit{if } u(y_{i+1},d) \in L(n,j), & \textit{then } u'(y_i,d) \in L(n,j); \\
1 \leq i < n-1,\ d \geq 2, & \text{(iv)} & \textit{if } v(y_{i+1},d) \in L(n,j), & \textit{then } v'(y_i,d) \in L(n,j); \\
1 \leq i < n-2,\ d \geq 1, & \text{(v)} & \textit{if } v(y_{i+2},d) \in L(n,j), & \textit{then } u''(y_i,d) \in L(n,j).
\end{array}$$

Proof. (See Figure 2.) Write x as $x = 0^k1\alpha$ where $\alpha \in \Sigma^*$. Then,

$$\begin{array}{lll}
u(y_{i+1},d) = (\tau\sigma)^d\tau\sigma^{i+1}(x) & = & 0^{k-i-d-1}1\alpha 0^i 1^{d+1}, \\
u(y_i,d) = (\tau\sigma)^d\tau\sigma^i(x) & = & 0^{k-i-d}1\alpha 0^{i-1}1^{d+1}, \\
v(y_i,d) = \tau\sigma^2(\tau\sigma)^{d-1}\tau\sigma^i(x) & = & 0^{k-i-d-1}1\alpha 0^{i-1}1^d01, \\
u'(y_i,d) = (\tau\sigma)^{d-1}\tau\sigma^2\tau\sigma^i(x) & = & 0^{k-i-d-1}1\alpha 0^{i-1}101^d, \\
v(y_{i+1},d) = \tau\sigma^2(\tau\sigma)^{d-1}\tau\sigma^{i+1}(x) & = & 0^{k-i-d-2}1\alpha 0^i 1^d01, \\
v'(y_i,d) = \tau\sigma^2(\tau\sigma)^{d-2}\tau\sigma^2\tau\sigma^i(x) & = & 0^{k-i-d-2}1\alpha 0^{i-1}101^{d-1}01, \\
v(y_{i+2},d) = \tau\sigma^2(\tau\sigma)^{d-1}\tau\sigma^{i+2}(x) & = & 0^{k-i-d-3}1\alpha 0^{i+1}1^d01, \\
u''(y_i,d) = (\tau\sigma)^{d-1}\tau\sigma^3\tau\sigma^i(x) & = & 0^{k-i-d-2}1\alpha 0^{i-1}1001^d.
\end{array}$$

Cases (i)–(iii) of Lemma 2.3 follow from Lemma 2.2 with $a = k-i-d-1$, $b = i$, $\beta = 1^{d+1}$, and

$$\begin{array}{l}
\text{for (i), with } \quad c = 1, \quad \beta' = 1^d, \\
\text{for (ii), with } \quad c = 0, \quad \beta' = 1^{d-1}01, \\
\text{for (iii), with } \quad c = 0, \quad \beta' = 01^d.
\end{array}$$

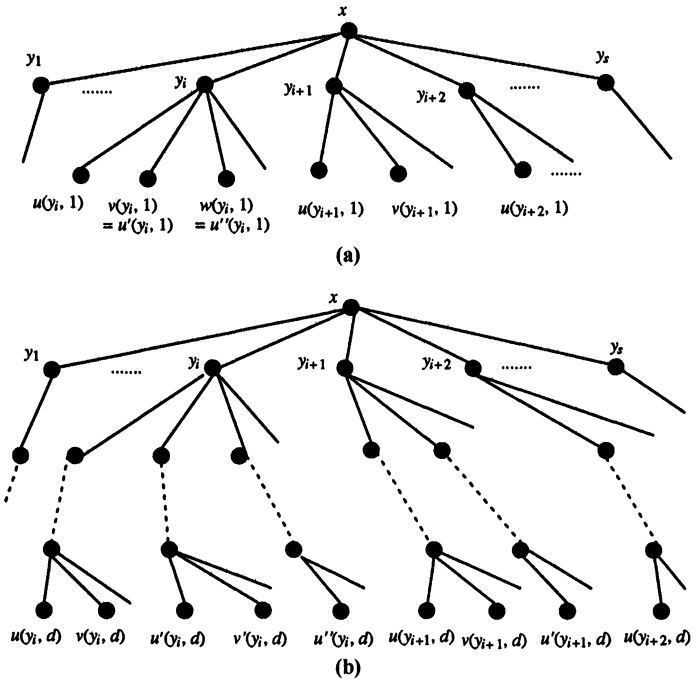


FIG. 2. *Labeling key tree nodes* (a) *when* $d = 1$, (b) *when* $d \geq 2$.

In each case (i)–(iii), $\max(\beta') \leq 1$. Note that by Lemma 2.2 either $a > 1$, so that $a + c > 1 + c > 1 - c = \max(\beta')$, or

$$a + c \geq \#ones(\beta) - \#zeros(\beta) + c = d + 1 + c \geq 2 + c \geq 2 > \max(\beta').$$

Case (iv) follows from Lemma 2.2 with $a = k-i-d-2$, $b = i$, $\beta = 1^d01$, $c = 0$, and $\beta' = 01^{d-1}01$. Note that $d \geq 2$ in this case and by Lemma 2.2, either $a > 1 = \max(\beta')$ or $a \geq \#ones(\beta) - \#zeros(\beta) = d \geq 2 > \max(\beta')$.

Case (v) follows from Lemma 2.2 with $a = k-i-d-3$, $b = i$, $\beta = 01^d01$, $c = 1$, and $\beta' = 001^d$. Note that $a \geq i + 1 \geq 2$, since $v(y_{i+2}, d) \in L(n)$. Thus $a + c = a + 1 > 2 = \max(\beta')$. □

Let x be a node in $TREE(n)$ with c children $y_1, \ldots, y_c$, where $y_j = \tau\sigma^j(x)$. For $d \geq 1$ and $1 \leq i \leq c$, define

$$V(x, d, i) = \bigcup_{j=i}^{c} DESC(y_j, d - 1).$$

Then note that $DESC(x, d) = V(x, d, 1)$. In Figure 1,

$$\begin{aligned} V(0000001, 2, 2) &= DESC(0000101, 1) \cup DESC(0001001, 1) \\ &= \{0001011,\ 0010101,\ 0010011\}. \end{aligned}$$

The next section gives a recursive construction of a Gray code for $V(x, d, i)$ based on the following recursive decomposition of $V(x, d, i)$:

$$V(x, d, i) = DESC(y_i, d - 1) \cup V(x, d, i + 1).$$

Note that for a given $d \geq 1$, not every child of x need be $(d-1)$-*rich*. Let $r(x,d)$ be the number of $(d-1)$-*rich* children of x. Corollary 2.4(a) below generalizes Theorem 2.1 to establish that the $(d-1)$-*rich* children of x must be exactly $y_1, y_2, \ldots, y_{r(x,d)}$.

COROLLARY 2.4. *Let* $x \in L(n,b)$, $b \geq 1$, $d \geq 1$, $b+d \leq \lfloor n/2 \rfloor$. *For* $1 \leq i \leq n$, *let* $y_i = \tau\sigma^i(x)$.

(a) *If* $V(x,d,i) \neq \emptyset$, *then* $u(y_i, d-1) \in DESC(y_i, d-1)$.

(b) *If* $|V(x,d,i)| \geq 2$, *then* $DESC(y_i, d-1)$ *contains both* $u(y_i, d-1)$ *and* $v(y_i, d-1)$.

Proof. The corollary is proved by induction on d. If $d = 1$, the result follows from Theorem 2.1. Assume the result is true for d' satisfying $1 \leq d' < d$. If $V(x,d,i) \neq \emptyset$, then $DESC(y_{i+j}, d-1) \neq \emptyset$ for some j satisfying $0 \leq j < n-i$ and $y_{i+j} \in L(n)$.

Let $z = \tau\sigma(y_{i+j})$. Then $u(z, d-2) = u(y_{i+j}, d-1)$. Since $DESC(y_{i+j}, d-1) = V(y_{i+j}, d-1, 1)$, by induction $u(z, d-2) \in DESC(z, d-2) \subseteq DESC(y_{i+j}, d-1)$ and therefore $u(y_{i+j}, d-1) \in DESC(y_{i+j}, d-1)$. So by Lemma 2.3(i), $u(y_{i+j-1}, d-1) \in DESC(y_{i+j-1}, d-1)$. Part (a) follows from $j-1$ further applications of Lemma 2.3(i). Part (b) follows from (a) and Lemma 2.3(ii). □

3. The Gray code construction. We first introduce a different collection of necklace representatives which are a slight variation on the L-representatives. Note that although the L-representatives $x = 00000011101$ and $y = 00001110001$ differ in four bit positions, there exist rotations x', y' of x, y, respectively, which differ in only two bit positions. Although we could call x and y "adjacent" if there exist rotations of x and y which differ in only two bit positions, we found that by using the M-representatives, defined below, we could make explicit the rotations of x and y which differ in only two bit positions, at least for the adjacencies which will be used in our construction.

For a necklace representative $x \in L(n)$, either $x = 0^n$ or x can be written uniquely as $x = 0^k 1\alpha$ for some $k \geq 0$ and $\alpha \in \Sigma^*$. Define a function M on $L(n)$ by

$$M[x] = \begin{cases} x & \text{if } x = 0^n, \\ 1\alpha 0^k & \text{if } x = 0^k 1\alpha. \end{cases}$$

Since $M[x]$ and x are in the same necklace, the set

$$\{M[x] \mid x \in L(x)\}$$

is an alternate set of necklace representatives, called *M-representatives.*

Define $x, y \in L(n)$ to be *M-adjacent* if $M[x]$ and $M[y]$ differ only by the interchange of a 0 and a 1. Let G_n be the graph whose vertex set is $L(n)$, with two vertices adjacent in G_n if and only if they are M-adjacent. Let $G_n[x,d,i]$ be the subgraph of G_n induced by $V(x,d,i)$. Note that if $z_1, z_2, \ldots, z_s$ is a Hamilton path in $G_n[x,d,i]$, then $M[z_1], M[z_2], \ldots, M[z_s]$ is a Gray code for the necklaces in $V(x,d,i)$. For example, in Figure 1, the graph $G_7[0000001, 2, 1]$ is a graph with vertex set

$$V(0000001, 2, 1) = \{0000111,\ 0001101,\ 0001011,\ 0010101,\ 0010011\}.$$

A Hamilton path in this graph is

$$0000111,\ 0010101,\ 0001011,\ 0010011,\ 0001101,$$

and the corresponding list of M-representatives, giving a Gray code, is

$$1110000,\ 1010100,\ 1011000,\ 1001100,\ 1101000.$$

Our goal is to find a Gray code for the necklaces in $DESC(0^n, d) = V(0^n, d, 1) = V(0^{n-1}1, d-1, 1)$ when $1 \le d \le \lfloor n/2 \rfloor$. This will follow if we can show that $G_n[x, d, i]$ has a Hamilton path for every (x, d, i) satisfying $x \in L(n)$, $lev(x) + d \le \lfloor n/2 \rfloor$, and $1 \le i \le r(x, d)$.

The strategy will be recursive. Let $y_i = \tau\sigma^i(x)$. Since it was shown in §2 that $V(x, d, i) = V(y_i, d-1, 1) \cup V(x, d, i+1)$, Hamilton paths/cycles recursively constructed in $G_n[y_i, d-1, 1]$ and $G_n[x, d, i+1]$ will be linked to form a Hamilton path/cycle in $G_n[x, d, i]$. The full construction is contained in Theorem 3.7 at the end of this section, following some further technical lemmas.

In the next sequence of lemmas, we examine the structure of $G_n[x, d, i]$. In particular, we show that the graph is complete whenever $d = 1$ or $i = r(x, d)$. In addition, we establish the existence of certain edges which will be used to link together Hamilton cycles recursively constructed. When x is fixed, we let $y_i = \tau\sigma^i(x)$.

LEMMA 3.1. *For any node x in $TREE(n)$, $G_n[x, 1, 1]$ is complete.*

Proof. If $x = 0^n$, the only child of x is $0^{n-1}1$. Otherwise, x has the form $x = 0^k1\alpha$ for some $\alpha \in \Sigma^*$. Let $\tau\sigma^i(x)$ and $\tau\sigma^j(x)$ be distinct children of x with $1 \le i, j \le n$. Then

$$M[\tau\sigma^i(x)] \;=\; 1\alpha 0^{i-1}10^{k-i}$$

and

$$M[\tau\sigma^j(x)] \;=\; 1\alpha 0^{j-1}10^{k-j},$$

and these differ only by the exchange of a 0 and 1. □

LEMMA 3.2. *For $x \neq 0^n$ in $TREE(n)$ and $d \ge 2$, let $r = r(x, d)$. Let $y_i = \tau\sigma^i(x)$. Any z in $DESC(y_r, d-1)$ has the form*

$$z = \tau\sigma^{a_{d-1}}\tau\sigma^{a_{d-2}}\cdots\tau\sigma^{a_1}(y_r),$$

where $1 \le a_i \le 2$ for $1 \le i \le d-1$ and $a_i = 2$ for at most one $i \in \{1, \ldots, d-1\}$.

Proof. Since $x \neq 0^n$, x can be written as $0^k1\alpha$ for some $\alpha \in \Sigma^*$ where $\alpha = \lambda$ or α ends in 1. Any $z \in DESC(y_r, d-1)$ can be written uniquely as $z = \tau\sigma^{a_{d-1}}\cdots\tau\sigma^{a_1}(y_r)$, where $1 \le a_i \le n$ for $1 \le i \le d-1$. We show that $a_1 + \cdots + a_{d-1} \le d$.

By definition of $r = r(x, d)$, $z' = u(y_{r+1}, d-1) \notin L(n)$. Expand z and z' as

$$z = \tau\sigma^{a_{d-1}}\cdots\tau\sigma^{a_1}\tau\sigma^r(x) = 0^{k-r-a_1-\cdots-a_{d-1}}1\alpha 0^{r-1}10^{a_1-1}1\cdots 0^{a_{d-1}-1}1,$$

$$z' = u(y_{r+1}, d-1) = (\tau\sigma)^{d-1}\tau\sigma^{r+1}(x) = 0^{k-r-d}1\alpha 0^r 1^d.$$

Let $s = a_1 + \cdots + a_{d-1}$ and assume that $s > d$. Since $z \in L(n)$, $k - r - s \ge \max(\alpha)$ and

$$k - r - d \;>\; k - r - d - 1 \;\ge\; k - r - s \;\ge\; r - 1. \tag{3}$$

But then since $z' \notin L(n)$, it must be that $k - r - d = r$ and z' is larger than its rotation $r(z')$:

$$0^r1\alpha 0^r 1^d \;=\; z' \;>\; r(z') \;=\; 0^r 1^d 0^r 1\alpha. \tag{4}$$

Let $\beta = 0^{a_1-1}1\cdots 0^{a_{d-1}-1}1$ and let $r(z)$ be the following rotation of $z = 0^{k-r-s}1\alpha 0^{r-1}1\beta$:

$$r(z) \;=\; 0^{r-1}1\beta 0^{k-r-s}1\alpha.$$

Since $|\beta| = s > d$ and $\#ones(\beta) = d - 1$,

$$\beta 0^{r-1}1\alpha \ < \ 1^{d-1}0^r1\alpha. \tag{5}$$

But then,

$$\begin{aligned}
0z &= 0^{k-r-s+1}1\alpha 0^{r-1}1\beta, \\
&\geq 0^{k-r-d}1\alpha 0^{r-1}1\beta && \text{since } s > d, \\
&= 0^r1\alpha 0^{r-1}1\beta && \text{since } k-r-d = r, \\
&> z' > r(z') && \text{by (4)}, \\
&= 0^r1^d0^r1\alpha, \\
&> 0^r1\beta 0^{r-1}1\alpha && \text{by (5)}, \\
&\geq 0^r1\beta 0^{k-r-s}1\alpha && \text{by (3)}, \\
&= 0r(z).
\end{aligned}$$

Thus $0z > 0r(z)$, so $z > r(z)$, which is impossible since z is a necklace. □

LEMMA 3.3. *For $x \neq 0^n$ in $TREE(n)$ and $d \geq 2$, let $r = r(x,d)$. Then, $G_n[x,d,r]$ is complete.*

Proof. Since $x \neq 0^n$, x can be written as $x = 0^k1\alpha$, where $\alpha \in \Sigma^*$ and either $\alpha = \lambda$ or α ends in 1. By Lemma 3.2, any vertex of $G_n[x,d,r] = G_n[y_r, d-1, 1]$ has the form

$$\tau\sigma^{a_{d-1}} \cdots \tau\sigma^{a_1}\tau\sigma^r(x) \ = \ 0^{k-r-a_1-a_2-\cdots-a_{d-1}}1\alpha 0^{r-1}10^{a_1-1}1 \cdots 0^{a_{d-1}-1}1,$$

where $1 \leq a_i \leq 2$ for $1 \leq i \leq d-1$ and $a_i = 2$ for at most one $i \in \{1, \ldots, d-1\}$. This means that $s = a_1 + \cdots + a_{d-1}$ is either $d-1$ or d and any vertex has the form

$$\text{(i) } 0^{k-r-d+1}1\alpha 0^{r-1}1^d \qquad (\text{if } \ s = d-1)$$

or

$$\text{(ii) } 0^{k-r-d}1\alpha 0^{r-1}1^i01^{d-i} \qquad (\text{if } \ s = d).$$

Clearly, any two vertices of this form are M-adjacent. □

LEMMA 3.4. *Let $x \neq 0^n$ be a node in $TREE(n)$ with $d \geq 1$ and $r = r(x,d)$. For each of the following pairs of strings z, z', if both z and z' are in $L(n)$, then they are M-adjacent:*

(a)	$u(y_i,d)$	*and*	$v(y_i,d)$	*for*	$d \geq 1$	*and*	$1 \leq i \leq r$,
(b)	$u(y_i,d)$	*and*	$u(y_{i+1},d)$	*for*	$d \geq 1$	*and*	$1 \leq i < r$,
(c)	$u(y_i,d)$	*and*	$v(y_{i+1},d)$	*for*	$d \geq 1$	*and*	$1 \leq i < r$,
(d)	$v(y_i,d)$	*and*	$u(y_{i+1},d)$	*for*	$d \geq 1$	*and*	$1 \leq i < r$,
(e)	$u'(y_i,d)$	*and*	$u(y_{i+1},d)$	*for*	$d \geq 2$	*and*	$1 \leq i < r$,
(f)	$v'(y_i,d)$	*and*	$v(y_{i+1},d)$	*for*	$d \geq 2$	*and*	$1 \leq i < r$,
(g)	$u'(y_i,d)$	*and*	$u(y_{i+2},d)$	*for*	$d \geq 2$	*and*	$1 \leq i < r-1$,
(h)	$v'(y_i,d)$	*and*	$u(y_{i+2},d)$	*for*	$d \geq 2$	*and*	$1 \leq i < r-1$,
(i)	$w(y_i,1)$	*and*	$v(y_{i+1},1)$	*for*	$1 \leq i < r$,		
(j)	$v(y_i,1)$	*and*	$u(y_{i+2},1)$	*for*	$1 \leq i < r-1$,		
(k)	$w(y_i,1)$	*and*	$u(y_{i+2},1)$	*for*	$1 \leq i < r-1$.		

Proof. These edges are illustrated in Figure 3, when $d = 1$, and Figure 4, when $d > 1$. Using the definitions of the vertices and the fact that x has the form $0^k1\alpha$

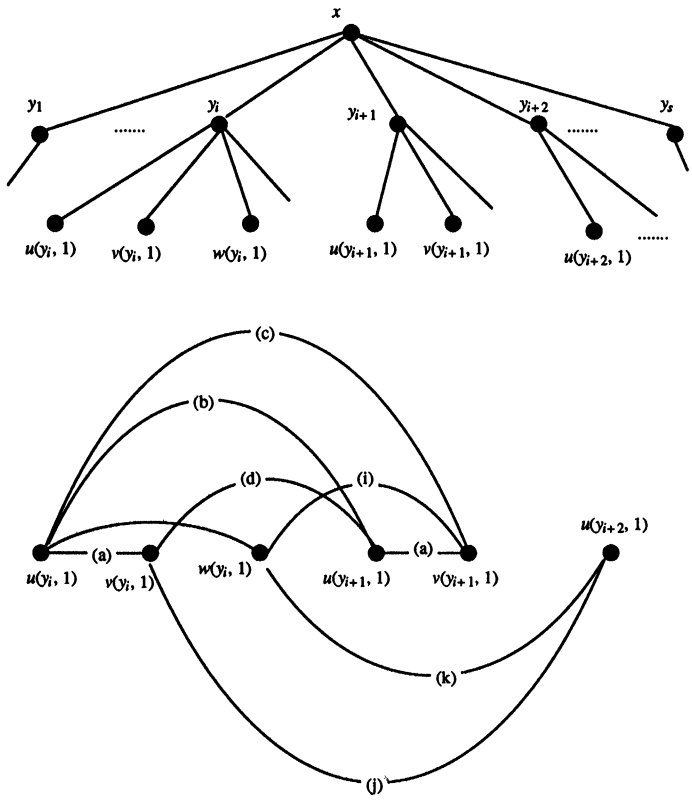


FIG. 3. *Edges in G_n, when $d = 1$, labeled by corresponding part of Lemma* 3.4.

for $k > 0$ and $\alpha \in \Sigma^*$, the lemma can be verified directly for each pair (a)–(k). For example, in (c),

$$\begin{aligned} M[u(y_i, d)] &= M[(\tau\sigma)^d(y_i)] \\ &= M[(\tau\sigma)^d\tau\sigma^i(x)] \\ &= M[(\tau\sigma)^d\tau\sigma^i(0^k1\alpha)] \\ &= M[0^{k-i-d}1\alpha0^{i-1}1^{d+1}] \\ &= 1\alpha0^{i-1}1^{d+1}0^{k-i-d} \end{aligned}$$

and, similarly,

$$\begin{aligned} M[v(y_{i+1}, d)] &= M[\tau\sigma^2(\tau\sigma)^{d-1}(y_{i+1})] \\ &= M[\tau\sigma^2(\tau\sigma)^{d-1}\tau\sigma^{i+1}(0^k1\alpha)] \\ &= M[0^{k-i-d-2}1\alpha0^i1^d01] \\ &= 1\alpha0^i1^d010^{k-i-d-2}. \end{aligned}$$

Exchanging the last 1 in $M[v(y_{i+1}, d)]$ with the last 0 in the block of i zeros gives $M[u(y_i, d)]$. □

The following two lemmas describe strategies which will be used repeatedly in Theorem 3.7 for linking together Hamilton paths/cycles in $G_n[y_i, d-1, 1]$ and $G_n[x, d, i+1]$ to get a Hamilton path/cycle in $G_n[x, d, i]$. Although linking is straightforward in

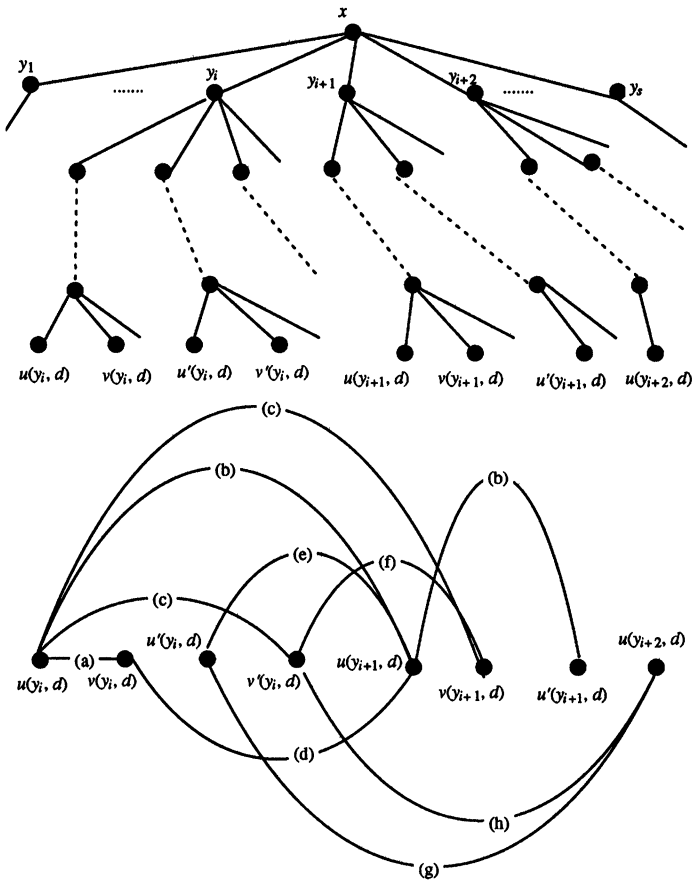


FIG. 4. *Edges in G_n, when $d \geq 2$, labeled by corresponding part of Lemma* 3.4.

the general case, many cases require special attention if d is small or if i is close to $r(x, d)$.

First, define Hamilton paths and cycles of special types in $G_n[x, d, i]$ as follows:

$O(x, d, i)$ cycle: a Hamilton cycle containing edges
$u(y_i, d-1)v(y_i, d-1)$ and $u(y_i, d-1)v(y_{i+1}, d-1)$,

$E(x, d, i)$ cycle: a Hamilton cycle containing edges
$u(y_i, d-1)v(y_i, d-1)$ and $u(y_i, d-1)u(y_{i+1}, d-1)$,

$UV(x, d, i)$ path: a Hamilton path from $u(y_i, d-1)$ to $v(y_i, d-1)$,

$P(x, d, i)$ path: a Hamilton path from $v(y_i, d-1)$ to $u(y_{i+1}, d-1)$,

$Q(x, d, i)$ path: a Hamilton path from $u(y_i, d-1)$ to $u(y_{i+1}, d-1)$.

Note that $O(x, d, i)$ and $E(x, d, i)$ cycles contain $UV(x, d, i)$ paths. The O- and E-type Hamilton cycles are so named because for large enough d, it turns out that $G_n[x, d, i]$ has an O-type cycle for odd d and an E-type cycle for even d. The UV-, P-, and Q-type Hamilton paths are actually cycles, since by Lemma 3.4 their start and end vertices are adjacent. However, they will be used only as paths for splicing

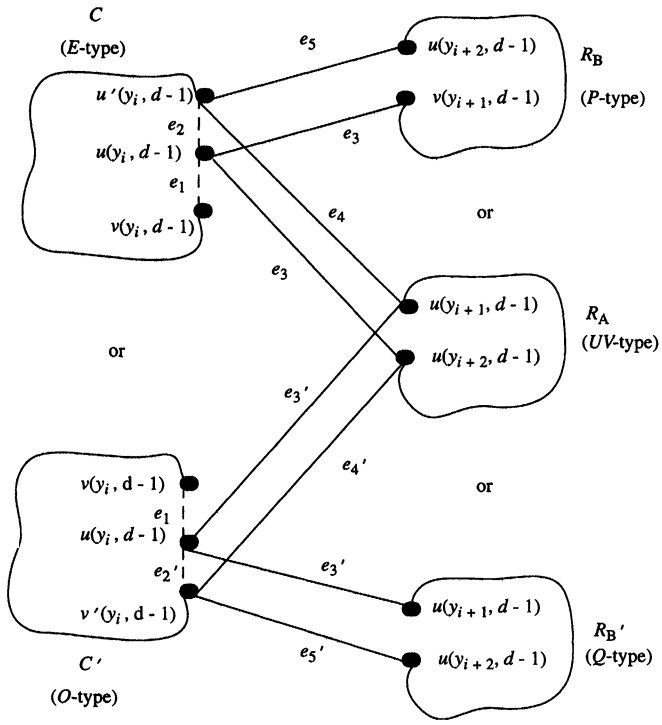


FIG. 5. *Construction of $O(x, d, i)$ and $E(x, d, i)$ cycles in Lemma* 3.5.

into O- and E-type cycles.

The construction of the main theorem splits $G_n[x, d, i]$ into two subgraphs, one of which contains an O- or E-type cycle and the other which contains a Hamilton path of type UV, P, or Q. An O-type cycle for $G_n[x, d, i]$ is then formed by splicing together a cycle and path recursively constructed in the subproblems: either an E-type cycle plus a UV path or an E-type cycle plus a P-type path. Similarly, An E-type cycle for $G_n[x, d, i]$ can be formed by splicing together an O-type cycle plus a UV path or an O-type cycle plus a Q-type path. The details are given in Lemmas 3.5 and 3.6 below.

LEMMA 3.5. *For $x \in L(n, b)$ with $b, d \geq 1$ and $b + d \leq \lfloor n/2 \rfloor$, assume that $r(x, d) \geq 2$ and let y_i be the ith child of x. For $i < r(x, d)$, if $G_n[y_i, d-1, 1]$ has an $E(y_i, d-1, 1)$ (respectively, $O(y_i, d-1, 1)$) cycle, then $G_n[x, d, i]$ has an $O(x, d, i)$ (respectively, $E(x, d, i)$) cycle as long as $G_n[x, d, i+1]$ has the following:*

(A) *a $UV(x, d, i+1)$ path or*

(B) *both $P(x, d, i+1)$ and $Q(x, d, i+1)$ paths.*

Proof. (See Figure 5.) Let C be an $E(y_i, d-1, 1)$ cycle in $G_n[y_i, d-1, 1]$ and for $1 \leq j \leq r(y_i, 1)$, let z_j be the jth child of y_i. Then C contains the edges $e_1 = u(z_1, d-2)v(z_1, d-2) = u(y_i, d-1)v(y_i, d-1)$ and $e_2 = u(z_1, d-2)u(z_2, d-2) = u(y_i, d-1)u'(y_i, d-1)$. We seek a Hamilton cycle in $G_n[x, d, i]$ containing edge e_1 and the edge $e_3 = u(y_i, d-1)v(y_{i+1}, d-1)$, which edge exists by Lemma 3.4(c).

Let R_A be a $UV(x, d, i+1)$ path in $G_n[x, d, i+1]$ from $u(y_{i+1}, d-1)$ to $v(y_{i+1}, d-1)$. Note that $e_4 = u'(y_i, d-1)u(y_{i+1}, d-1)$ is an edge by Lemma 3.4(e). Then $(C - e_2) + e_3 + e_4 + R_A$ is a Hamilton cycle in $G_n[x, d, i]$ containing edges e_1 and e_3.

Let R_B be a $P(x, d, i+1)$ path in $G_n[x, d, i+1]$ from $v(y_{i+1}, d-1)$ to $u(y_{i+2}, d-1)$ and let $e_5 = u'(y_i, d-1)u(y_{i+2}, d-1)$, which is an edge by Lemma 3.4(g). Then

$(C - e_2) + e_3 + e_5 + R_B$ is a Hamilton cycle in $G_n[x, d, i]$ containing edges e_1 and e_3.

On the other hand, if C' is an $O(y_i, d-1, 1)$ cycle in $G_n[y_i, d-1, 1]$, it contains edge e_1 as well as e_2' where $e_2' = u(z_1, d-2)v(z_2, d-2) = u(y_i, d-1)v'(y_i, d-1)$. We seek a Hamilton cycle in $G_n[x, d, i]$ containing edge e_1 and the edge $e_3' = u(y_i, d-1)u(y_{i+1}, d-1)$, which edge exists by Lemma 3.4(b).

With R_A as before, note that $e_4' = v'(y_i, d-1)v(y_{i+1}, d-1)$ is an edge by Lemma 3.4(f). Then $(C' - e_2) + e_3' + e_4' + R_A$ is the required Hamilton cycle in $G_n[x, d, i]$.

Let R_B' be a $Q(x, d, i+1)$ path in $G_n[x, d, i+1]$ from $u(y_{i+1}, d-1)$ to $u(y_{i+2}, d-1)$ and let $e_5' = v'(y_i, d-1)u(y_{i+2}, d-1)$, which is an edge by Lemma 3.4(h). Then $(C - e_2') + e_3' + e_5' + R_B'$ is the required Hamilton cycle in $G_n[x, d, i]$. □

A graph is *trivial* if it has only one vertex.

LEMMA 3.6. *For $x \in L(n, b)$ with $b, d \geq 1$ and $b + d \leq \lfloor n/2 \rfloor$, assume that $r(x, d) \geq 2$ and let $r = r(x, d)$. If $G_n[x, d, r]$ is trivial, then $G_n[x, d, r-1]$ has both $P(x, d, r-1)$ and $Q(x, d, r-1)$ paths as long as $G_n[y_{r-1}, d-1, 1]$ has either*

(A) *a $UV(y_{r-1}, d-1, 1)$ path or*

(B) *both $P(y_{r-1}, d-1, 1)$ and $Q(y_{r-1}, d-1, 1)$ paths.*

Proof. (See Figure 6.) If $G_n[x, d, r]$ is trivial, it contains only the vertex $u(y_r, d-1)$, by Corollary 2.4(a). By Lemma 3.4(d) and (b), respectively, $G_n[x, d, r-1]$ contains the edges $e_1 = v(y_{r-1}, d-1, 1)u(y_r, d-1, 1)$ and $e_2 = u(y_{r-1}, d-1, 1)u(y_r, d-1, 1)$, as long as the endpoints exist. It suffices to show that if (A) or (B) holds, $G_n[x, d, r-1]$ has a Hamilton cycle C_1 containing e_1 and a Hamilton cycle C_2 containing e_2.

If (A) holds, let H be a $UV(y_{r-1}, d-1, 1)$ path in $G_n[y_{r-1}, d, 1]$ from $u(y_{r-1}, d-1)$ to $v(y_{r-1}, d-1)$. Then $C_1 = C_2 = H + e_1 + e_2$ gives both required cycles in $G_n[x, d, r-1]$.

If (B) holds, let z_j be the jth child of y_{r-1} and let H_1 and H_2 be Hamilton paths in $G_n[y_{r-1}, d, 1]$ from $v(z_1, d-2) = v(y_{r-1}, d-1)$ to $u(z_2, d-2) = u'(y_{r-1}, d-1)$ and from $u(z_1, d-2) = u(y_{r-1}, d-1)$ to $u(z_2, d-2) = u'(y_{r-1}, d-1)$, respectively. Let e_3 be the edge $e_3 = u'(y_{r-1}, d-1)u(y_r, d-1)$ which exists by Lemma 3.4(e). Then the required cycles in $G_n[x, d, r-1]$ are $C_1 = H_1 + e_1 + e_3$ and $C_2 = H_2 + e_2 + e_3$. □

We can now prove the main result of the paper.

THEOREM 3.7. *For $x \in L(n, b)$ with $b \geq 1$, let d and i be integers such that $1 \leq d + b \leq \lfloor n/2 \rfloor$ and $1 \leq i \leq r(x, d)$. Then $G_n[x, d, i]$ has a Hamilton cycle whenever $|V(x, d, i)| \geq 3$.*

Proof. We prove the following claim under the hypotheses of the theorem.

CLAIM. Assume $|V(x, d, i)| \geq 2$.

Exception A subclaim. If

(i) $d = 1$, or

(ii) $i = r(x, d)$, or

(iii) $d = 2$ and $V(x, d, i) = \{u(y_i, 1),\ v(y_i, 1),\ u(y_{i+1}, 1),\ v(y_{i+1}, 1)\}$,

then $G_n[x, d, i]$ has a $UV(x, d, i)$ path.

Exception B subclaim. Otherwise, if $i + 1 = r(x, d)$ and $G_n[x, d, i+1]$ is trivial, then $G_n[x, d, i]$ has both $P(x, d, i)$ and $Q(x, d, i)$ paths.

General case. Otherwise, $G_n[x, d, i]$ has an $E(x, d, i)$ cycle when d is even and an $O(x, d, i)$ cycle when d is odd.

Note that the theorem follows from the Claim, since the origin and terminus of any UV, P, or Q path are adjacent by Lemma 3.4.

Proof of the Claim. Let $r = r(x, d)$. The proof is by induction on d and $r - i$. If $d = 1$, then $G_n[x, d, i]$ is complete since it is a subgraph of $G_n[x, 1, 1]$, which is complete by Lemma 3.1. Similarly, when $r - i = 0$ and $d \geq 2$, $G_n[x, d, i]$ is complete by Lemma 3.3. In both cases, by Corollary 2.4(b), $G_n[x, d, i]$ contains $u(y_i, d-1)$ and

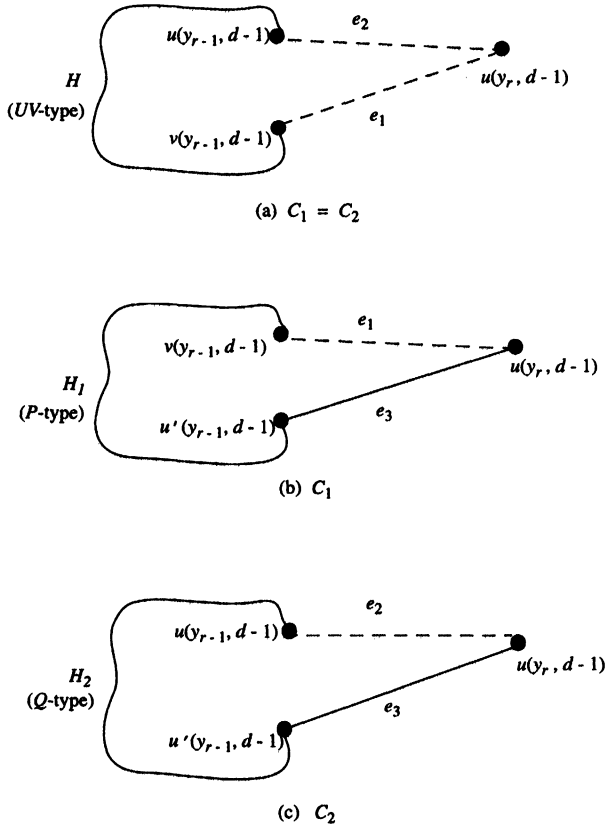


FIG. 6. *Construction of* $P(x, d, r-1)$ *and* $Q(x, d, r-1)$ *paths in Lemma* 3.6.

$v(y_i, d-1)$, and, since it is complete, it contains a Hamilton path joining these two vertices. This satisfies Subclaim A(i, ii) of the Claim.

Assume inductively that when $d \geq 2$ and $r - i \geq 1$, the Claim is true for all (d', i') with both $d' = d-1$ and $r - i' \geq 0$ or both $d' = d$ and $r - i' \leq r - i - 1$. We show the claim is true for (d, i).

Since $r - i \geq 1$, $u(y_{i+1}, d-1) \in L(n)$, so by Lemma 2.3(iii), $u'(y_i, d-1) \in L(n)$. Thus, $r(y_i, d-1) \geq 2$ and by Corollary 2.4(b), $G_n[y_i, d-1, 1]$ contains $u(y_i, d-1)$ and $v(y_i, d-1)$, as well as $u'(y_i, d-1)$.

Case $d = 2$. If $d = 2$ and $r - i \geq 1$, by induction, $G_n[y_i, 1, 1]$ has a $UV(y_i, 1, 1)$ path H from $u(y_i, 1)$ to $v(y_i, 1)$. By Lemma 3.4(a,b), $G_n[x, 2, i]$ contains the edges $e_1 = u(y_i, 1)$ to $v(y_i, 1)$ and $e_2 = u(y_i, 1)$ to $u(y_{i+1}, 1)$.

If $G_n[x, 2, i+1]$ is trivial (Exception B), let e_3 be the edge $e_3 = v(y_i, 1)$ to $u(y_{i+1}, 1)$, which exists by Lemma 3.4(d). Then $H + e_2 + e_3$ is a Hamilton cycle in $G_n[x, 2, i]$ containing both $P(x, 2, i)$ and $Q(x, 2, i)$ paths, as required for Exception B. (See Figure 7(a).)

If $G_n[x, 2, i]$ is Exception A(iii) of the Claim, edges $u(y_{i+1}, 1)v(y_{i+1}, 1)$ and $u(y_i, 1)v(y_{i+1}, 1)$ exist by Lemma 3.4(a,c). These two edges together with e_3 give a $UV(x, 2, i)$ path from $u(y_i, 1)$ to $v(y_i, 1)$, as required. (See Figure 7(b).)

Otherwise, $G_n[x, 2, i]$ is none of the exceptions, so, since $d = 2$ is even, we must show it has an $E(x, 2, i)$ cycle containing e_1 and e_2. By induction, $G_n[x, 2, i+1]$ has

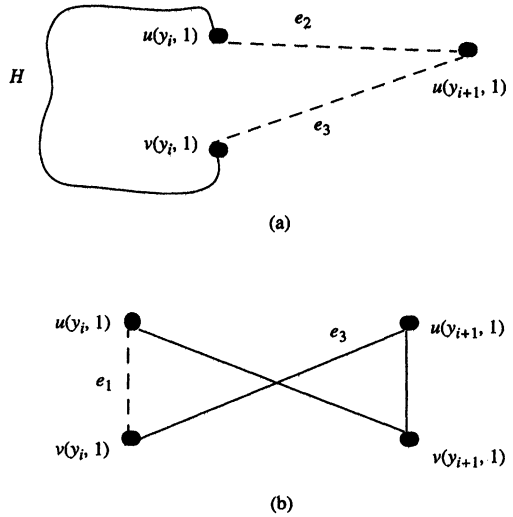


FIG. 7. *Special cases when* $d = 2$ *in Theorem* 3.7.

either a Hamilton path R_1 from $u(y_{i+1}, d-1)$ to $u(y_{i+2}, d-1)$ (Exception B) or a Hamilton path R_2 from $u(y_{i+1}, d-1)$ to $v(y_{i+1}, d-1)$ (Exception A and General case).

If $r(y_i, 1) = 2$, then by Lemma 2.3(v), $v(y_{i+2}, 1) \notin L(n)$ and therefore $G_n[x, 2, i+1]$ is Exception B, so it contains R_1. In this case, let e_4 be the edge $v(y_i, 1)\, u(y_{i+2}, 1)$, which exists by Lemma 3.4(j). Then $e_1 + e_2 + R_1 + e_4$ is the required Hamilton cycle. (See Figure 8(a).)

If $r(y_i, 1) \geq 3$, then it also contains vertex $u''(y_i, 1) = w(y_i, 1)$, by Lemma 2.3. Thus, since $G_n[y_i, 1, 1]$ is complete, it contains a Hamilton path H' from $u(y_i, 1)$ to $w(y_i, 1)$ containing e_1. Let e_5 and e_6 be the edges

$$\begin{aligned} e_5 &= w(y_i, 1)u(y_{i+2}, 1), \\ e_6 &= w(y_i, 1)v(y_{i+1}, 1), \end{aligned}$$

which exist by Lemma 3.4(k, i). Since $G_n[x, 2, i+1]$ contains either R_1 or R_2, then one of the following is a Hamilton cycle in $G_n[x, 2, i]$ containing e_1 and e_2: $H' + e_2 + R_1 + e_5$ or $H' + e_2 + R_2 + e_6$. (See Figure 8(b,c).)

Case $d = 3$. In this case, $G_n[x, 3, i]$ cannot fall under any case of Exception A of the Claim. If $G_n[x, 3, i]$ is Exception B, then $i + 1 = r(x, 3)$ and $G_n[x, 3, i+1]$ is trivial. We must find $P(x, 3, i)$ and $Q(x, 3, i)$ paths in $G_n[x, 3, i]$. But these paths exist by Lemma 3.6 since, by induction, $G_n[y_i, 2, 1]$ satisfies conditions (A) and (B) of that lemma.

Otherwise, $G_n[x, 3, i]$ is the general case of the Claim and since $d = 3$ is odd, we must find an $O(x, 3, i)$ cycle containing edges $e_1 = u(y_i, 2)v(y_i, 2)$ and $e_2 = u(y_i, 2)v(y_{i+1}, 2)$. Since $G_n[x, 3, i+1]$ is nontrivial, by induction, it satisfies (A) or (B) of Lemma 3.5.

If $G_n[y_i, 2, 1]$ is in the general case of the Claim, by induction it has an $E(y_i, 2, 1)$ cycle and therefore by Lemma 3.5, $G_n[x, 3, i]$ has an $O(x, 3, i)$ cycle. Otherwise, $G_n[y_i, 2, 1]$ is not the general case of the Claim. But $G_n[x, 3, i+1]$ is nontrivial, so by Corollary 2.4(b) it contains $v(y_{i+1}, 2)$. But then by Lemma 2.3(iv), $v'(y_i, 2) \in V(y_i, 2, 1)$. Clearly, then, $G_n[y_i, 2, 1]$ cannot be Exception B or cases (i) or (ii) of

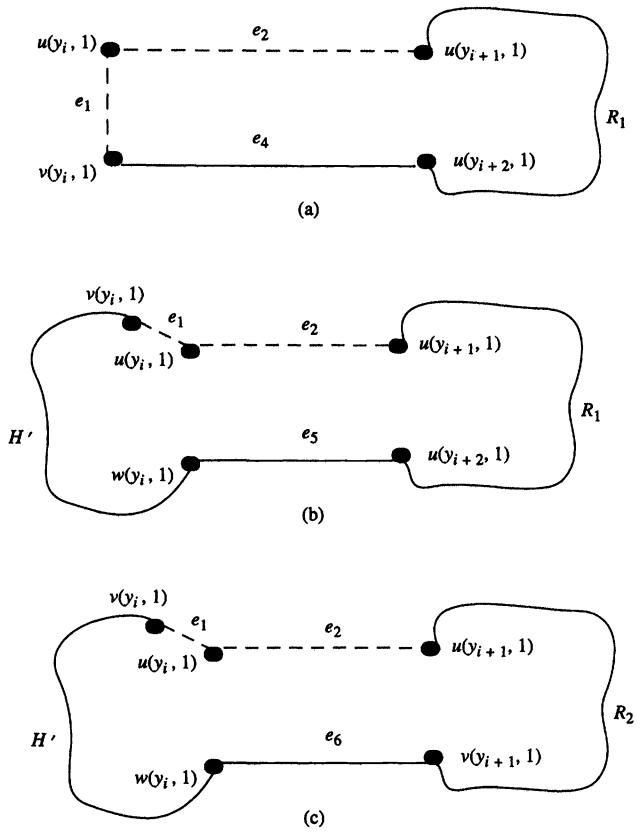


FIG. 8. *The $d = 2$ case of Theorem* 3.7.

Exception A. So, suppose $G_n[y_i, 2, 1]$ is Exception A(iii). Then $G_n[y_i, 2, 1]$ contains only the vertices $\{u(y_i, 2),\ v(y_i, 2),\ u'(y_i, 2),\ v'(y_i, 2)\}$. Since $u''(y_i, 2) \notin V(y_i, 2, 1)$, by Lemma 2.3(v), $v(y_{i+2}, 2) \notin L(n)$. Thus either $r = i + 2$ and $G_n[x, 3, i + 1]$ is Exception B or $r = i + 1$ and therefore by Lemma 3.2, $G_n[x, 3, i + 1]$ has vertex set either $\{u(y_{i+1}, 2),\ v(y_{i+1}, 2)\}$ or $\{u(y_{i+1}, 2),\ v(y_{i+1}, 2),\ u'(y_{i+1}, 2)\}$. ($G_n[x, 3, i + 1]$ trivial was considered earlier.) If $G_n[x, 3, i + 1]$ is Exception B, then by induction it has a Hamilton path R from $v(y_{i+1}, 2)$ to $u(y_{i+2}, 2)$. See Figure 9(a) for an $O(x, 3, i)$ cycle in this case. If $V(x, 3, i + 1) = \{u(y_{i+1}, 2),\ v(y_{i+1}, 2),\ u'(y_{i+1}, 2)\}$, it can be verified that $v(y_i, 2)$ and $u'(y_{i+1}, 2)$ are M-adjacent and Figure 9(b) shows an $O(x, 3, i)$ cycle in this case. An $O(x, 3, i)$ cycle in the remaining case that $V(x, 3, i + 1) = \{u(y_{i+1}, 2),\ v(y_{i+1}, 2)\}$ is shown in Figure 9(c), where edges exist by Lemma 3.5.

Case $d \geq 4$. If $G_n[x, d, i]$ is Exception B of the Claim, then $i + 1 = r(x, d)$ and $G_n[x, d, i+1]$ is trivial. By induction, $G_n[y_{i-1}, d-1, 1]$ satisfies (A) and (B) of Lemma 3.6, so in this case, by Lemma 3.6, $G_n[x, d, i]$ has both $P(x, d, i)$ and $Q(x, d, i)$ paths, as required.

Otherwise, $G_n[x, d, i+1]$ is nontrivial, so by Corollary 2.4(b) it contains $v(y_{i+1}, d-1)$ and therefore by Lemma 2.3(iv), $v'(y_i, d-1) \in L(n)$. Thus, $G_n[y_i, d-1, 1]$ is in the general case of the Claim, so by induction, it has an $E(y_i, d-1, 1)$ cycle if $d-1$ is even and an $O(y_i, d-1, 1)$ cycle if $d-1$ is odd. Furthermore, since $G_n[x, d, i+1]$ is nontrivial, it satisfies (A) or (B) of Lemma 3.5 and therefore by Lemma 3.5, $G_n[x, d, i]$

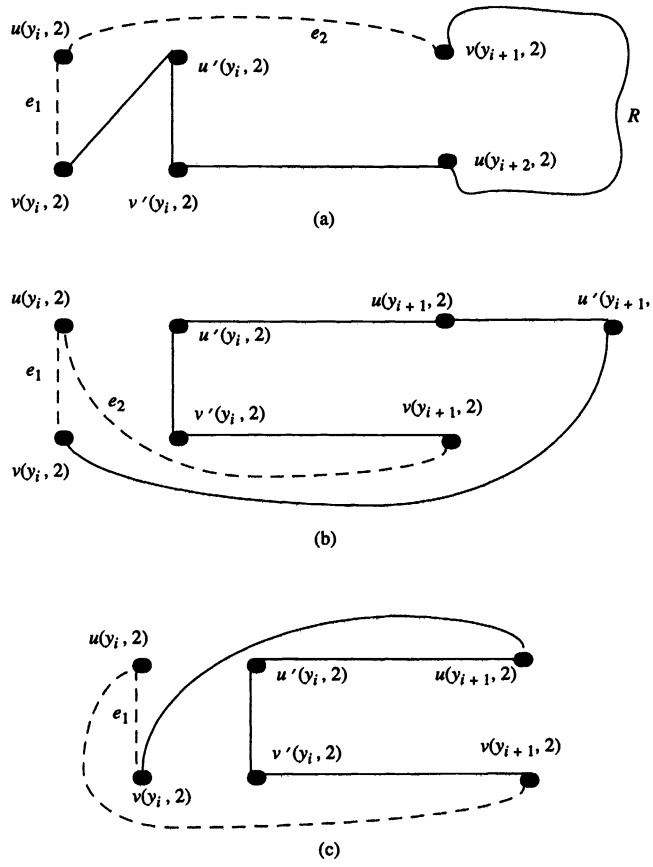


FIG. 9. *The $d = 3$ case of Theorem* 3.7.

has an $E(x, d, i)$ cycle if d is even and an $O(x, d, i)$ cycle if d is odd. This completes the proof of the theorem. □

An example of the construction of the theorem is shown in Figure 10. The tree of 10-bit necklaces is shown at the top, from levels 0 through 5. In the center is the Hamilton cycle C in $G_{10}[0^{10}, 5, 1]$, resulting from the construction of the theorem. At the bottom is the Gray code obtained by replacing each x on C by $M[x]$.

4. The algorithm. The proof of Theorem 3.7 gives a recursive procedure for constructing a Gray code for necklaces of fixed density. The procedure has been implemented in C and is included in the appendix to [Wan]. (A subsequent modification requires only O(n) storage.) In this section, we show the time required is $O(nN(n, d))$, where $N(n, d)$ is the number of n-bit necklaces of density d.

Below, we give a crude outline of the procedure CYCLE(x, d, i) for constructing a Hamilton cycle in the graph $G_n[x, d, i]$. For simplicity, we ignore differences between the different types of cycles (O, E) since these do not affect the time analysis.

```
CYCLE(x,d,i).

1.   compute y_i, the ith child of x
2.   if d=1 then complete graph
3.   else if i=r(x,d) then complete graph
```

```
     else if d=2 then
4.      if G[x,d,i+1] is trivial then link cycle in complete graph
           G[y_i,d-1,1] with vertex as in Figure 7(a)
5.      else if G[x,d,i] is exception A(iii) then construct cycle
           as in Figure 7(b)
6.      else if r(y_i,d-1)=2 then link vertices to CYCLE(x,d,i+1)
           as in Figure 8(a)
7.      else link CYCLE(y_i,d-1,1) with CYCLE(x,d,i+1) as in
           Figures 8(b,c)
     else if d=3 then
8.      if G[x,d,i] is Exception B then link CYCLE(y_i,d-1,1) with
           vertex as in Lemma 3.6
9.      else if G[y_i,d-1,1] is Exception A(iii) then
10.        if G[x,d,i+2] is empty then construct cycle
              as in Figure 9(b,c)
11         else link vertices to CYCLE(x,d,i+1) as in Figure 9(a)
12.     else link CYCLE(y_i,d-1,1) and CYCLE(x,d,i+1) as in Lemma 3.5
     else {d >= 4 }
13.     if G[x,d,i] is Exception B then link CYCLE(y_i,d-1,1) and
           vertex as in Lemma 3.6
14.     else link CYCLE(y_i,d-1,1) with CYCLE(x,d,i+1) as in Lemma 3.5
```

Recall that $L(n)$ is the set of lexicographically smallest representatives of the n-bit necklaces. It is shown in [Shi] that for an arbitrary n-bit string, x, it is possible to check whether $x \in L(n)$ in time $O(n)$. Using this fact, we show that all the tests in the CYCLE algorithm can be made in time $O(n)$.

First note that by definition of $r(x,d)$ and Corollary 2.4(a), $r(x,d) \geq t$ if and only if $u(y_t, d-1) \in L(n)$, so the tests on lines 3 and 6 can be made in time $O(n)$ using the algorithm of [Shi]. By Corollary 2.4(a), $G_n[x,d,i]$ is empty if $u(y_i, d-1) \notin L(n)$ and is trivial if and only if $u(y_i, d-1) \in L(n)$ but $v(y_i, d-1) \notin L(n)$. By Corollary 2.4(b) and Lemma 2.3(iii), $G_n[x,d,i]$ has only two vertices if and only if $v(y_i, d-1) \in L(n)$ but $w(y_i, d-1), u'(y_i, d-1) \notin L(n)$. Thus, tests in lines 4 and 10 take time $O(n)$. Finally, $G_n[x,d,i]$ is Exception B if and only if $i = r(x,d) - 1$ and $G_n[x,d,i+1]$ is trivial; $G_n[x,d,i]$ is Exception A(iii) if and only if $d = 2$, $r(y_i, d-1) = 2$, and $r(y_{i+1}, d-1) = 2$. Thus, tests on lines 5, 8, 9, and 13 can be done in time $O(n)$ by testing whether certain binary strings are in $L(n)$. This means that the CYCLE procedure spends no more than $O(n)$ time to determine whether to make a recursive call.

If CYCLE(x,d,i) makes no recursive call, it takes one of the branches 2, 3, 4, 5, or 10, each of which can be implemented in time $O(n|V(x,d,i)|)$.

If CYCLE(x,d,i) makes a recursive call, it does so to one or both of the disjoint subgraphs $G_n[y_i, d-1, 1]$ (a left call) and $G_n[x,d,i+1]$ (a right call) and never to a trivial graph. To count the number of recursive subcalls over the entire execution, consider the subtree $RICH(x,d,i)$ of $TREE(n)$ consisting of all nodes x with descendants in $V(x,d,i)$. Note that no recursive call is made on any node not in $RICH(x,d,i)$. Further, if w is a nonleaf node with only one child in $RICH(x,d)$, no recursive call is made at w (line 2 or 3). Thus, the number of recursive calls is at most the number of nodes in $RICH(x,d,i)$ with at least two children and this number cannot exceed the number of leaves of $RICH(x,d,i)$, which is $|V(x,d,i)|$.

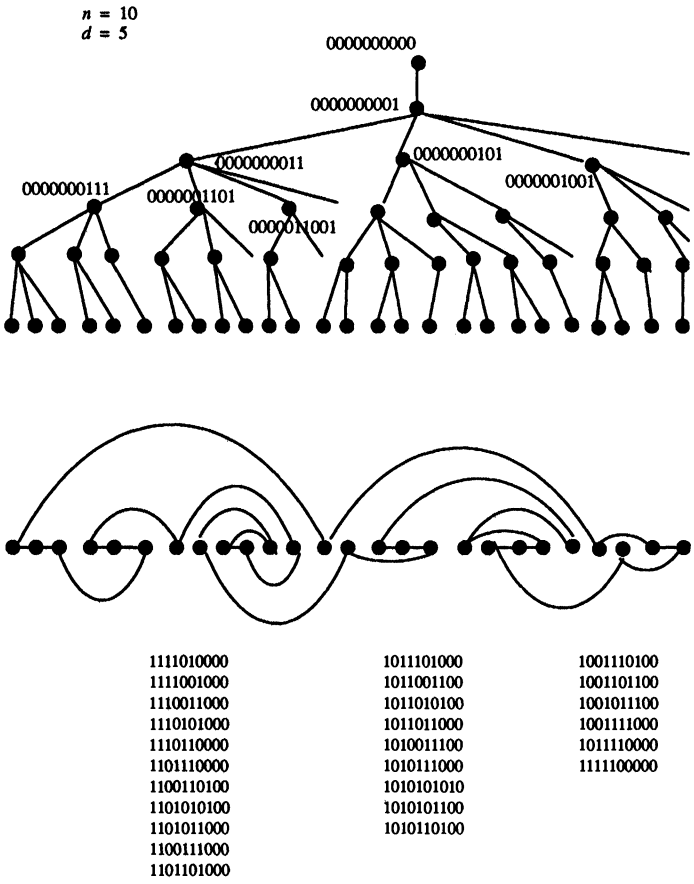


FIG. 10. *The Hamilton cycle in* $G_{10}[0^{10}, 5, 1]$ *and the corresponding Gray code for* 10*-bit necklaces of density* 5.

In summary, the total time for CYCLE(x, d, i) is

$$O(n * \text{ number of recursive calls } + \sum n * |V(z, d', j)|),$$

where the sum on the right is over every call CYCLE(z, d', j) which does not itself make a recursive call. Both terms are $O(n|V(x, d, i)|)$.

In particular, for $d \geq 2$, CYCLE$(x, d-1, 1)$ with $x = 0^{n-1}1$ gives a Gray code for $V(0^{n-1}, d-1, 1) = L(n, d)$ in time $O(n * N(n, d))$.

We mention that to avoid storing cycles, when recursively constructed "left" and "right" cycles are to be linked by two edges, the procedure recursively computes and outputs the left cycle (in an appropriate order), then one link edge is output, and then the right cycle is computed recursively and output (in an appropriate order). To complete the cycle, the second link edge is output. Thus the additional storage required is no more than the depth of the recursion which is $O(n)$.

Although the time analysis of the algorithm can be made tighter in several places, we have found no way to reduce the overall time bound of $O(n * N(n, d))$, either by a tighter analysis or by an alternative implementation. Even for the simpler problem of listing n-bit necklaces of fixed density d in *any* order, no asymptotically faster algorithm is yet known.

Acknowledgment. We are grateful to the anonymous referee who offered many suggestions to improve the presentation.

REFERENCES

[BuWi] M. Buck and D. Wiedemann, *Gray codes with restricted density*, Discrete Math., 48 (1984), pp. 163–171.

[ChChCh] C. C. Chang, H. Y. Chen, and C. Y. Chen, *Symbolic Gray code as a data allocation scheme for two-disc systems*, Comput. J., 35 (1992), pp. 299–305.

[ChLeDu] C. C. Chang, R. C. T. Lee, and M. W. Du, *Symbolic Gray code as a perfect multiattribute hashing scheme for partial match queries*, IEEE Trans. Software Engineering, 8 (1982), pp. 235–249.

[FrKe] H. Fredricksen and I. J. Kessler, *An algorithm for generating necklaces of beads in two colors*, Discrete Math., 61 (1986), pp. 181–188.

[FrMa] H. Fredricksen and J. Maiorana, *Necklaces of beads in k colors and k-ary de Bruijn sequences*, Discrete Math., 23 (1978), pp. 207–210.

[Gar] M. Gardner, *The curious properties of the Gray code and how it can be used to solve puzzles*, Scientific American, 227 (1972), pp. 106–109.

[Gra] F. Gray, *Pulse Code Communication*, U. S. Patent 2632058, March 17, 1953.

[Joh] S. M. Johnson, *Generation of permutations by adjacent transpositions*, Math. Comput., 17 (1963), pp. 282–285.

[Kay] R. Kaye, *A Gray code for set partitions*, Inform. Process. Lett., 5 (1976), pp. 171–173.

[Kli] P. Klingsberg, *A Gray code for compositions*, J. Algorithms, 3 (1982), pp. 41–44.

[Los] R. M. Losee, *A Gray code based ordering for documents on shelves*, J. Amer. Soc. Inform. Sci., 43 (1992), pp. 312–322.

[Luc] J. M. Lucas, *The rotation graph of binary trees is hamiltonian*, J. Algorithms, 8 (1987), pp. 503–535.

[LuRoRu] J. M. Lucas, D. Roelants van Baronaigien, and F. Ruskey, *On rotations and the generation of binary trees*, J. Algorithms, 15 (1993), pp. 343–366.

[NiWi] A. Nijenhuis and H. S. Wilf, *Combinatorial Algorithms for Computers and Calculators*, Academic Press Inc., New York, 1978.

[PrRu1] G. Pruesse and F. Ruskey, *Generating the linear extensions of certain posets by transpositions*, SIAM J. Discrete Math., 4 (1991), pp. 413–422.

[PrRu2] G. Pruesse and F. Ruskey, *Generating linear extensions fast*, SIAM J. Comput., 23 (1994), pp. 373–386.

[RaSaWe] D. Rasmussen, C. Savage, and D. West, *Gray codes for families of integer partitions*, J. Combin. Theory Ser. A, 70 (1995), pp. 201–229.

[Ric] D. Richards, *Data compression and Gray-code sorting*, Inform. Process. Lett., 22 (1986), pp. 201–205.

[Rus1] F. Ruskey, *Adjacent interchange generation of combinations*, J. Algorithms, 9 (1988), pp. 162–180.

[Rus2] F. Ruskey, *Generating linear extensions of posets by transpositions*, J. Combin. Theory Ser. B, 54 (1992), pp. 77–101.

[RuPr] F. Ruskey and A. Proskurowski, *Generating binary trees by transpositions*, J. Algorithms, 11 (1990), pp. 68–84.

[RuSaWa] F. Ruskey, C. D. Savage, and T. M. Wang, *Generating necklaces*, J. Algorithms, 13 (1992), pp. 414–430.

[Sav] C. D. Savage, *Gray code sequences of partitions*, J. Algorithms, 10 (1989), pp. 577–595.

[Shi] Y. Shiloach, *Fast canonization of circular strings*, J. Algorithms, 2 (1981), pp. 107–121.

[Sta] G. Stachowiak, *Hamilton paths in graphs of linear extensions for unions of posets*, SIAM J. Discrete Math., 5 (1992), pp. 199–206.

[Tro] H. F. Trotter, *PERM (Algorithm 115)*, Comm. ACM, 5 (1962), pp. 434–435.

[Wes] D. B. West, *Generating linear extensions by adjacent transpositions*, J. Combin. Theory Ser. B, 57 (1993), pp. 58–64.

[WaSa] T. M. Wang and C. D. Savage, *A New Algorithm for Generating Necklaces*, Technical report TR-90-20, Department of Computer Science, North Carolina State University, Raleigh, NC, 1990.

[Wan] T. M. Wang, *Gray Codes for Necklaces of Fixed Density*, Ph.D. thesis, Department of Computer Science, North Carolina State University, Raleigh, NC, 1994.

[Wil] H. S. Wilf, *Combinatorial Algorithms: An Update*, SIAM, Philadelphia, 1989.

SIAM J. DISCRETE MATH.
Vol. 9, No. 4, pp. 674–681, November 1996

011

FINDING INDEPENDENT SETS IN TRIANGLE-FREE GRAPHS*

KATHRYN FRAUGHNAUGH† AND STEPHEN C. LOCKE‡

Abstract. Finding a maximum independent set in a graph is well known to be an NP-complete problem. Here an $O(n^2)$-time algorithm that finds an independent set of order at least $(6n-m)/13$ in a triangle-free graph with n vertices and m edges is presented. A tight lower bound on independence in 4-regular triangle-free graphs is $4n/13$, so the bound is sharp for this class.

Key words. independence, triangle-free, algorithm, maximum degree four

AMS subject classification. 05C

1. Introduction. In general we follow the notation and terminology of [2]. A graph $G = (V, E)$ is simple and loop-free. The maximum and minimum degrees of G are denoted $\Delta(G)$ and $\delta(G)$, respectively. The degree of a vertex v is $d(v)$, the set of neighbors of v is $N(v)$, and the closed neighborhood of v is $N[v]$. The graph induced by the vertices of $V - N[v]$ is $H(v)$. The sum of the degrees of the vertices of $N(v)$ is denoted $s(v)$. A *triangle-free* graph is one that contains no complete graph on three vertices. A set of vertices is *independent* if no two of them are adjacent. The *independence* of a graph is the maximum cardinality of an independent set.

In [5], Fraughnaugh showed that every triangle-free graph with maximum degree at most 4, n vertices, m edges, and independence α satisfies $m \geq 6n - 13\alpha$. It follows from this inequality that the independence in such a graph is always at least $4n/13$. This is best possible for this class. In [7], Kreher and Radziszowski demonstrated that this same inequality holds for all triangle-free graphs, without being aware of the earlier work in [5]. We will develop an algorithm to find independent sets in triangle-free graphs relying heavily on the proof in [5]. Although there is a close relationship between induction in mathematics and recursion in algorithm design, the proof in [5] does not yield an explicit algorithm. Crucial parts of the proof depend upon the existence of sufficiently large independent sets in certain subgraphs but give no means of finding them. Thus the proof of algorithm correctness provides a new proof of both Fraughnaugh's and Kreher and Radziszowski's results.

For ease of proof, we introduce the algorithm in three parts. The most difficult part of the proof of correctness is for the algorithm applied to triangle-free graphs with maximum degree at most 4, which we present in §2. In §3 we extend the algorithm to triangle-free graphs with no degree restrictions. In §4 we discuss the computational complexity of all algorithms.

2. The algorithm in graphs with maximum degree four. The algorithm that we are about to describe must handle 3-regular graphs as a special case. We will use an algorithm, built from two algorithms of Bondy and Locke, to find independent sets of order at least $7n/20$ in triangle-free cubic graphs of order n. First, in [1], an algorithm is given for constructing a bipartite subgraph of a triangle-free graph G with maximum degree at most 3 that has at least $4|E|/5$ edges. We refer the reader to that paper for a description of the algorithm. Then, in Theorem 1 of [8], Bondy and Locke give an algorithm to construct an independent set in a 3-regular triangle-free

* Received by the editors April 5, 1994; accepted for publication (in revised form) December 28, 1995.

† University of Colorado at Denver, Denver, CO 80217 (kfraughn@carbon.cudenver.edu).

‡ Florida Atlantic University, Boca Raton, FL 33431 (lockes@acc.fau.edu).

graph G from a given bipartite subgraph B of G. Although the algorithm described in [8] assumes that $|E(B)|$ is maximum, all that is necessary for the argument is that the subgraph induced by B contain no vertices of degree 0 or 1 in B. The bipartite subgraph constructed in [1] has no such vertices. Let $B = (X, Y)$ be the bipartite subgraph of G constructed by the algorithm of [1]. Let $X_i = \{v \in X : d_B(v) = i\}$, $Y_i = \{v \in Y : d_B(v) = i\}$, $x_i = |X_i|$, and $y_i = |Y_i|$. We note that X_0, X_1, Y_0, and Y_1 are all empty. Assume that X has been chosen so that $x_2/2 + x_3 \geq y_2/2 + y_3$. The vertices of X_2 split into at most two independent sets, so let X_2' be the larger of these sets. Then $\alpha \geq x_2/2+x_3 \geq (1/2)(x_2/2+x_3+y_2/2+y_3) = (1/4)(x_2+2x_3+y_2+2y_2) = (1/4)(2x_2 + 3x_3 + 2y_2 + 3y_3 - (x_2 + x_3 + y_2 + y_3)) = (1/4)(2|E(B)| - 2|E|/3) = (1/4)(3|E(B)|/|E|-1)(2|E|/3) = (1/4)(3|E(B)|/|E|-1)|V|$. Since $|E(B)|/|E| \geq 4/5$, the combined algorithm finds an independent set of order at least $7n/20$ in a triangle-free cubic graph of order n. We refer to the combined algorithm as the *Bondy–Locke algorithm.*

Bondy and Locke showed that the only graphs for which the algorithm in [1] can possibly construct a bipartite subgraph of size only $4|E|/5$ are the Petersen graph and the dodecahedron. Thus these are the only graphs for which the Bondy–Locke algorithm can possibly construct an independent set of order exactly $7n/20$. However, the Petersen graph cannot have a subset of V with exactly $7n/20 = 7/2$ vertices. This leaves the dodecahedron as the only graph in which the Bondy–Locke algorithm might construct an independent set of order $7n/20$.

Now we give the algorithm to find independent sets in graphs with maximum degree at most 4. It treats 3-regular graphs as a special case. When the graph is not 3-regular, the algorithm is basically a greedy algorithm where the choice of the next vertex to add to the independent set under construction is among vertices of the minimum degree.

ALGORITHM INDEPENDENT SET FOR MAXIMUM DEGREE 4 (IS4)

Input: A triangle-free graph H with maximum degree at most 4.
Output: An independent set I.
Set $I = \emptyset$.
For each component G of H do
  Set V = vertex set of G.
  While $V \neq \emptyset$ do
    If G is 3-regular then
      apply the Bondy–Locke algorithm to get an independent set I'.
      Set $I = I \cup I'$.
      Set $V = \emptyset$.
    else
      If $\delta(G) \leq 3$ then
        choose v with degree $\delta(G)$ and with $s(v)$ as large as
          possible
      else (G is 4-regular)
        if G contains a 4-cycle then choose v in the 4-cycle
        else choose any v.
      Set $I = I \cup \{v\}$.
      Set $V = V - N[v]$.
      Set G = the subgraph induced by V.

Before proving that the algorithm finds independent sets that are appropriately large, we examine its performance on cycles.

LEMMA 2.1. *Let G be a graph with n vertices and e edges. If G is the 5-cycle, Algorithm* IS4 *(hereafter referred to as "IS4") constructs an independent set of order $(6n-e+1)/13$. If G is any other cycle, the algorithm constructs an independent set of order greater than $(6n-e+2)/13$.*

Proof. It is easy to verify that IS4 constructs a maximum independent set in a cycle. If $n = 2k$ and $k \geq 2$, then IS4 constructs an independent set of order k, and $k > (10k+2)/13 = (6n-e+2)/13$. If $n = 2k+1$ and $k \geq 3$, then IS4 constructs an independent set of order k, and $k > (10k+7)/13 = (6n-e+2)/13$. For the 5-cycle, the algorithm constructs a 2-element independent set, and $(6n-e+1)/13 = 2$. □

In order to prove that the algorithm constructs independent sets that are large enough, we need to prove a stronger statement.

THEOREM 2.2. *For every triangle-free graph G with n vertices, e edges, and maximum degree at most four, the algorithm* IS4 *finds an independent set I with*

$$|I| \geq \frac{1}{13}(6n-e).$$

Moreover, $|I| \geq (6n-e+1)/13$, unless either G is 4-regular or $\delta(G) = 3$, $\Delta(G) = 4$ and G contains a 4-cycle.

In addition, if $\delta(G) \leq 2$, then $|I| \geq (6n-e+2)/13$ unless G is a 5-cycle, or G is the disjoint union of a 5-cycle and a 4-regular graph, or $\delta(G) = 2$ and G contains a 4-cycle.

Proof. The proof is by induction on the number of vertices of G. The verification of the correctness of the algorithm is trivial for a graph with one vertex. Assume all statements of the theorem for graphs of order less than n and consider a triangle-free graph G with n vertices, e edges, and maximum degree at most 4.

First suppose G is disconnected. Let G_1 be a component of G with $\delta(G_1) = \delta(G)$, and let $G_2 = G - G_1$. Let n_i and e_i be the number of vertices and edges in G_i for $i = 1, 2$. Then by the induction hypothesis, IS4 constructs an independent set I_2 in G_2 with $|I_2| \geq (6n_2 - e_2 + 1)/13$ unless G_2 is 4-regular, or G_2 has minimum degree 3, maximum degree 4, and contains a 4-cycle. In any case, $|I_2| \geq (6n_2 - e_2)/13$.

If $\delta(G) = \delta(G_1) \leq 2$, then IS4 finds an independent set I_1 in G_1 with $|I_1| \geq (6n_1 - e_1 + 2)/13$, unless G_1 is a 5-cycle or $\delta(G_1) = 2$ and G_1 contains a 4-cycle. In the exceptional cases, $|I_1| \geq (6n_1 - e_1 + 1)/13$. Then the independent set I constructed by the algorithm has order $|I_1| + |I_2| \geq (6n-e+2)/13$, unless G_1 is a 5-cycle and G_2 is 4-regular, or G contains a 4-cycle. In both exceptional cases, $|I| \geq (6n-e+1)/13$, as desired.

If $\delta(G) = \delta(G_1) \geq 3$, then IS4 constructs I_1 with $|I_1| \geq (6n_1 - e_1 + 1)/13$, unless G_1 is 4-regular or G_1 is nonregular and contains a 4-cycle, and in both of these cases $|I_1| \geq (6n_1 - e_1)/13$. Now $|I| \geq (6n-e+1)/13$ with the only case in which we get $|I| = (6n-e)/13$ being when either both G_1 and G_2, and also G, are 4-regular, or G_1, and hence G, is a nonregular graph that contains a 4-cycle. This establishes all statements of the theorem for G disconnected.

Now suppose G is connected and consider the various possibilities for the minimum degree of G.

Case 1. $\delta(G) = 1$. Then IS4 chooses a vertex v with degree 1. At the next step, the input to IS4 is a graph G' with $n-2$ nodes and at most $e-1$ edges. Thus by induction IS4 constructs an independent set I' in G' with $|I'| \geq (6(n-2) - (e-1))/13$ so that $|I| = |I'| + 1 \geq (6n-e+2)/13$.

Case 2. $\delta(G) = 2$. If IS4 chooses v with $s(v) = 4$, then G is a cycle and the result follows from Lemma 2.1. So we may assume that IS4 chooses v with $s(v) \geq 5$. At the next step the input to the algorithm is a graph G' with $n - 3$ nodes and at most $e - s(v)$ edges. Thus by induction $|I'| \geq (6(n-3) - (e - s(v)))/13$ and $|I| \geq (6n - e + s(v) - 5)/13$.

If $s(v) = 5$ (and, by choice, $s(v)$ is maximum among all vertices of degree 2), then v has a neighbor w of degree 2 with $s(w) \leq 5$. Thus G' has minimum degree at most 2 and by the induction hypothesis $|I'| \geq (6(n-3) - (e-5) + 1)/13$. Thus $|I| \geq (6n-e+1)/13$. Moreover, if $|I| < (6n-e+2)/13$, then $|I'| < (6(n-3)-(e-5)+2)/13$ and since $\delta(G') \leq 2$, either G' contains a 4-cycle, is a 5-cycle, or is the disjoint union of a 5-cycle and a 4-regular graph. Since G is connected, G' cannot have a 4-regular component or else so does G. If G' is a 5-cycle and w is the neighbor of v that has degree 3, then w's two neighbors in the 5-cycle are nonadjacent or else there is a triangle. Thus w and its neighbors in the 5-cycle belong to a 4-cycle. Hence, G contains a 4-cycle.

If $s(v) = 6$, then $|I| \geq (6n - e + 1)/13$ with $|I| \leq (6n - e + 2)/13$ only if $|I'| < (6(n-3) - (e-6) + 1)/13$. By induction, since G' cannot be 4-regular, G' contains a 4-cycle and so does G. Finally, if $s(v) \geq 7$, then $|I| \geq (6n-e+2)/13$. This establishes all statements of the theorem when $\delta(G) = 2$.

Case 3. $\delta(G) = 3$. If G is 3-regular, then we use the Bondy–Locke algorithm to find an independent set of order at least $7n/20$. It is easy though tedious to show that $\lceil 7n/20 \rceil \geq (9n+2)/26 = (6n - e + 1)/13$ for all even $n \geq 4$.

If G is not 3-regular, then there is a vertex v with degree 3 and $s(v) \geq 10$, which IS4 chooses. At the next step the input to IS4 is a graph G' with $n - 4$ vertices and $e - s(v)$ edges. Suppose that IS4 chooses v with $s(v) = 10$. Then v has a neighbor w with degree 3. If both neighbors of w in G' have degree 4, then $s(w) = 11$, which contradicts the choice of v. Thus $\delta(G') \leq 2$ and by induction $|I'| \geq (6(n-4)-(e-10)+1)/13$, which yields $|I| \geq (6n-e)/13$. If $|I| < (6n-e+1)/13$, then since G' has minimum degree 2, G' contains a 4-cycle (and hence G has a 4-cycle) or G' is a 5-cycle or the disjoint union of a 5-cycle and a 4-regular graph. If G' is a 5-cycle, then the neighbor of v with degree 4 has three neighbors in G', which is impossible in a triangle-free graph. The last case also cannot occur, since G is connected.

If $s(v) \geq 11$, then since $\delta(G') \leq 3$, it follows that $|I'| \geq (6(n-4)-(e-11))/13$ with $|I'| < (6(n-4) - (e-11) + 1)/13$ only if G' contains a 4-cycle. Thus $|I| \geq (6n-e)/13$ and $|I| < (6n - e + 1)/13$ only if G contains a 4-cycle. This establishes the theorem when $\delta(G) \leq 3$.

Case 4. G is 4-regular. If G contains a 4-cycle, then IS4 chooses v in the 4-cycle. In this case, the input to IS4 at the next step is a graph G' with minimum degree 2 and with $n-5$ vertices and $e-16$ edges. By induction, IS4 constructs an independent set I' in G' with $|I'| \geq (6(n-5) - (e-16) + 1)/13$. Thus $|I| \geq (6n-e)/13$. On the other hand, if G contains no 4-cycle, then IS4 chooses v arbitrarily. Then G' has no 4-cycle and cannot be 4-regular, and hence by induction $|I'| \geq (6(n-5) - (e-16) + 1)/13$. So again $|I| \geq (6n - e)/13$. This completes the proof. □

COROLLARY 2.3. *The algorithm* IS4 *constructs an independent set of order at least* $4n/13$ *in every triangle-free graph with* n *vertices and maximum degree at most* 4.

Proof. Since every graph with maximum degree 4 has at most $2n$ edges, $(6n - e)/13 \geq 4n/13$. □

The 4-regular Cayley graph on 13 vertices labeled 0 through 12 with edges determined by $ij \in E$ if and only if $(i-j) \bmod 13 \in \{1,5,8,12\}$ has independence 4. This graph also appears in [2] on page 105. This graph demonstrates that the bound is sharp.

3. General algorithm for triangle-free graphs. Now we will extend the algorithm to find independent sets in triangle-free graphs with no restriction on maximum degree. We begin by presenting an algorithm for triangle-free graphs with maximum degree at most 5. The extension to graphs with no restriction on maximum degree is then easy. Recall that $H(v)$ is the subgraph induced by the vertices of $V - N[v]$.

ALGORITHM INDEPENDENT SET FOR MAXIMUM DEGREE 5 (IS5)

Input: A triangle-free graph H with maximum degree at most 5.
Output: An independent set I.
Set $I = \emptyset$.
For each component G of H do
  Set V = vertex set of G.
  While $V \neq \emptyset$ do
    If $\Delta(G) \leq 4$ then
      apply IS4 to G to get an independent set I'.
      Set $I = I \cup I'$.
      Set $V = \emptyset$.
    else ($\Delta(G) = 5$)
      if $\delta(G) \leq 2$ then choose v with degree $\delta(G)$ and $s(v)$ as large as possible
      if $\delta(G) \geq 3$ then choose v with $s(v) \geq 6d(v) - 7$ (if it exists)
      else choose v with degree 3, with $s(v) = 10$ and $\delta(H(v)) \leq 2$.
    Set $I = I \cup \{v\}$.
    Set $V = V - N[v]$.
    Set G = the subgraph induced by V.

Before proving that the algorithm produces an independent set of the desired order, we show that vertices of the type required by the construction can always be found.

LEMMA 3.1. *In every triangle-free graph G with $\Delta(G) = 5$ and $\delta(G) \geq 3$, either there is a vertex v with $s(v) \geq 6d(v) - 7$ or there is a vertex v with degree 3, with $s(v) = 10$ and $\delta(H(v)) \leq 2$.*

Proof. If there is a vertex v with $d(v) = 5$ and $s(v) \geq 23$, then we are finished. So suppose v is a vertex of degree 5 with $s(v) \leq 22$. Then v has a neighbor w of degree 3 or 4.

Suppose $d(w) = 3$. Since $\delta(G) \geq 3$ and $v \in N(w)$, then $s(w) \geq 11$ and w is a vertex of the type desired.

Next suppose $d(w) = 4$. If $s(w) \geq 17$, we obtain the result. Otherwise, since $\delta(G) \geq 3$ and $v \in N(w)$, the vertex w has a neighbor u of degree 3. Since $w \in N(u)$ and $\delta(G) \geq 3$, $s(u) \geq 10$. Now either $s(u) \geq 11$ and the result follows, or $s(u) = 10$ and u has a neighbor x with $d(x) = 3$. Similarly, either $s(x) \geq 11$ or x has a neighbor $y \in H(u)$ with degree 3. Thus $\delta(H(u)) \leq 2$ and u is the vertex desired. □

Now we prove correctness of the algorithm.

THEOREM 3.2. *For every triangle-free graph G with n vertices, e edges, and $\Delta(G) \leq 5$, Algorithm* IS5 *(hereafter referred to as "IS5") constructs an independent set I with $|I| \geq (6n - e)/13$. Moreover, if $\delta(G) \leq 2$, then $|I| \geq (6n - e + 1)/13$.*

Proof. The proof is by induction on n. The statement is trivial to show for a graph with one vertex. We assume both statements are true for triangle-free graphs with maximum degree at most 5 and fewer than n vertices and let G be a triangle-free graph with n vertices, e edges, and maximum degree at most 5. If $\Delta(G) \leq 4$, then the result follows from Theorem 2.2. So suppose $\Delta(G) = 5$.

First we make a general observation to simplify what follows. If IS5 chooses a vertex v with $s(v) \geq 6d(v) - k$ for some positive integer k, then at the next stage the input to the algorithm is a graph G' with $n - d(v) - 1$ vertices and $e - s(v)$ edges. Now according to whether $\Delta(G') \leq 4$ or $\Delta(G') = 5$, we either use IS4 or the construction above. If $\Delta(G') \leq 4$, we apply Theorem 2.2. If $\Delta(G') = 5$, we apply the induction hypothesis. In either situation, at the next step we get an independent set I' in G' with $|I'| \geq (6(n - d(v) - 1) - (e - s(v)))/13 \geq (6n - e - (k + 6))/13$. So $|I| = |I'| + 1 \geq (6n - e + (7 - k))/13$. If $\delta(H(v)) \leq 2$, then since $G' = H(v)$, a similar argument shows that it follows from Theorem 2.2 or from the induction hypothesis that $|I| \geq (6n - e + (8 - k))/13$. In summary, when IS5 chooses v with $s(v) \geq 6d(v) - k$, we get $|I| \geq (6n - e + (7 - k))/13$, while if $\delta(H(v)) \leq 2$, we get $|I| \geq (6n - e + (8 - k))/13$.

Now we consider the various possibilities for minimum degree of G. If $\delta(G) = 0$, then IS5 chooses a vertex v of degree 0. Then $s(v) \geq 0 > 6d(v) - 6$; that is, we can take $k = 6$ in the above discussion. Thus IS5 constructs an independent set I with $|I| \geq (6n - e + 1)/13$. If $\delta(G) = 1$, then IS5 chooses a vertex v of degree 1. Then $s(v) \geq 1 \geq 6d(v) - 6$ and the result follows.

Suppose $\delta(G) = 2$. Then IS5 chooses a vertex v with $d(v) = 2$ and $s(v)$ as large as possible. If $s(v) \geq 6 = 6d(v) - 6$, then IS5 constructs I of appropriate order. Suppose $s(v) = 5$. Since $\delta(G) = 2$, then v must have a neighbor w of degree 2. Since $s(v)$ was chosen as large as possible, $s(w) \leq 5$ and hence w has a neighbor in $H(v)$ of degree 2. Thus $\delta(H(v)) \leq 2$ and it follows that $|I| \geq (6n - e + 1)/13$. If $s(v) = 4$ for every vertex v of degree 2, then G is a cycle or the disjoint union of cycles and components with minimum degree at least 3. Since IS5 is the same as IS4 for cycles, the result follows from Lemma 2.1 and an easy calculation.

Finally, if $\delta(G) \geq 3$, it follows from Lemma 3.1 that we can always find a vertex v with $s(v) \geq 6d(v) - 7$ or with $s(v) \geq 6d(v) - 8$ and with $\delta(H(v)) \leq 2$. (The vertex of degree 3 in Lemma 3.1 satisfies the latter condition.) In either case IS5 constructs an independent set I of order at least $(6n - e)/13$. □

Finally, we describe the general algorithm for triangle-free graphs with no restriction on maximum degree. The procedure is simply to remove vertices of degree 6 or more until a graph with maximum degree 5 is reached and then to apply IS5.

ALGORITHM INDEPENDENT SET (IS)

Input : A triangle-free graph G.
Output: An independent set I.
Set V = the vertex set of G.
While $\Delta(G) \geq 6$,
    choose a vertex v with degree 6 or more.
    Set $V = V - \{v\}$.
Perform IS5 on the subgraph induced by the vertices of V.

THEOREM 3.3. *For every triangle-free graph G,* IS *constructs an independent set I with $|I| \geq (6n - e)/13$.*

Proof. Again we prove correctness of the algorithm using induction. As usual, when G is a single vertex, the verification is trivial and we assume that the statement is true for graphs with fewer than n vertices. We only need establish that Algorithm IS (hereafter referred to as "IS") performs as desired when $\Delta(G) \geq 6$. Suppose IS chooses a vertex v with degree 6 or more. Then the input to the algorithm at the next stage is a graph G' with $n-1$ vertices and $e-d(v)$ edges. By the induction hypothesis, IS constructs an independent set I' with $|I'| \geq (6(n-1)-(e-d(v)))/13 \geq (6n-e)/13$. Since $|I| = |I'|$, the result is established. □

4. Algorithm complexity and conclusion. We describe the complexity of IS in terms of the order n of the graph. First we discuss the complexity of the Bondy–Locke algorithm. In [1] the algorithm to construct a bipartite subgraph of order at least $4|E|/5$ in triangle-free graphs with maximum degree at most 3 is shown to be $O(n^2)$. The construction of the independent set from the bipartite subgraph is clearly $O(n)$ as it involves a simple examination of degrees and membership in the sets forming the bipartition. Thus the Bondy–Locke algorithm is $O(n^2)$.

The procedure in IS4 that requires determining whether a 4-regular graph contains a 4-cycle can be seen to be $O(n)$ as follows: to find a 4-cycle in a graph with maximum degree Δ, list all paths at v with length 2 for each vertex v. Then check whether two of these paths end at the same vertex. This requires $O(\Delta^2)$ steps to list the paths and $O(\log \Delta)$ steps to examine $O(\Delta(\Delta-1))$ vertices for duplicates; thus determining whether the graph containing a 4-cycle requires $O(\Delta^3 n)$ steps. Since $\Delta \leq 4$ in IS4, this procedure is $O(n)$. The other expensive procedures in IS4, IS5, and IS require only the calculation of degrees of vertices, the maximum and minimum degree of the graph, and the components of the graph, all of which can be done in at most $O(n^2)$ steps. Thus each of the algorithms IS4, IS5, and IS has complexity $O(n^2)$.

In conclusion, we observe that the performance of the algorithm could be improved by replacing the Bondy–Locke algorithm, in Case 3 of Theorem 2.2, with an algorithm that finds an independent set of order at least $5n/14$ in triangle-free graphs with maximum degree 3. Staton [9] showed that this is a tight lower bound for independence in such graphs. In [6] a stronger result (similar to the result in [5] upon which this algorithm is based) is proved, from which Staton's lower bound follows. Designing an algorithm based on the proof in [6] appears to be difficult. However, Griggs and Murphy [4] have recently devised an algorithm to construct an independent set of order at least $5(n-1)/14$ in 3-regular connected triangle-free graphs and greater than $5n/14$ in connected nonregular triangle-free graphs with maximum degree 3. A more recent algorithm of Fraughnaugh and Locke [3] constructs an independent set of order at least $(11n-4)/30$ in connected triangle-free graphs with maximum degree 3, including the cubics. This newer Fraughnaugh–Locke algorithm provides results at least as good as those guaranteed by the Bondy–Locke algorithm used in this proof. The Bondy–Locke algorithm produces an independent set of cardinality at least $\lceil 7n/20 \rceil$. The Griggs–Murphy algorithm produces an independent set of cardinality at least $\lceil 5(n-1)/14 \rceil$. The Fraughnaugh–Locke algorithm produces an independent set of cardinality at least $\lceil (11n-4)/30 \rceil$. When the input is a cubic triangle-free graph, each of these algorithms produces an independent set of cardinality at least $\lceil (9n+2)/26 \rceil$ as needed in Case 3—with the possible exception of the cases $n = 6, 12, 26$ for the Griggs–Murphy algorithm. Furthermore, each of these three algorithms is polynomial time.

REFERENCES

[1] J. A. BONDY AND S. C. LOCKE, *Largest bipartite subgraphs in triangle-free graphs with maximum degree three*, J. Graph Theory, 10 (1986), pp. 477–504.

[2] J. A. BONDY AND U. S. R. MURTY, *Graph Theory with Applications*, American Elsevier, New York, NY, 1976.

[3] K. FRAUGHNAUGH AND S. C. LOCKE, 11/30 *(finding large independent sets in connected triangle-free 3-regular graphs)*, J. Combin. Theory Ser. B, 65 (1995), pp. 51–72.

[4] J. R. GRIGGS AND O. MURPHY, *Edge density and independence ratio in triangle-free graphs with maximum degree three*, Discrete Math., 152 (1996), pp. 157–170.

[5] K. FRAUGHNAUGH JONES, *Independence in graphs with maximum degree four*, J. Combin. Theory Ser. B, 37 (1984), pp. 254–269.

[6] ———, *Size and independence in triangle-free graphs with maximum degree three*, J. Graph Theory, 14 (1990), pp. 525–535.

[7] D. L. KREHER AND S. P. RADZISZOWSKI, *Minimum triangle-free graphs*, Ars Combin., 31 (1991), pp. 65–92.

[8] S. C. LOCKE, *Bipartite density and the independence ratio*, J. Graph Theory, 10 (1986), pp. 47–53.

[9] W. STATON, *Some Ramsey-type numbers and the independence ratio*, Trans. Amer. Math. Soc., 256 (1979), pp. 353–370.